Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομέας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Κινητων Επικοινωνιων

Ηλεκτρονικός Ιατρικός Φάκελος σε Κινητό Τηλέφωνο και Εφαρμογές του

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτης Ν. Τεμπριώτης

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Κινητων Επικοινωνιων

Ηλεκτρονικός Ιατρικός Φάκελος σε Κινητό Τηλέφωνο και Εφαρμογές του

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτης Ν. Τεμπριώτης

Επιβλέπων: Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ..... 2010.

....
Φίλιππος Κωνσταντίνου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2010

.....

Παναγιώτης Ν. Τεμπριώτης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας είναι η υλοποίηση ηλεκτρονικού ιατρικού φακέλου σε κινητό τηλέφωνο. Ο ηλεκτρονικός ιατρικός φάκελος θα βρίσκεται στο κινητό τηλέφωνο του ασθενούς με σκοπό να διευκολύνει το ιατρικό προσωπικό και τον ιατρό κατά την εξέταση του ασθενή. Η εφαρμογή στο κινητό τηλέφωνο θα παρέχει στον ασθενή, πέραν από τη δυνατότητα ανάγνωσης του ιατρικού φακέλου τη δυνατότητα τροποποίησης των δημογραφικών του στοιχείων. Παράλληλα με την ασύρματη σύνδεση του κινητού τηλεφώνου με ηλεκτρονικό υπολογιστή, δίνεται η δυνατότητα στον εκάστοτε ιατρό τροποποίησης του ιατρικού φακέλου καθώς και ενημέρωσης του με απλό γρήγορο και άμεσο τρόπο.

Στο πρώτο μέρος περιγράφονται οι υπάρχουσες τεχνολογίες κινητών επικοινωνιών.

Στο δεύτερο μέρος γίνεται ανάλυση των τεχνολογιών καθώς και του τρόπου εφαρμογής των τεχνολογιών αυτών στην υλοποίηση του ηλεκτρονικού ιατρικού φακέλου.

Στο τρίτο μέρος περιγράφεται η υλοποίηση του ηλεκτρονικού ιατρικού φακέλου. Συγκεκριμένα περιγράφονται οι εφαρμογές σε κινητό τηλέφωνο και ηλεκτρονικό υπολογιστή.

Λέξεις Κλειδιά

J2ME, Java, applet, MIDP, tomcat, Bluetooth, ιατρικός φάκελος, κινητό

Abstract

The purpose of this thesis is the implementation of electronic medical record on a mobile phone. The electronic medical record will be on the patient's phone to facilitate the medical staff and doctors in the examination of the patient. The implementation of the mobile phone will provide the patient the ability to read the medical record and to modify his demographic elements. With the wireless connection of the mobile phone with the computer, it is feasible to change the medical record information in a simple quick and direct manner.

The first part describes the existing mobile technologies.

The second part analyzes the technologies described and how the application makes use of these technologies in the implementation of electronic medical record.

The third part describes the implementation of electronic medical record. Specifically the applications in the mobile phone and computer are been described.

Keywords

J2ME, Java, applet, MIDP, tomcat, Bluetooth, medical record, mobile phone

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Φίλιππο Κωνσταντίνου, για την πολύτιμη συμβολή του στην ολοκλήρωση της παρούσης εργασίας.

Παράλληλα ευχαριστώ θερμά τον κ. Δημήτρη Νικητόπουλο, υποψήφιο διδάκτορα, για την βοήθεια και την υποστήριξή του καθ' όλη τη διάρκεια της εκπόνησης της διπλωματικής εργασίας.

Περιεχόμενα

Ευρετήριο Σχημάτων

Ευρετήριο Πινάκων

Μέρος Α

1. Γενική θεωρία

Οι κινητές επικοινωνίες έκαναν για πρώτη φορά την εμφάνιση τους τη δεκαετία του 1940, όταν τα ραδιοτηλέφωνα υπήρξαν η πρώτη μορφή διαπροσωπικής επικοινωνίας.

Οι χρήστες των ραδιοτηλεφώνων πατώντας ένα κουμπί είχαν τη δυνατότητα να μιλήσουν μεταξύ τους ανταλλάσσοντας πληροφορίες. Η χρήση τους όμως ήταν περιορισμένη και εντοπισμένη σε συγκεκριμένες εφαρμογές (στρατιωτικές, επικοινωνίες πλοίων, αεροσκαφών). Η πλειονότητα των πολιτών δεν είχε τη δυνατότητα να χρησιμοποιήσει τις εφαρμογές αυτές.

Μόλις το 1961 το παράρτημα της Σουηδικής εταιρείας Ericsson με την επωνυμία Svenska Radio Aktiebolaget (SRA) παρήγαγε για πρώτη φορά διεθνώς εξοπλισμό κινητών επικοινωνιών για εμπορική χρήση, απευθυνόμενο στο ευρύ κοινό.

Το πρώτο δίκτυο κινητών επικοινωνιών που καταγράφεται διεθνώς είναι αυτό που ανέπτυξε η εταιρεία Bell Systems, στην περιοχή της Πενσιλβανία των ΗΠΑ, μεταξύ 1962 - 1964 και του έδωσε την ονομασία Improved Mobile Telephone Service (IMTS).

Το δίκτυο αυτό ήταν το πρώτο στον κόσμο που επέτρεπε την αμφίδρομη επικοινωνία (full duplex) και έτσι οι συνομιλητές δεν αναγκάζονταν να πιέζουν πλήκτρο στο τερματικό τους για να μπορούν να ομιλούν. Το σύστημα αυτό αναπτύχθηκε στις ΗΠΑ και στον Καναδά, παρέχοντας τηλεπικοινωνιακές υπηρεσίες μέχρι και τα μέσα της δεκαετίας του 1980.

Στην Ιαπωνία το 1967 η εταιρεία Nippon Telegraph and Telephone company (NTT) άρχισε να αναπτύσσει δίκτυο σε όλη την Ιαπωνία σε συχνότητες στην περιοχή των 800 MHz.

Μόλις το 1969 η εταιρεία Bell Systems παρείχε δίκτυο κινητών επικοινωνιών στο ευρύ κοινό, το οποίο λειτουργούσε σε συχνότητες στα 450 MHz και με δυνατότητα πραγματοποίησης μεταπομπών (handovers) για πρώτη φορά. Το δίκτυο αυτό ήταν βασισμένο στο πρωτόκολλο IMTS. Στις 17 Οκτωβρίου του 1973 η εταιρεία Motorola παρουσίασε το πρώτο φορητό τηλέφωνο που μπορούσε να κρατηθεί με το ένα χέρι και να λειτουργήσει αυτοτελώς, χωρίς να χρειάζεται ξεχωριστά συστήματα τροφοδοσίας. Το βήμα αυτό ήταν και καθοριστικό για την μετέπειτα εξέλιξη των κινητών επικοινωνιών.

Το 1978 τόσο η AT&T (όπως μετονομάστηκε η Bell South), όσο και η εταιρεία Bahrain Telephone Company, ανέπτυξαν και λειτούργησαν εμπορικά δίκτυα κινητών

επικοινωνιών βασισμένα στο πρότυπο Advanced Mobile Phone Service (AMPS), που είναι ένα αναλογικό σύστημα κινητών επικοινωνιών και πρόδρομος του Digital Advanced Mobile Phone Service (D-AMPS) και το οποίο λειτουργεί μέχρι και σήμερα στις ΗΠΑ.

Στις αρχές της δεκαετίας του 1980 αρκετές Ευρωπαϊκές χώρες (Νορβηγία, Σουηδία, Φιλανδία, Βέλγιο, Ολλανδία, Μεγάλη Βρετανία, EIPE) είχαν αναπτύξει δίκτυα κινητών επικοινωνιών βασισμένα σε πρότυπα αναλογικών επικοινωνιών (NMT 450 και 900) τα οποία ήταν μη συμβατά μεταξύ τους. Την ίδια στιγμή οι ανάγκες για τηλεπικοινωνιακές υπηρεσίες αυξάνονταν ραγδαία. Εξαιτίας του γεγονότος αυτού, καθώς και της ανάγκης συμβατών δικτύων σε όλη την Ευρώπη, η Ευρωπαϊκή Επιτροπή Τηλεπικοινωνιών και Ταχυδρομείων (CEPT) ίδρυσε μια ομάδα εμπειρογνωμόνων επιφορτισμένη με την έκδοση των προδιαγραφών ενός κοινού συστήματος κινητών επικοινωνιών για όλη τη Δυτική Ευρώπη. Η ομάδα αυτή ονομάστηκε Groupe Spéciale Mobile, από τα αρχικά γράμματα της οποίας προέκυψε για πρώτη φορά το όνομα GSM. Η συντομογραφία αυτή έγινε όμως γνωστή με τον όρο Global System for Mobile Communications (GSM). Από το σημείο αυτό ξεκινά η εξέλιξη του πλέον διαδεδομένου και επιτυχημένου συστήματος κινητών επικοινωνιών, του γνωστού σε όλους GSM.

Το 1982 η CEPT αρχικοποίησε το νέο σύστημα (GSM), ενώ το 1986 πραγματοποιήθηκαν οι πρώτες δοκιμές του συστήματος στο Παρίσι. Το 1987 καθορίσθηκαν οι συχνότητες λειτουργίας του συστήματος στην περιοχή των 900 MHz και το 1988 δημιουργήθηκε το European Telecommunications Standards Institute (ETSI), το οποίο επωμίστηκε την ευθύνη της έκδοσης προδιαγραφών και συστάσεων για το GSM, που εξεδόθησαν για πρώτη φορά το 1989.

Στις 1-7-1991 πραγματοποιήθηκε το πρώτο επίσημο εμπορικό τηλεφώνημα από δίκτυο GSM. Το 1992 οι Αυστραλοί πάροχοι κινητών επικοινωνιών έγιναν οι πρώτοι μη Ευρωπαίοι οι οποίοι εξεδήλωσαν επιθυμία να αναπτύξουν το νέο σύστημα, ενώ την ίδια χρονιά καθορίστηκαν και νέες συχνότητες στην περιοχή των 1800 MHz. Στην περιοχή αυτή συχνοτήτων αναπτύχθηκε και το σύστημα PCS στις ΗΠΑ, που αποτελεί σύστημα συμβατό (μερικώς) με το GSM.

Έτσι ξεκίνησε η ανάπτυξη των δικτύων GSM, που είναι πλέον απο τα πιο διαδεδομένα δίκτυα διεθνώς διαθέτοντας εκατοντάδες εκατομμύρια συνδρομητές και αποτελώντας ταυτόχρονα την σταθερή βάση για την ανάπτυξη των δικτύων νέας γενιάς (3G/UMTS), των οποίων η εξάπλωση ξεκίνησε ήδη από το 2002.

Τι είναι όμως και από τι αποτελείται ένα δίκτυο GSM;

Ένα δίκτυο τύπου GSM αποτελείται από τα ακόλουθα υποσυστήματα:

- Το MSC (Mobile Switching Center) είναι ψηφιακό τηλεπικοινωνιακό σύστημα το οποίο χρησιμοποιείται για την διαχείριση των κλήσεων φωνής των συνδρομητών στο GSM δίκτυο. Δίπλα σε κάθε MSC βρίσκεται και μια VLR (Visitor Location Register) που είναι προσωρινή βάση δεδομένων και περιέχει τις υπηρεσίες του συνδρομητή, αλλά και την ευρύτερη περιοχή που βρίσκεται.
- Το SGSN (Serving GPRS Support Node) είναι ψηφιακό τηλεπικοινωνιακό σύστημα το οποίο χρησιμοποιείται για την διαχείριση των κλήσεων δεδομένων των συνδρομητών στο GSM-GPRS δίκτυο. Δίπλα σε κάθε SGSN βρίσκεται επίσης και η αντίστοιχη GPRS VLR.
- Το BSC (Base Station Controller): είναι ένα ψηφιακό κέντρο και χρησιμοποιείται για να ελέγχει τους Σταθμούς Βάσης και την διασύνδεσή τους με το MSC.
- Το BTS (Base Transceiver Station) ή Σταθμός Βάσης (ΣΒ), χρησιμοποιείται για την διασύνδεση των κινητών τηλεφώνων με το υπόλοιπο δίκτυο. Περιέχει πομποδέκτες (συχνότητας : 900-1800 MHz). Επίσης στα BTS συγκαταλέγονται και οι μικροκυματικές κατευθυντικές ζεύξεις είτε τα μισθωμένα κυκλώματα που χρησιμοποιούνται για τη μεταφορά των δεδομένων του κάθε σταθμού βάσης στα αντίστοιχα κέντρα (BSC – MSC) και τα οποία αποτελούν το δίκτυο μετάδοσης (transmission network).
- Η HLR (Home Location Register) είναι μόνιμη βάση δεδομένων με όλες τις υπηρεσίες των συνδρομητών πχ. προωθήσεις, Data, Fax, GPRS.

2. Κινητή Τηλεφωνία

Η κινητή τηλεφωνία είναι η παροχή τηλεφωνικών υπηρεσιών σε τηλέφωνα τα οποία μπορούν να μετακινούνται ελεύθερα και όχι να παραμένουν σταθερά σε μία θέση. Τα κινητά τηλέφωνα συνδέονται με ένα επίγειο κυψελοειδές δίκτυο των σταθμών βάσης (περιοχές κυψελών), ενώ τα δορυφορικά τηλέφωνα συνδέονται με δορυφόρους εν τροχιά. Τα δύο δίκτυα είναι διασυνδεδεμένα με το δημόσιο τηλεφωνικό δίκτυο μεταγωγής (PSTN), επιτρέποντας σε οποιοδήποτε τηλέφωνο στον κόσμο να κληθεί. Το 2008 υπήρχαν 4,1 δισεκατομμύρια συνδρομές κινητής τηλεφωνίας στον κόσμο.

3. Κυψελοειδές δίκτυο

Ένα κυψελοειδές δίκτυο είναι ένα ραδιοφωνικό δίκτυο που αποτελείται από ένα αριθμό κυψελών, με την κάθε μια να εξυπηρετείται από τουλάχιστον ένα σταθερού-

σημείου πομποδέκτη γνωστό ως τοποθεσία κυψέλης ή σταθμό βάσης. Όταν ενώνονται τα κύτταρα αυτά παρέχουν ραδιοκάλυψη σε μια ευρεία γεωγραφική περιοχή. Αυτό δίνει τη δυνατότητα σε μεγάλο αριθμό φορητών πομποδεκτών (κινητά τηλέφωνα, συσκευές τηλεειδοποίησης, κλπ.) να επικοινωνούν μεταξύ τους με σταθερούς πομποδέκτες και τηλέφωνα οπουδήποτε στο δίκτυο, μέσω των σταθμών βάσης, έστω και αν ορισμένοι από τους πομποδέκτες κινούνται μέσω περισσότερων της μιας κυψέλης κατά τη διάρκεια μετάδοσης. Τα κυψελοειδή δίκτυα προσφέρουν μια σειρά από πλεονεκτήματα έναντι των εναλλακτικών λύσεων:

- Αύξηση της χωρητικότητας.
- Μειωμένη κατανάλωση ενέργειας.
- Μεγαλύτερη επιφάνεια κάλυψης.
- Μειωμένες παρεμβολές από άλλα σήματα.

4. GSM

Το GSM (Global System for Mobile Communications: αρχική ονομασία Groupe Spécial Mobile) είναι το πιο δημοφιλές πρότυπο για τα συστήματα κινητής τηλεφωνίας στον κόσμο. Η GSM Association, ο οργανισμός προώθησης του GSM στην βιομηχανία των υπηρεσιών κινητής τηλεφωνίας και κατασκευαστές τηλεφώνων, υπολογίζει ότι το 80% της παγκόσμιας αγοράς κινητής τηλεφωνίας χρησιμοποιεί το πρότυπο αυτό. Το GSM χρησιμοποιείται από πάνω από 3 δισεκατομμύρια ανθρώπους σε περισσότερες από 212 χώρες και εδάφη. Η πανταχού παρουσία του δίνει τη δυνατότητα διεθνούς περιαγωγής, ρυθμίσεις μεταξύ παρόχων κινητής τηλεφωνίας, παρέχοντας στους συνδρομητές τη χρήση των τηλεφώνων τους σε πολλά μέρη του κόσμου. Το GSM διαφέρει από τις τεχνολογίες προκατόχους του στο ότι τόσο η σηματοδότηση όσο και τα κανάλια ομιλίας είναι ψηφιακά και έτσι GSM θεωρείται *δεύτερης γενιάς* (2G) δίκτυο κινητής τηλεφωνίας. Αυτό διευκολύνει επίσης την ευρεία υλοποίηση των εφαρμογών επικοινωνίας δεδομένων στο σύστημα. Η πανταχού παρουσία της εφαρμογής του προτύπου GSM έχει ένα πλεονέκτημα τόσο τους καταναλωτές, οι οποίοι μπορούν να επωφεληθούν από τη δυνατότητα να περιπλανηθούν και να αλλάξουν μεταφορείς χωρίς να αλλάξουν τηλέφωνα, καθώς επίσης και για τους παρόχους δικτύων, οι οποίοι μπορούν να επιλέξουν εξοπλισμό από πολλούς προμηθευτές εξοπλισμού GSM. Το GSM πρωταγωνίστησε, επίσης, στην χαμηλού κόστους υλοποίηση της υπηρεσίας σύντομων μηνυμάτων (SMS), η οποία έκτοτε έχει υποστηριχθεί και από άλλα πρότυπα κινητής τηλεφωνίας. Το πρότυπο περιλαμβάνει και ένα παγκόσμιο τηλεφωνικό αριθμό

έκτακτης ανάγκης (112). Νεώτερες εκδόσεις του προτύπου ήταν συμβατές με το αρχικό σύστημα GSM. Για παράδειγμα, η έκδοση '97 του προτύπου προσέθεσε δυνατότητες μεταφοράς δεδομένων μέσω του General Packet Radio Service (GPRS). Η έκδοση '99 παρουσιάζει υψηλότερη ταχύτητα μετάδοσης δεδομένων χρησιμοποιώντας Enhanced Data Rates for GSM Evolution (EDGE).

5. SMS

Το **Short Message Service (S**MS) είναι μια συνιστώσα υπηρεσιών επικοινωνίας του GSM συστήματος κινητής τηλεφωνίας, χρησιμοποιώντας τυποποιημένα πρωτόκολλα επικοινωνίας που επιτρέπουν την ανταλλαγή σύντομων μηνυμάτων κειμένου ανάμεσα σε κινητά τηλέφωνά. Τα SMS μηνύματα κειμένου είναι η πιο ευρέως χρησιμοποιούμενη εφαρμογή των δεδομένων στον κόσμο, με 2,4 δισεκατομμύρια ενεργούς χρήστες, ή το 74% του συνόλου των συνδρομητών κινητής τηλεφωνίας. Ο όρος SMS χρησιμοποιείται ως συνώνυμο για όλους τους τύπους των μικρών μηνυμάτων κειμένου, καθώς όπως και η ίδια η δραστηριότητα του χρήστη, σε πολλά μέρη του κόσμου. Το SMS, όπως χρησιμοποιείται στις σύγχρονες συσκευές είχε αρχικά οριστεί ως τμήμα σειράς προτύπων του GSM το 1985 ως μέσο για την αποστολή μηνυμάτων μέχρι 160 χαρακτήρες, προς και από κινητά τηλέφωνα GSM. Έκτοτε, η υποστήριξη για την υπηρεσία έχει επεκταθεί και σε άλλες κινητές τεχνολογίες, όπως τα δίκτυα CDMA ANSI και Digital AMPS, καθώς και δορυφορικά και επίγεια δίκτυα. Τα περισσότερα μηνύματα SMS είναι από κινητό προς κινητό mobile-to-mobile μηνύματα κειμένου, αν και το πρότυπο υποστηρίζει επίσης και άλλους τύπους μετάδοσης μηνυμάτων επίσης.

5.1 SMS σε GSM δίκτυα

Η υπηρεσία *Short Message Service - Point to Point (SMS-PP)* ορίζεται στη σύσταση GSM 03,40. Η σύσταση GSM 03,41 καθορίζει το Short *Message Service - Cell Broadcast (SMS-CB),* το οποίο επιτρέπει μηνύματα για διαφήμιση, ενημέρωση του κοινού, κλπ. να μεταδίδονται σε όλους τους χρήστες κινητών τηλεφώνων σε μια οριοθετημένη γεωγραφική περιοχή.

Τα μηνύματα που αποστέλλονται σε Short Message Service Center (SMSC), το οποίο παρέχει ένα μηχανισμό “αποθήκευσης και προώθησης”. Επιχειρεί να στείλει μηνύματα προς τους αποδέκτες της, SMSC του. Αν ο αποδέκτης δεν είναι προσβάσιμος, το SMSC βάζει το μήνυμα σε ουρά για να το ξαναστείλει αργότερα.

Ορισμένα SMSCs επίσης παρέχουν μια “forward and forget" επιλογή, όπου η μετάδοση δοκιμάζεται μόνο μία φορά. Η παράδοση των μηνυμάτων είναι "best effort", οπότε δεν υπάρχει καμία εγγύηση ότι ένα μήνυμα όντως θα παραδοθεί στον αποδέκτη του, αλλά η καθυστέρηση και ολική απώλεια ενός μηνύματος είναι σπάνια.

Οι χρήστες μπορούν να ζητήσουν τις αναφορές παράδοσης για να επιβεβαιωθούν ότι τα μηνύματα φθάνουν στους αποδέκτες, είτε μέσω των ρυθμίσεων SMS από τα πιο σύγχρονα τηλέφωνα, ή με την προσθήκη κάθε μήνυμα με το * 0 # ή * N #.

5.2 Μέγεθος Μηνύματος

Η μετάδοση των σύντομων μηνυμάτων μεταξύ των SMSC και του κινητού τηλεφώνου γίνεται κάθε φορά που χρησιμοποιείται το Mobile Application (MAP) του SS7 πρωτοκόλλου.

Τα μηνύματα στέλνονται με τις mo-MAP και mt-ForwardSM λειτουργίες, των οποίων το μήκος του ωφέλιμου φορτίου είναι περιορισμένο από τους περιορισμούς του πρωτοκόλλου σηματοδοσίας σε ακριβώς 140 οκτάδες – octes (140octes = 140 * 8 bits = 1120 bits). Αυτό συνεπάγεται ότι το μήνυμα μπορεί να περιέχει 140 χαρακτήρες με αλφάβητο 8-bit δεδομένων, 70 χαρακτήρες με αλφάβητο 16-bit UTF-16 ή 160 χαρακτήρες με αλφάβητο 7-bit που είναι και το προεπιλεγμένο για τα δίκτυα GSM. Η δρομολόγηση των δεδομένων και άλλων μετα-δεδομένων είναι πρόσθετη προς το ωφέλιμο φορτίο του μηνύματος.

Μεγαλύτερα μηνύματα (multipart-SMS) μπορούν να σταλούν με πολλαπλά μηνύματα, οπότε κάθε μήνυμα θα ξεκινήσει με τα δεδομένα των χρηστών κεφαλίδα (UDH) που περιέχει ο κατακερματισμός των πληροφοριών. Δεδομένου ότι UDH αποτελεί μέρος του ωφέλιμου φορτίου, ο αριθμός των διαθέσιμων χαρακτήρων ανά τμήμα είναι μικρότερος: 153 για την κωδικοποίηση των 7-bit, 133 για την κωδικοποίηση των 8-bit και 67 για την κωδικοποίηση 16-bit. Η συσκευή του παραλήπτη είναι τότε υπεύθυνη για την επανασύνδεσή του μηνύματος και την παρουσίαση στο χρήστη ως ένα μεγάλο μήνυμα.

6. MMS

Το Multimedia Messaging Service, ή MMS, είναι ένας κλασικός τρόπος για αποστολή μηνυμάτων που περιλαμβάνουν περιεχομένου πολυμέσων (multimedia) από και προς κινητά τηλέφωνα. Επεκτείνει τη βασική SMS (Short Message Service)

δυνατότητα η οποία επιτρέπει την ανταλλαγή μηνυμάτων κειμένου μόνο μέχρι 160 χαρακτήρες.

Η πιο δημοφιλής χρήση του MMS είναι η αποστολή φωτογραφιών από κινητά τηλέφωνα εξοπλισμένα με φωτογραφική μηχανή. Είναι επίσης δημοφιλής ως μια μέθοδος παράδοσης ειδήσεων και περιεχομένου ψυχαγωγίας, συμπεριλαμβανομένων βίντεο, φωτογραφιών και ringtones .

Το πρότυπο αναπτύχθηκε από την Open Mobile Alliance (OMA), αν και κατά τη διάρκεια της ανάπτυξης ήταν μέρος των ομάδων 3GPP και WAP.

7. Bluetooth

Το Bluetooth είναι ένα ιδιόκτητο ανοικτό ασύρματο πρωτόκολλο για την ανταλλαγή δεδομένων σε μικρές αποστάσεις (χρησιμοποιώντας ραδιοκύματα μικρής διάρκειας) από σταθερές και κινητές συσκευές, δημιουργώντας δίκτυα προσωπικής περιοχής (Personal Area Networks PANs). Αρχικά είχε σχεδιαστεί ως μια ασύρματη εναλλακτική λύση για τα RS-232 καλώδια δεδομένων. Μπορεί να συνδέσει διάφορες συσκευές, ξεπερνώντας τα προβλήματα συγχρονισμού.

Το Bluetooth χρησιμοποιεί μια τεχνολογία που ονομάζεται Frequency-hopping spread spectrum (FHSS), η οποία κομματιάζει τα δεδομένα που αποστέλλονται και μεταδίδει τα κομμάτια και μέχρι 79 ζώνες του 1 MHz πλάτους στην περιοχή 2402-2480 Mhz. Αυτή είναι η παγκοσμίως μη αδειοδοτημένη βιομηχανική, επιστημονική και ιατρική (ISM) 2,4 GHz μικρής εμβέλειας ραδιοσυχνοτήτων μπάντα.

Στο βασικό ρυθμό (Basic Rate - BR) λειτουργίας, η αρθρωμάτωση (modulation) είναι Gaussian frequency-shift keying (GFSK). Μπορεί να επιτύχει ένα ρυθμό μετάδοσης δεδομένων του 1 Mbit/s. Στον εκτεταμένο ρυθμό δεδομένων (Extended Data Rate EDR) χρησιμοποιούνται π/4-DQPSK και 8DPSK, δίνοντας 2, και 3 Mbit/s, αντίστοιχα.

Το Bluetooth είναι ένα πρωτόκολλο με βάση πακέτα με τη δομή αφέντη-σκλάβου (master-slave). Ένας αφέντης μπορεί να επικοινωνεί με έως και 7 σκλάβους σε ένα piconet: όλες οι συσκευές μοιράζονται το ρολόι του αφέντη για συγχρονισμό. Η ανταλλαγή πακέτων βασίζεται στο βασικό ρολόι, που ορίζεται από τον αφέντη, το οποίο κτυπάει σε διαστήματα 312,5 μs . Δύο κτύποι ρολογιού συνθέτουν μια υποδοχή (slot) των 625 μs: Δύο υποδοχές συνθέτουν ένα ζευγάρι υποδοχών των 1250 μs.

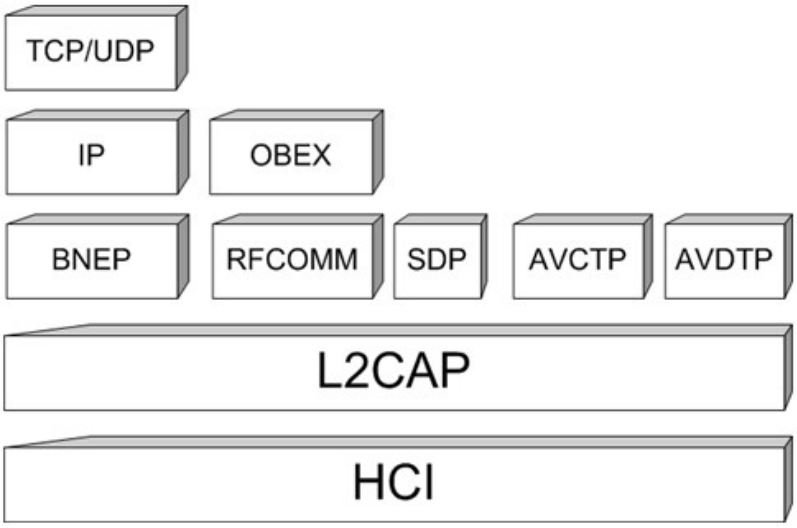
Στην απλή περίπτωση της απλής-χρονοθυρίδας (single-slot) ο αφέντης μεταδίδει πακέτα σε ζυγές χρονοθυρίδες και δέχεται, σε μονές χρονοθυρίδες: Ο σκλάβος,

αντίθετα, λαμβάνει σε ζυγές χρονοθυρίδες και μεταδίδει σε μονές χρονοθυρίδες. Τα πακέτα μπορεί να έχουν μέγεθος 1, 3 ή 5 υποδοχών, αλλά σε κάθε περίπτωση, η μετάδοση του αφέντη θα ξεκινήσει σε ζυγές χρονοθυρίδες και η μετάδοση του σκλάβου σε μονές χρονοθυρίδες.

Το Bluetooth παρέχει έναν ασφαλή τρόπο για τη σύνδεση και ανταλλαγή πληροφοριών ανάμεσα σε συσκευές, όπως φαξ, κινητά τηλέφωνα, τηλέφωνα, φορητοί υπολογιστές, προσωπικοί υπολογιστές, εκτυπωτές, Global Positioning System (GPS) δέκτες, ψηφιακές φωτογραφικές μηχανές και βίντεο κονσόλες παιχνιδιών .

Οι προδιαγραφές του Bluetooth αναπτύσσονται και αδειοδοτούνται από το Bluetooth Special Interest Group (SIG). Το Bluetooth SIG αποτελείται από εταιρείες στους τομείς των τηλεπικοινωνιών, της πληροφορικής, της δικτύωσης, καθώς και των ηλεκτρονικών ειδών ευρείας κατανάλωσης. Για να διατεθεί στην αγορά μια συσκευή ως συσκευή Bluetooth, πρέπει να συμφωνεί με τα πρότυπα που ορίζονται από το SIG.

7.1 Πρωτόκολλα Bluetooth



Σχήμα 1: Πρωτόκολλα Bluetooth

7.1.1 Στοίβα Ελεγκτή (Controller Stack)

Asynchronous Connection Less link(ACL)

Ο κανονικός τύπος ραδιοζεύξης που χρησιμοποιείται για πακέτα δεδομένων χρησιμοποιώντας ένα περισκοπικό (polling) σύστημα TDMA για αυθαίρετη πρόσβαση μπορεί να μεταφέρει πολλά διαφορετικού τύπου πακέτα, τα οποία διακρίνονται στα:

- μήκος: 1, 3, ή 5 χρονοθυρίδες ανάλογα με το απαιτούμενο μέγεθος του ωφέλιμου φορτίου.
- forward error correction: προαιρετικά μειώνει το ποσοστό των δεδομένων προς όφελος της αξιοπιστίας
- διαμόρφωση (modulation): EDR - enhanced data rate - τα πακέτα επιτρέπουν έως και τριπλάσιο ρυθμό δεδομένων, χρησιμοποιώντας μια διαφορετική διαμόρφωση RF για το ωφέλιμο φορτίο.

Μια σύνδεση πρέπει να στηθεί ρητά και να γίνει αποδεκτή μεταξύ δύο συσκευών πριν να αρχίσει η μεταφορά των πακέτων.

Τα πακέτα ACL αναμεταδίδονται αυτόματα, αν δεν αναγνωριστούν (unacknowledged), επιτρέποντας τη διόρθωση μιας ραδιοζεύξης που υπόκειται σε παρεμβολές. Για ισοχρονισμένα δεδομένα, ο αριθμός των αναμεταδόσεων μπορεί να περιορίζεται από ένα εξωχρονισμό (timeout). Αλλά χωρίς τη χρησιμοποίηση αναμετάδοση L2CAP και λειτουργία έλεγχου ροής ή EL2CAP, ένα υψηλότερο στρώμα πρέπει να χειρίζεται την απώλεια πακέτων.

Οι ζεύξεις ACL αποσυνδέονται αν δεν έχουν λάβει τίποτα για ένα εποπτικό εξωχρονισμό. Ο προκαθορισμένος χρόνος εξωχρονισμού είναι 20 δευτερόλεπτα, αλλά αυτό μπορεί να τροποποιηθεί από τον αφέντη.

Synchronous connection oriented (SCO) link

Ο τύπος της ριαδιοζεύξης που χρησιμοποιείται για τα δεδομένα φωνής. Μια ζεύξη SCO είναι ένα σύνολο δεσμευμένων χρονοθυρίδων σε μια υπάρχουσα ζεύξη ACL. Κάθε συσκευή εκπέμπει κωδικοποιημένα δεδομένα φωνής στη δεσμευμένη χρονοθυρίδα. Δεν υπάρχουν αναμεταδόσεις, αλλά μπορεί να εφαρμοστεί προαιρετικά forward error correction.

Οι ζεύξεις enchanced SCO (eSCO) χρησιμοποιούν αναμεταδόσεις για την επίτευξη αξιοπιστίας και συμπεριφέρονται περισσότερο όπως τα γενικά δεδομένα.

Link management protocol (LMP)

Χρησιμοποιείται για τον έλεγχο της ζεύξης μεταξύ δύο συσκευών και εφαρμόζεται στον ελεγκτή.

Host/controller interface (HCI)

Τυποποιημένη επικοινωνία μεταξύ της στοίβα του ξενιστή (πχ. ηλεκτρονικού υπολογιστή ή λειτουργικού συστήματος κινητού τηλεφώνου) και του ελεγκτή (Bluetooth IC). Το πρότυπο αυτό επιτρέπει στην στοίβα του ξενιστή ή στον ελεγκτή IC να ανταλλαγούν με την ελάχιστη δυνατή προσαρμογή.

Υπάρχουν διάφορα πρότυπα HCI για το στρώμα μεταφοράς, με το καθένα από αυτά να χρησιμοποιεί μια διαφορετική διεπαφή υλικού (hardware interface) για να μεταφέρει την ίδια εντολή, συμβάν και πακέτα δεδομένων. Οι πιο συχνά χρησιμοποιούμενες είναι το USB (στους ηλεκτρονικούς υπολογιστές) και το UART (στα κινητά τηλέφωνα και PDAs).

Σε συσκευές Bluetooth με απλές λειτουργίες (πχ. ακουστικά), η στοίβα του ξενιστή και ο ελεγκτής μπορούν να εφαρμοστούν στον ίδιο μικροεπεξεργαστή. Στην περίπτωση αυτή το HCI είναι προαιρετικό, αν και συχνά εφαρμόζεται ως εσωτερική διεπαφή λογισμικού (software interface).

7.1.2 Στοίβα Ξενιστή (Host Stack)

7.1.2.1 Logical link control and adaptation protocol (L2CAP)

Το L2CAP χρησιμοποιείται μέσα στην στοίβα του πρωτοκόλλου Bluetooth. Περνά τα πακέτα είτε με την Host Controller Interface (HCI) ή σε hostless συστήματα, απευθείας στο Διαχειριστή ζεύξεων (Link Manager).

Οι λειτουργίες του L2CAP περιλαμβάνουν:

- Πολυπλεξία δεδομένων μεταξύ διαφορετικών πρωτοκόλλων υψηλότερου επιπέδου.
- Κατακερματισμού και επανασυναρμολόγησης των πακέτων.
- Παροχή διαχείρισης μονόδρομης διαβίβασης των δεδομένων πολυεκπομπής (multicast) σε μια ομάδα άλλων συσκευών Bluetooth.
- Διαχείριση Ποιότητα της υπηρεσίας Quality of Service (QoS) για τα πρωτόκολλα υψηλότερου επιπέδου.

Το L2CAP χρησιμοποιείται για να επικοινωνούν μέσω της ζεύξης ACL του ξενιστή. Η σύνδεση της εγκαθιστάται μετά τη σύσταση της ζεύξης ACL.

Στη βασική λειτουργία, το L2CAP παρέχει πακέτα με ωφέλιμο φορτίο διαμορφώσιμο μέχρι 64 kB, με 672 bytes ως το προεπιλεγμένο MTU, και 48 bytes ως το ελάχιστο υποστηριζόμενο υποχρεωτικό μέγεθος MTU. Στην αναμετάδοση και τους τρόπους ελέγχου της ροής, το L2CAP μπορεί να ρυθμιστεί για αξιόπιστα ή ισοχρονισμένα δεδομένα ανά κανάλι, εκτελώντας την αναμετάδοση και ελέγχους CRC.

Η αξιοπιστία και στους δύο από αυτούς τρόπους είναι προαιρετική και/ή εγγυάται επιπλέον από το χαμηλότερο στρώμα Bluetooth BDR/διεπαφή αέρα EDR, με τη ρύθμιση του αριθμού των αναμεταδόσεων και του εξωχρονισμού κένωσης (ο χρόνος μετά τον οποίο τα πακέτα θα κενωθούν). Η in-order αλληλουχία είναι εγγυημένη από το χαμηλότερο στρώμα.

Οι προδιαγραφές του EL2CAP προσθέτουν μια επιπλέον ενισχυμένη λειτουργία αναμετάδοσης (enhanced retransmission mode ERTM) των προδιαγραφών πυρήνα, που είναι μια βελτιωμένη έκδοση της αναμετάδοσης και τρόπου ελέγχου της ροής. Η ERTM απαιτείται κατά τη χρήση ενός AMP, όπως 802.11abgn.

7.1.2.2 Bluetooth network encapsulation protocol (BNEP)

Το BNEP χρησιμοποιείται για την παράδοση πακέτων δικτύου πάνω από το L2CAP. Το πρωτόκολλο αυτό χρησιμοποιείται από το προφίλ personal area network *(PAN).* Το BNEP επιτελεί λειτουργία όμοια με του Subnetwork Access Protocol (SNAP) σε ασύρματο δίκτυο LAN.

Στην στοίβα πρωτοκόλλου, το BNEP είναι δεμένο με το L2CAP.

7.1.2.3 Radio frequency communication (RFComm.)

Το πρωτόκολλο Bluetooth RFComm. είναι ένα απλό σύνολο πρωτοκόλλων μεταφοράς, πάνω από το πρωτόκολλο L2CAP, προσφέροντας εξομοίωση RS-232 σειριακών θυρών (μέχρι εξήντα ταυτόχρονες συνδέσεις με μια συσκευή Bluetooth σε μια στιγμή). Το πρωτόκολλο βασίζεται στο πρότυπο ETSI TS 07.10.

Το RFComm. μερικές φορές ονομάζεται εξομοίωση σηριακής θύρας (serial port emulation). The Bluetooth *serial port profile* is based on this protocol. Το προφίλ σειριακής θύρας (serial port profile) του Bluetooth είναι βασισμένο σε αυτό το πρωτόκολλο.

Το RFComm. παρέχει μια απλή αξιόπιστη ροή δεδομένων προς το χρήστη, παρόμοια με το TCP. Χρησιμοποιείται άμεσα από πολλά προφίλ τηλεφωνίας ως φορέας για εντολές AT, καθώς είναι επίσης ένα στρώμα μεταφορών για OBEX μέσω Bluetooth.

Πολλές εφαρμογές Bluetooth χρησιμοποιούν RFComm. λόγω της ευρείας υποστήριξης των διαθέσιμων στο κοινό API, για τα περισσότερα λειτουργικά συστήματα. Επιπλέον, οι εφαρμογές που χρησιμοποιούν σειριακή θύρα για επικοινωνία μπορούν εύκολα να μεταφερθούν ούτως ώστε να χρησιμοποιούν το RFComm.

Στην στοίβα πρωτοκόλλου, το RFComm. είναι δεμένο με το L2CAP.

7.1.2.4 Service discovery protocol (SDP)

Χρησιμοποιείται για να επιτρέψει σε συσκευές να ανακαλύψουν τι υπηρεσίες προσφέρει η κάθε μία και τι παραμέτρους να χρησιμοποιήσουν για να συνδεθούν με αυτές. Για παράδειγμα, όταν συνδέετε ένα κινητό τηλέφωνο με ένα ακουστικό Bluetooth, το SDP θα χρησιμοποιηθεί για να καθορίσει ποια προφίλ Bluetooth υποστηρίζονται από το ακουστικό (προφίλ ακουστικού, προφίλ hands-free, προηγμένο προφίλ διανομής ήχου, κλπ.), καθώς και τις ρυθμίσεις πολυπλεξίας του πρωτοκόλλου που απαιτούνται για τη σύνδεση με καθένα από αυτά. Κάθε υπηρεσία αναγνωρίζεται από ένα διεθνώς μοναδικό αναγνωριστικό (Universally Unique Identifier UUID), με επίσημες υπηρεσίες (προφίλ Bluetooth) να αποδίδονται σε μια μικρή φόρμα του UUID (16 bits και όχι η πλήρης 128).

Στην στοίβα πρωτοκόλλου, το SDP είναι δεμένο με το L2CAP.

7.1.2.5 Telephony control protocol (TCP)

Αναφέρεται επίσης και ως telephony control protocol specification binary (TCS δυαδικό).

Χρησιμοποιείται για την εγκατάσταση και τον έλεγχο της ομιλίας και των δεδομένων κλήσης μεταξύ συσκευών Bluetooth. Το πρωτόκολλο βασίζεται στο πρότυπο της ITU-T Q.931, με τις διατάξεις του παραρτήματος Δ να εφαρμόζονται, κάνοντας μόνο τις ελάχιστες αλλαγές που είναι αναγκαίες για Bluetooth.

Το TCP χρησιμοποιείται από τα προφίλ ενδοεπικοινωνίας (ICP) και ασύρματης τηλεφωνία (CTP).

Στην στοίβα πρωτοκόλλου, το πρωτόκολλο TCP είναι δεμένο με το L2CAP.

7.1.2.6 Audio/visual control transport protocol (AVCTP)

Χρησιμοποιείται από το προφίλ εξ αποστάσεως ελέγχου (remote control profile) για τη μεταφορά εντολών AV/C πάνω από ένα κανάλι L2CAP. Τα κουμπιά ελέγχου της μουσικής σε ένα στερεοφωνικό ακουστικό χρησιμοποιούν αυτό το πρωτόκολλο για τον έλεγχο της αναπαραγωγής μουσικής.

Στην στοίβα πρωτοκόλλου, AVCTP είναι δεμένο με το L2CAP.

7.1.2.7 Audio/visual data transport protocol (AVDTP)

Χρησιμοποιείται από το προφίλ προηγμένης διανομής ήχου για να ροοθηκεύση (stream) μουσική προς τα στερεοφωνικά ακουστικά πάνω από ένα κανάλι L2CAP. Προορίζεται να χρησιμοποιηθεί από το προφίλ διανομής βίντεο.

Στην στοίβα πρωτοκόλλου, AVDTP είναι δεμένο με το L2CAP.

7.1.2.8 Object exchange (OBEX)

Το OBEX περιγράφεται αναλυτικά στην επόμενη παράγραφο.

7.2 Ζευγοποίηση (Pairing)

7.2.1 Κίνητρα

Πολλές από τις υπηρεσίες που παρέχονται μέσω Bluetooth μπορούν να εκθέσουν τα ιδιωτικά στοιχεία ή να επιτρέψουν τη σύνδεση ομάδας ατόμων για τον έλεγχο της συσκευής Bluetooth. Για λόγους ασφαλείας, είναι επομένως απαραίτητο να ελεγχθεί ποιες συσκευές μπορούν να συνδεθούν με μια συγκεκριμένη συσκευή Bluetooth. Ταυτόχρονα, είναι χρήσιμο για τις συσκευές Bluetooth να δημιουργούν αυτόματα μια σύνδεση χωρίς την παρέμβαση του χρήστη, μόλις είναι σε απόσταση.

Για την επίλυση αυτής της σύγκρουσης, το Bluetooth χρησιμοποιεί μια διαδικασία που ονομάζεται ζευγοποίηση, η οποία συνήθως εκκινείται χειροκίνητα για την συσκευή με το να καταστεί ορατή η Bluetooth σύνδεση σε άλλες συσκευές Δύο συσκευές θα πρέπει να ζευγοποιηθούν για να επικοινωνούν μεταξύ τους. Η διαδικασία ζευγοποίησης ενεργοποιείται συνήθως αυτόματα την πρώτη φορά που μια συσκευή λαμβάνει μια αίτηση σύνδεσης από μια συσκευή με την οποία δεν έχει ακόμη συνδεθεί. Άπαξ η ζευγοποίηση έχει καθιερωθεί, αποθηκεύεται από τις συσκευές, ούτως ώστε να μπορούν να συνδεθούν μεταγενέστερα χωρίς την παρέμβαση του χρήστη. Η ζευγοποίηση μπορεί αργότερα να αφαιρεθεί από τον χρήστη.

7.2.2 Εφαρμογή

Κατά τη διάρκεια της διαδικασίας ζευγοποίησης, οι δύο συσκευές που συμμετέχουν καθορίζουν μια σχέση με τη δημιουργία ενός κοινού μυστικού (shared key)που είναι γνωστό ως κλειδί σύνδεσης (link key).

Αν τα κλειδιά σύνδεσης αποθηκευτούν και από τις δύο συσκευές τότε λέγεται ότι οι συσκευές έχουν δέσμιση (bonding). Μια συσκευή που θέλει να επικοινωνήσει μόνο με μια δεσμική συσκευή μπορεί να επικυρώσει κρυπτογραφικά την ταυτότητα της άλλης συσκευής, και έτσι να είναι βέβαια ότι είναι η ίδια συσκευή με την οποία είχε συνδεθεί στο παρελθόν. Άπαξ ένα κλειδί σύνδεσης δημιουργηθεί, μια επικυρωμένη ACL σύνδεση μεταξύ των συσκευών μπορεί να είναι κρυπτογραφημένη, έτσι ώστε τα δεδομένα που ανταλλάσσουν μέσω του αέρα να προστατεύονται από υποκλοπές.

Τα κλειδιά σύνδεσης μπορούν να διαγραφούν ανά πάσα στιγμή από οποιαδήποτε συσκευή. Αν γίνει είτε από μια είτε από την άλλη συσκευή αυτό θα αφαιρέσει σιωπηρά

το δεσμικό μεταξύ των συσκευών. Έτσι είναι δυνατό για μια από τις συσκευές να έχει κεντρικά αποθηκευμένη το κλειδί σύνδεσης, αλλά δεν είναι δυνατόν να γνωρίζει ότι δεν έχει πλέον δεσμικό με τη συσκευή που συνδέεται με το συγκεκριμένο κλειδί σύνδεσης.

Οι υπηρεσίες Bluetooth, γενικά, απαιτούν είτε κρυπτογράφηση είτε ταυτοποίηση και συνεπώς, απαιτούν τη ζευγοποίηση πριν να επιτρέψουν σε μια απομακρυσμένη συσκευή να χρησιμοποιήσει την δεδομένη υπηρεσία. Ορισμένες υπηρεσίες, όπως η προφίλ ώθησης αντικειμένου (Object Push Profile), επιλέγουν να μην απαιτούν ρητά επαλήθευση (authentication) ή κρυπτογράφηση έτσι ώστε η ζευγοποίηση να μην επηρεάζει την εμπειρία του χρήστη που συνδέεται με την υπηρεσία.

7.2.3 Μηχανισμοί Ζευγοποίησης

Οι μηχανισμοί ζευγοποίησης έχουν αλλάξει σημαντικά με την εισαγωγή του Simple Secure Pairing σε Bluetooth v2.1. Στα επόμενα συνοψίζονται οι μηχανισμοί ζευγοποίησης:

7.2.3.1 Παραδοσιακή ζευγοποίηση (legacy pairing)

Αυτή είναι η μόνη διαθέσιμη μέθοδος πριν το Bluetooth v2.1. Σε κάθε συσκευή πρέπει να εισάγεται ο κωδικός PIN. Η ζευγοποίηση είναι επιτυχής μόνο εάν και οι δύο συσκευές πληκτρολογήσουν τον ίδιο κωδικό PIN. Κάθε 16-byte UTF-8 στοιχειοσειρά μπορεί να χρησιμοποιηθεί ως ένα κωδικός PIN, ωστόσο μπορεί να μην είναι όλες οι συσκευές σε θέση να εισάγουν όλους τους πιθανούς κωδικούς PIN.

Περιορισμένες συσκευές εισόδου (limited input devices): Το προφανές παράδειγμα αυτής της κατηγορίας είναι μια λυσιχερής συσκευή Bluetooth, που κατά κανόνα έχει λίγες εισόδους. Οι συσκευές αυτές συνήθως έχουν ένα σταθερό κωδικό PIN, για παράδειγμα, "0000" ή "1234", που είναι γραμμένο μέσα στη συσκευή.

Αριθμητικές συσκευές εισόδου (numeric input devices): Τα κινητά τηλέφωνα είναι τα κλασικά παραδείγματα αυτών των συσκευών. Παρέχουν τη δυνατότητα στο χρήστη να εισάγει μια αριθμητική τιμή έως και 16 ψηφία στο μήκος.

Αλφαριθμητικές συσκευές εισόδου (alpha-numeric input devices): Παραδείγματα αυτών των συσκευών είναι οι ηλεκτρονικοί υπολογιστές και τα έξυπνα τηλέφωνα. Επιτρέπουν στο χρήστη να εισάγει πλήρως UTF-8 κείμενο ως ένα κωδικό PIN. Αν η ζευγοποίηση γίνεται με μια λιγότερο ικανή συσκευή τότε ο χρήστης πρέπει να γνωρίζει τους περιορισμούς στην εισαγωγή χαρακτήρων για την άλλη συσκευή. Δεν υπάρχει κανένας μηχανισμός για μια ικανή συσκευή για να καθορίσει πώς θα πρέπει να περιορίσει την διαθέσιμη είσοδο που μπορεί να χρησιμοποιήσει ένας χρήστης.

7.2.3.2 Ασφαλής και σίγουρη ζευγοποίηση (Simple and Secure pairing SSP)

Αυτό είναι που απαιτείται από Bluetooth v2.1. Μια συσκευή Bluetooth v2.1 μπορεί να χρησιμοποιήσει μόνο παραδοσιακή ζευγοποίηση όταν πρόκειται να συνδεθεί με συσκευή v2.0 ή νωρίτερα. Η ασφαλής και σίγουρη ζευγοποίηση χρησιμοποιεί μια μορφή κρυπτογραφίας δημόσιου κλειδιού, και έχει τους ακόλουθους τρόπους λειτουργίας:

Απλά δουλεύει (just works): Όπως συνάγεται από την ονομασία, αυτή η μέθοδος απλά δουλεύει. Δεν απαιτείται αλληλεπίδραση του χρήστη. Ωστόσο, μια συσκευή μπορεί να ζητήσει από το χρήστη να επιβεβαιώσει τη διαδικασία ζευγοποίησης. Αυτή η μέθοδος χρησιμοποιείται συνήθως από τα ακουστικά με πολύ περιορισμένες δυνατότητες εισόδου/εξόδου, και είναι πιο ασφαλές από το σταθερό μηχανισμό PIN που χρησιμοποιείται συνήθως για την παραδοσιακή ζευγοποίηση με αυτό το σύνολο των περιορισμένων συσκευών. Η μέθοδος αυτή δεν παρέχει προστασία από επιθέσεις man in the middle (MITM).

Σύγκριση Αριθμητικού (numeric comparison): Αν και οι δύο συσκευές έχουν μια οθόνη και μια τουλάχιστον μπορεί να δεχθεί ένα δυαδικό NAI/OXI σαν είσοδο από τον χρήστη, τότε μπορούν να χρησιμοποιήσουν την σύγκριση αριθμητική. Η μέθοδος αυτή εμφανίζει ένα 6-ψήφιο αριθμητικό κωδικό για κάθε συσκευή. Ο χρήστης θα πρέπει να συγκρίνει τους αριθμούς για να εξασφαλιστεί ότι είναι πανομοιότυποι. Εάν η σύγκριση είναι επιτυχής, ο χρήστης θα πρέπει να επιβεβαιώσει την ζευγοποίηση με τη συσκευή που μπορεί να αποδεχθεί μια είσοδο. Αυτή η μέθοδος παρέχει MITM προστασία, αν υποθέσουμε ότι ο χρήστης επιβεβαιώνει και στις δύο συσκευές και όντως εκτελεί τη σύγκριση σωστά.

Είσοδος κωδικού (passkey entry): Αυτή η μέθοδος μπορεί να χρησιμοποιηθεί από μια συσκευή με μια οθόνη και μια συσκευή με αριθμητικό πληκτρολόγιο εισόδου (όπως ένα πληκτρολόγιο), ή δύο συσκευές με αριθμητικό πληκτρολόγιο εισόδου. Στην πρώτη περίπτωση, η οθόνη χρησιμοποιείται για να δείξει ένα 6-ψήφιο αριθμητικό κωδικό για τον χρήστη, ο οποίος εισάγει στη συνέχεια τον κωδικό στο πληκτρολόγιο. Στη δεύτερη περίπτωση, ο χρήστης της κάθε συσκευής εισάγει τον ίδιο 6-ψήφιο αριθμό. Και οι δύο περιπτώσεις παρέχουν MITM προστασία.

Εκτός ζώνης (out of band OOB): Η μέθοδος αυτή χρησιμοποιεί ένα εξωτερικό μέσο επικοινωνίας (όπως το NFC), για την ανταλλαγή πληροφοριών που χρησιμοποιούνται στη διαδικασία ζευγοποίησης. Η ζευγοποίηση ολοκληρώνεται με τη χρήση του Bluetooth **ραδιόφωνο**, αλλά απαιτεί πληροφορίες από τον μηχανισμό

OOB. Αυτό προβλέπει επίπεδο προστασίας από επιθέσεις MITM στο ίδιο βαθμό με αυτό που παρέχεται από τον μηχανισμό OOB.

Η SSP θεωρείται απλή για τους ακόλουθους λόγους:

- Στις περισσότερες περιπτώσεις, δεν χρειάζεται ο χρήστης για να δημιουργήσει ένα κωδικό.
- Για περιπτώσεις που δεν απαιτούν MITM προστασία, η παρέμβαση του χρήστη, έχει εξαλειφθεί.
- Για την αριθμητική σύγκριση, MITM προστασία μπορεί να επιτευχθεί με μια απλή σύγκριση της ισότητας από το χρήστη.
- Με τη χρήση OOB με NFC επιτρέπεται η ζευγοποίηση όταν συσκευές απλά πλησιάσουν κοντά, χωρίς να χρειάζεται η χρονοβόρα διαδικασία ανακάλυψης.

8. OBject EXchange (OBEX)

Το OBEX ή IrOBEX είναι ένα πρωτόκολλο επικοινωνίας που επιτρέπει την ανταλλαγή δυαδικών αντικειμένων μεταξύ συσκευών.

Διατηρείται από την Infrared Data Association, αλλά έχει επίσης υιοθετηθεί από το Bluetooth Special Interest Group και το SyncML πτέρυγα της Open Mobile Alliance (OMA). Το OBEX χρησιμοποιείται ευρέως για την ανταλλαγή προσωπικών στοιχείων, δεδομένων, ακόμα και εφαρμογών.

Αν και το OBEX αρχικά είχε σχεδιαστεί για υπέρυθρες ακτινοβολίες, έχει πλέον υιοθετηθεί από το Bluetooth και χρησιμοποιείται επίσης σε RS232, USB και WAP.

Σε Bluetooth, το OBEX χρησιμοποιείται για πολλά προφίλ που απαιτούν απλή ανταλλαγή δεδομένων (π.χ., object push, μεταφορά αρχείων, βασικές λειτουργίες επεξεργασίας εικόνας, βασικές λειτουργίες εκτύπωσης, πρόσβαση στον τηλεφωνικό κατάλογο, κλπ.).

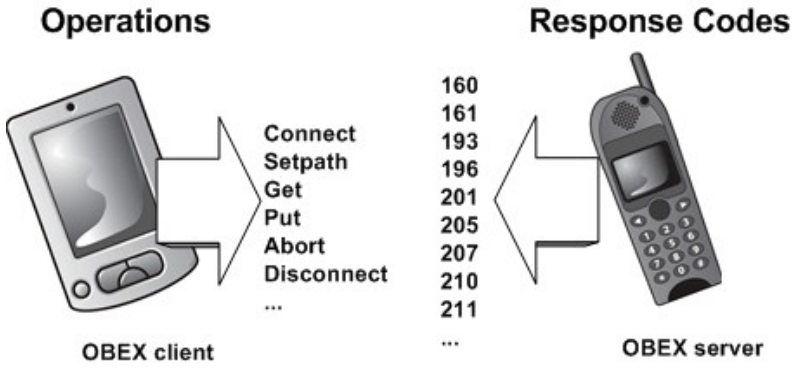
In the protocol stack, OBEX is bound to RFComm. Στην στοίβα πρωτοκόλλου, OBEX είναι υποχρεωμένη να RFComm.

8.1 HTTP

Το OBEX είναι παρόμοιο στο σχεδιασμό και τη λειτουργία με το HTTP για την παροχή στον χρήστη με ένα αξιόπιστο μέσο για τη σύνδεση σε ένα διακομιστή όπου μετά μπορεί να ζητήσει ή να παρέχει αντικείμενα. Αλλά το OBEX διαφέρει σε πολλά σημαντικά σημεία:

- Το OBEX συνήθως εφαρμόζεται σε μια IrLAP/IrLMP/Tiny TP στοίβα σε μια IrDA συσκευή.

  Στο Bluetooth, το OBEX εφαρμόζεται σε μια baseband/ACL/L2CAP/RFCOMM στοίβα. Άλλα τέτοια “δεσίματα” του OBEX είναι δυνατά.
- Το HTTP χρησιμοποιεί αναγνώσιμο από τον άνθρωπο κείμενο, αλλά το OBEX χρησιμοποιεί δυαδικές τύπου-μήκος-τιμής τρίπλέτες που ονομάζονται “επικεφαλίδες” (Headers) για την ανταλλαγή πληροφοριών σχετικά με ένα αίτημα ή ένα αντικείμενο. Η ανάλυση τους είναι πολύ πιο εύκολη σε συσκευές περιορισμένων πόρων, όπως είναι τα κινητά τηλέφωνα.
- Οι συναλλαγές HTTP δεν έχουν εγγενώς κατάσταση. Γενικά έναν χρήστης ανοίγει μια HTTP σύνδεση, κάνει μία αίτηση, λαμβάνει την απάντησή της, και είτε κλείνει τη σύνδεση ή κάνει άλλα άσχετα αιτήματα. Στο OBEX, μία και μόνο σύνδεση μεταφοράς μπορεί να φέρει πολλές σχετικές δραστηριότητες. Στην πραγματικότητα, οι πρόσφατες προσθήκες με τις προδιαγραφές OBEX επιτρέπουν σε μια σύνδεση που έχει κλείσει ανώμαλα να επανακτηθεί με όλες τις πληροφορίες τις παλαιάσς της κατάστασης άθικτες.



Σχήμα 2: Λειτουργίες OBEX

8.2 Προφίλ (Profiles)

Το OBEX είναι η βάση για πολλά υψηλότερου στρώματος ”προφίλ”:

Συγκεκριμένα για Bluetooth υπάρχουν τα προφίλ:

- Περιληπτικό προφίλ ανταλλαγής αντικειμένων – Generic Object Exchange Profile

- Προφίλ ώθησης αντικειμένων – Object Push Profile (για μεταδόσεις από κινητό τηλέφωνο σε κινητό τηλέφωνο)
- Προφίλ μεταφοράς αρχείων – File Transfer Profile (για μεταδόσεις μεταξύ κινητού τηλεφώνου και ηλεκτρονικού υπολογιστή)
- Προφίλ συγχρονισμού – Synchronization Profile
- Προφίλ βασικής απεικόνισης – Basic Imaging Profile
- Προφίλ βασικής εκτύπωσης – Basic Printing Profile

8.3 Υποστηριζόμενες συσκευές

- Όλες οι Palms από το Palm III.
- Τα πλείστα Sharp, Motorola, Samsung, Sony Ericsson, HTC and Nokia κινητά τηλέφωνα με υπέρυθρη ή Bluetooth θύρα.
- Πολλές άλλες συσκευές PDA από το 2003.
- Πολλά άλλα κινητά τηλέφωνα με υπέρυθρη ή Bluetooth θύρα.

Το OBEX υποστηρίζεται και από κινητά τηλέφωνα με Android αλλά χρειάζεται μη τυποποιημένη διαδικασία ενεργοποίησης.

9. Λειτουργικά Συστήματα Κινητών Τηλεφώνων

9.1 Symbian OS

Το Symbian OS είναι ένα λειτουργικό σύστημα (OS) που προορίζεται για φορητές συσκευές και έξυπνα τηλέφωνα (smartphones) με τις σχετικές βιβλιοθήκες, διεπαφές χρήστη, πλαίσια και υλοποιήσεις αναφοράς για τα κοινά εργαλεία, που αρχικά αναπτύχθηκε από την Symbian Ltd. Ήταν απόγονος του EPOC της Psion και λειτουργούσε αποκλειστικά για την ARM επεξεργαστές.

Το 2008, το πρώην Symbian Software Limited εξαγοράστηκε από την Nokia και ένας νέος ανεξάρτητος, μη κερδοσκοπικός οργανισμός, που ονομάζεται Symbian Foundation ιδρύθηκε. Το Symbian OS και οι σχετιζόμενες διεπαφές χρήστη S60 , UIQ και MOAP(S) διατέθηκαν από τους ιδιοκτήτες τους στο ίδρυμα με στόχο τη δημιουργία της πλατφόρμας Symbian ως royalty-free, λογισμικό ανοικτού κώδικα. Η πλατφόρμα έχει χαρακτηριστεί ως ο διάδοχος του Symbian OS, μετά την επίσημη έναρξη του Symbian Foundation τον Απρίλιο του 2009. Η πλατφόρμα Symbian ήταν επίσημα διαθέσιμη ως ανοικτός πηγαίος κώδικας, τον Φεβρουάριο του 2010. Συσκευές που

βασίζονται στο λειτουργικό σύστημα Symbian υπολογίζονται στο 46,9% των πωλήσεων έξυπνων τηλεφώνων, γεγονός που το καθιστά το δημοφιλέστερο λειτουργικό σύστημα για κινητά τηλέφωνα στον κόσμο.

Το λειτουργικό σύστημα Symbian υποστηρίζει εγγενώς την Java ME.

9.1.1 Δομή

Το μοντέλο συστήματος του Symbian περιέχει τα ακόλουθα στρώματα, από πάνω προς τα κάτω:

- Στρώμα πλαισίου διεπαφών χρήστη (UI Framework Layer)
- Στρώμα υπηρεσιών εφαρμογών (Application Services Layer)
    - Java ME
- Στρώμα υπηρεσιών λειτουργικού συστήματος (OS Services Layer)
    - Υπηρεσίες γενικής χρήσης του λειτουργικού συστήματος
    - Υπηρεσίες επικοινωνιών
    - Υπηρεσίες πολυμέσων και γραφικών
    - Υπηρεσίες διασύνδεσης
- Στρώμα υπηρεσιών βάσης (Base Services Layer)
- Στρώμα υπηρεσιών πυρήνα και υλικού (Kernel Services & Hardware Interface Layer)

9.2 Android (Linux-based)

Το Android είναι ένα λειτουργικό σύστημα για κινητά τηλέφωνα που χρησιμοποιεί μια τροποποιημένη έκδοση του πυρήνα του Linux. Αρχικά αναπτύχθηκε από την Android Inc, μια εταιρεία που αργότερα αγοράστηκε από την Google, και πρόσφατα από την Open Handset Alliance. Επιτρέπει στους προγραμματιστές να γράψουν κώδικα στη γλώσσα προγραμματισμού Java, ελέγχοντας τη συσκευή μέσω ανεπτυγμένες βιβλιοθήκες Java του Google.

Τα αποκαλυπτήρια της διανομής Android ανακοινώθηκαν στις 5 Νοεμβρίου 2007, με την ίδρυση του Το Open Handset Alliance , μια κοινοπραξία 47 εταιρειών υλικού, λογισμικού και τηλεπικοινωνιών αφιερωμένη στην προώθηση των ανοικτών προτύπων για τις κινητές συσκευές. Η Google δημοσιοποίησε το μεγαλύτερο μέρος του κώδικα του Android στο πλαίσιο της άδειας Apache, δωρεάν λογισμικού και ανοικτού πηγαίου κώδικα.

9.2.1 Χαρακτηριστικά

Στα χαρακτηριστικά του Android περιλαμβάνονται καταστρώσεις χειροσυσκευών, αποθήκευση, συνδεσιμότητα, μηνύματα, φυλλομετρητής ιστού, υποστήριξη Java, υποστήριξη πολυμέσων, πρόσθετη υποστήριξη υλικού, περιβάλλον ανάπτυξης και οθόνη πολλαπλής αφής.

Συγκεκριμένα για την Java. Το λογισμικό γραμμένο σε Java να μπορεί να μεταγλωττιστεί για να εκτελεστεί στην εικονική μηχανή Dalvik, η οποία είναι μια εξειδικευμένη εφαρμογή εικονικής μηχανής (virtual machine) σχεδιασμένη για χρήση σε κινητά τηλέφωνα, αν και δεν είναι τεχνικά μια πρότυπη εικονική μηχανή Java. Το Android δεν υποστηρίζει εγγενώς J2ME, αλλά με τη χρήση της υλοποίησης j2me runner η εφαρμογές Java ME είναι δυνατόν να εκτελεστούν.

9.3 Blackberry OS Java ME MIDP 1.0

Το BlackBerry OS είναι μια ιδιόκτητη πλατφόρμα λογισμικού που υλοποιείται από την Research In Motion για την BlackBerry. Το BlackBerry OS παρέχει πολυέργεια (multi-tasking), και χρησιμοποιεί ευρέως τις εξειδικευμένες συσκευές εισόδου της συσκευής, και ιδίως την οθόνη αφής.

Το λειτουργικό σύστημα παρέχει υποστήριξη για MIDP 1.0 και WAP 1.2. Η τρέχουσα έκδοση του λειτουργικού OS 4 παρέχει ένα υποσύνολο των MIDP 2.0.

Τρίτοι προγραμματιστές μπορούν να γράψουν υλοποιήσεις χρησιμοποιώντας BlackBerry APIs, αλλά κάθε εφαρμογή πρέπει να έχει υπογραφεί ψηφιακά, ώστε να μπορεί να συνδεθεί σε ένα λογαριασμό του προγραμματιστή στο RIM. Η διαδικασία αυτή εγγυάται την υπογραφή του συντάκτη της υλοποίησης, αλλά δεν εγγυάται την ποιότητα ή την ασφάλεια του κωδικού.

9.4 iPhone OS

Το iPhone OS είναι ένα λειτουργικό σύστημα για κινητά τηλέφωνα που έχει αναπτυχθεί και διατίθεται στο εμπόριο από την Apple Inc. Είναι το προεπιλεγμένο λειτουργικό σύστημα του iPhone, iPod Touch, και iPad .

Προέρχεται από το Mac OS X και αποτελεί ένα Unix-like λειτουργικό σύστημα.

9.4.1 Java

Η Apple δεν έχει ανακοινώσει κανένα σχέδιο ώστε να μπορέσει η Java να τρέξει στο iPhone. Η Sun Microsystems ανακοίνωσε σχέδια για να κυκλοφορήσει μια εικονική

μηχανή Java (JVM) για το iPhone OS, με βάση την πλατφόρμα Java Platform, Micro Edition έκδοση της Java. Αυτό θα επέτρεπε εφαρμογές Java να τρέξουν στο iPhone.

9.5 Windows Mobile

Το Windows Mobile είναι ένα λειτουργικό σύστημα για κινητά τηλέφωνα που αναπτύχθηκε από τη Microsoft, και έχουν σχεδιαστεί για χρήση σε έξυπνα τηλέφωνα και φορητές συσκευές .

Είναι βασισμένο στο Windows CE 5.2 πυρήνα, και διαθέτει μια σουίτα βασικών εφαρμογών που έχουν αναπτυχθεί χρησιμοποιώντας το Microsoft Windows API. Έχει σχεδιαστεί να είναι παρόμοιο με την επιφάνεια εργασίας των Windows, τόσο από πλευράς χαρακτηριστικών όσο και αισθητικά.

Το Windows Mobile δεν υποστηρίζει Java ME.

10. Τεχνολογίες Ανάπτυξης

10.1 Java

10.1.1 Java Web Start

Η Java Web Start (επίσης γνωστή ως JavaWS ή javaws) είναι ένα πλαίσιο που αναπτύχθηκε από την Sun Microsystems που επιτρέπει στους χρήστες να εκκινήσουν μια εφαρμογή για την πλατφόρμα Java απευθείας από το διαδίκτυο χρησιμοποιώντας ένα πρόγραμμα φυλλομετρητή ιστού.

10.1.2 Java Network Launching Protocol (JNLP)

Το JNLP είναι γνωστό εναλλακτικά με τον όρο "Web Start". Το πρωτόκολλο JNLP, ορίζεται μια δομή XML, ορίζει τον τρόπο εκκίνησης μιας Java Web Start εφαρμογής. Το JNLP αποτελείται από ένα σύνολο κανόνων που καθορίζουν πώς ακριβώς να εφαρμόσουν τον μηχανισμό εκκίνησης. Τα αρχεία JNLP περιλαμβάνουν πληροφορίες όπως η θέση του jar αρχείου του πακέτου και το όνομα της main class της εφαρμογής, πέρα από οποιεσδήποτε άλλες παράμετρους της εφαρμογής. Ένας σωστά ρυθμισμένος φυλλομετρητής περνάει τα JNLP αρχεία σε ένα Java Runtime Environment (JRE), το οποίο με τη σειρά του κατεβάζει την εφαρμογή στον υπολογιστή του χρήστη και ξεκινά την εκτέλεση της. Η ανάπτυξη του JNLP πραγματοποιήθηκε στο πλαίσιο του Java Community Process ως JSR 56. Το JNLP είναι δωρεάν. Οι προγραμματιστές δεν χρειάζεται να καταβάλλουν τέλη αδείας, προκειμένου να το χρησιμοποιήσουν σε υλοποιήσεις.

Σημαντικά χαρακτηριστικά γνωρίσματα του Web Start περιλαμβάνουν τη δυνατότητα αυτόματης λήψης και εγκατάστασης ενός JRE στην περίπτωση που ο χρήστης δεν έχει εγκαταστήσει Java, καθώς και τη δυνατότητα στους προγραμματιστές να καθορίσουν την έκδοση του JRE που απαιτείται για την ορθή εκτέλεση μιας εφαρμογής. Ο χρήστης δεν χρειάζεται να παραμείνει συνδεδεμένος με το διαδίκτυο για να εκτελέσει την εφαρμογή που έχει κατεβάσει, επειδή αυτή εκτελείται τοπικά και διατηρείται σε προσωρινή μνήμη. Όταν ο χρήστης διαθέτει σύνδεση στο διαδίκτυο μπορεί να κατεβάσει τις εκδόσεις αυτές αυτόματα.

Οποιοσδήποτε υπολογιστής μπορεί να χρησιμοποιήσει JNLP απλά με την εγκατάσταση ενός JNLP client (συνηθέστερα Java Web Start). Η εγκατάσταση μπορεί να συμβεί αυτόματα έτσι ώστε ο τελικός χρήστης βλέπει τη λήψη και εγκατάσταση της εφαρμογής Java κατά την πρώτη εκτέλεση.

10.2 Java Micro Edition (Java ME)

10.2.1 Εισαγωγή

Η *Java 2 Micro Edition* ή *J2ME* είναι μια έκδοση της πλατφόρμας *Java* σχεδιασμένη για κινητές συσκευές και ενσωματωμένα συστήματα. Οι στοχευμένες συσκευές ποικίλλουν από βιομηχανικούς ελέγχους, κινητά τηλέφωνα μέχρι και μετατροπείς-αποκωδικοποιητές set-top-box (STP). Η πλατφόρμα J2ME έχει σχεδιαστεί από την *Sun Microsystems*. Αρχικά είχε αναπτυχθεί από το *Java Community Process* ή *JCP* σαν *Java Specification Request JSR-68*, αλλά αργότερα μετεξελίχθηκε σε διαφορετικά *JSRs*. Η *Sun* παρέχει δείγματα αναφοράς για των προδιαγραφών, η ίδια όμως δεν παρέχει υλοποιήσεις σε δυαδικό κώδικα, αλλά εξαρτάται από τρίτους στην παροχή αυτών σε κινητές συσκευές. Από τις 22 Δεκεμβρίου 2006 ο πηγαίος κώδικας της J2ME αδειοδοτούνται με την συμφωνία παραχώρησης άδειας *GNU Public License (GPL)*. Από το 2008 η J2ME περιορίζεται από τα χαρακτηριστικά του *Java Runtime Environment JRE 1.3*. Οι συσκευές J2ME υλοποιούν προφίλ (*profiles*) με πιο κοινό το *Mobile Information Device Profile (MIDP 2.0 JSR-118)*. Τα προφίλ είναι υποσύνολα ρυθμίσεων (configurations), εκ των οποίων τα επόμενα δύο:

- Connected Device Configuration (CDC) που χρησιμοποιείται κυρίως σε συκευές Προσωπικών Ψηφιακών Συσκευών Personal Digital Assistant (PDA) και STBs
- Connected Limited Device Configuration (CLDC 1.0 JSR-30) που στοχεύει κυρίως σε συσκευές κινητής τηλεφωνίας και συσκευές τηλεειδοποίησης pagers.

10.2.2 Στρώματα (Layers)

Η πλατφόρμα J2ME αποτελείται από τα εξής τρία στρώματα:

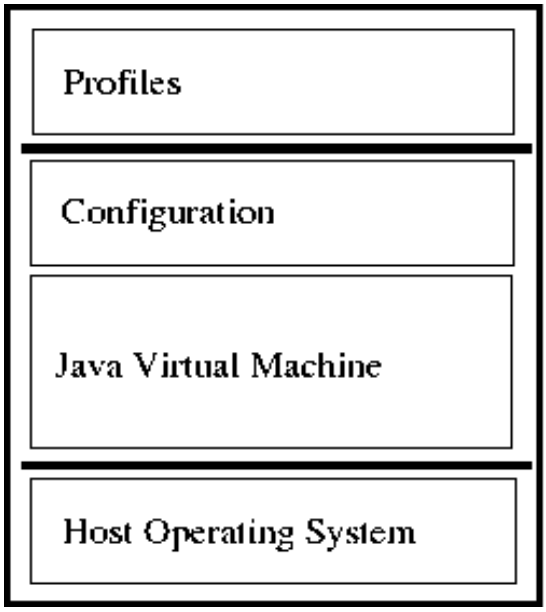
- *Java Virtual Machine JVM* – Υλοποίηση της εικονικής μηχανής της Java εξατομικευμένη για τις λειτουργικά συστήματα που τρέχουν σε συγκεκριμένες συσκευές και υποστηρίζει συγκεκριμένα configurations.
- *Configuration* – Αποτελεί το ελάχιστο σύνολο χαρακτηριστικών της εικονικής μηχανής της Java και των βιβλιοθηκών Java διαθέσιμα σε μια συγκεκριμένη κατηγορία συσκευών, που εκπροσωπούν ένα “οριζόντιο” κομμάτι της αγοράς. Είναι το ελάχιστο σύνολο που οι σχεδιαστές εφαρμογών μπορούν να θεωρούν ότι βρίσκεται σε όλες τις συσκευές της κατηγορίας.
- *Profile* – Ορίζει το ελάχιστο σύνολο διασυνδέσεων προγράμματος εφαρμογής *Application Programming Interfaces (APIs)* διαθέσιμων σε μια “οικογένεια” συσκευών, που εκπροσωπούν ένα “κάθετο” κομμάτι της αγοράς. Τα *profiles* υλοποιούν ένα συγκεκριμένο configuration. Οι εφαρμογές γράφονται βάσει ενός συγκεκριμένου profile και άρα μπορούν να τρέξουν σε οποιαδήποτε συσκευή που υποστηρίζει το profile. Κάθε συσκευή μπορεί να υποστηρίξει μόνο ένα profile.

10.2.3 K Virtual Machine

Η K (Kilo) Virtual Machine είναι μια συμπαγής, φορητή εικονική μηχανή Java σχεδιασμένη από την αρχή για συσκευές με περιορισμένη μνήμη και πόρους, που διαθέτουν σύνδεση στο δίκτυο. Χαρακτηριστικά της KVM είναι τα ακόλουθα:

- Μικρή, με στατικό αποτύπωμα του πυρήνα από 40 έως 80 kilobytes.
- Καθαρή, με ικανοποιητικά συνοδευτικά σχόλια, άκρως φορητή.
- Δομοστοιχειωτή (modular), εξατομικευμένη (customizable).
- Όσον το δυνατόν πιο πλήρης και γρήγορη χωρίς να θυσιάζονται άλλοι σχεδιαστικοί στόχοι.

Η ελάχιστη απαιτούμενη μνήμη που χρειάζεται για μια εφαρμογή σε KVM είναι 128kb, με συνιστώμενη μνήμη τα 256kb. Προς το παρόν το CLDC και η KVM είναι άρρηκτα συνδεδεμένες. Το CLDC τρέχει πάνω σε KVM και η KVM υποστηρίζει μόνο το CLDC. Στο μέλλον το CLDC θα μπορεί να τρέξει και σε άλλες εικονικές μηχανές J2ME, ίσως να συμβεί το ίδιο και με τη KVM.



Σχήμα 3: Στρώματα Java ME

10.2.4 Connected, Limited Device Configuration (CLDC)

Οι στόχοι του CLDC είναι οι εξής:

- Να ορίσει μια πρότυπη πλατφόρμα Java για μικρές συσκευές με περιορισμένους πόρους.
- Να επιτρέψει τη δυναμική παράδοση εφαρμογών Java και περιεχομένου σε τέτοιες συσκευές.
- Να επιτρέψει σε τρίτους τη δημιουργία εφαρμογών και περιεχομένου που θα μπορούν να εφαρμοστούν (deployed) σε αυτές τις συσκευές.
- Να μπορεί να τρέξει σε ευρεία γκάμα συσκευών όπως κινητά τηλέφωνα, συσκευές τηλεειδοποίησης, προσωπικούς ψηφιακούς βοηθούς και οικιακές συσκευές.
- Να κάνει τις ελάχιστες παραδοχές για το λογισμικό συστήματος διαθέσιμο σε CLDC συσκευές.
- Να ορίζει ένα ελάχιστο κοινό παρονομαστή τεχνολογίας Java σε φορητές συσκευές.
- Να εγγυηθεί φορητότητα και διαλειτουργικότητα του κώδικα σε CLDC συσκευές.

Το πεδίο εφαρμογής του CLDC απευθύνεται τις παρακάτω περιοχές:

- Γλώσσα προγραμματισμού Java και χαρακτηριστικά της εικονικής μηχανής της Java.
- Πυρήνας των βιβλιοθηκών Java (java.lang.*, java.util.*).
- Είσοδος / Έξοδος.
- Δικτύωση.
- Ασφάλεια.
- Διεθνοποίηση.

Το CLDC δεν υποστηρίζει:

- Τύπους δεδομένων float και double.
- Ανάκλαση (reflection).
- Java Native Interface (JNI).
- Finalization των στιγμιοτύπων κλάσεων.
- Αδύναμες αναφορές.
- Generics.

Όλες οι κλάσεις μιας εφαρμογής CLDC πρέπει να είναι συμπιεσμένες σε ένα αρχείο Java Archive (JAR) το οποίο περιέχει και μια στοίβα-χάρτη (stackmap) ιδιοτήτων της εφαρμογής. Το stackmap είναι αναγκαίο για την επαλήθευση των αρχείων των κλάσεων που περιέχονται.

Για την προς τα πάνω συμβατότητα, οι περισσότερες βιβλιοθήκες κλάσεων του CLDC αποτελούν υποσύνολο των βιβλιοθηκών μεγαλύτερων εκδόσεων Java (J2SE, J2EE). Μόνο οι κλάσεις που αφορούν συγκεκριμένα φορητές συσκευές καθορίζονται από το CLDC και περιέχονται στο πακέτο java.microedition.

10.2.5 Πλεονεκτήματα

Τα πλεονεκτήματα της πλατφόρμας J2ME είναι τα εξής:

- Υποστήριξη και λειτουργία σε διαφορετικές πλατφόρμες (cross-platform).
- Δυναμικό περιεχόμενο
- Ασφάλεια
- Μεγάλη κοινότητα σχεδιαστών (Developer Community)
- Δομοστοιχείωση (modularity).

Συγκεκριμένα η πλατφόρμα J2ME υποστηρίζει τεχνολογίες όπως Bluetooth (JSR-82), Web Services (JSR-172) και ασύρματη επικοινωνία (Wireless Messaging

API JSR-120 και JSR-205) που είναι πολύ σημαντικές για την υλοποίηση της διπλωματικής εργασίας.

10.2.6 Μειονεκτήματα

- Η J2ME είναι μια σειρά από JSRs τα οποία οι διάφοροι κατασκευαστές υλοποιούν με διαφορετικό τρόπο, με συνέπεια οι εφαρμογές που γράφονται σε J2ME εν γένει να μην είναι φορητές.

10.3 Συνοπτικός πίνακας

	Γλώσσα προγραμματισμού	Ολοκληρωμένο περιβάλλον ανάπτυξης	Cross-Platform Deployment
Symbian	C++	Πολλές επιλογές	Μεταγλώττιση ανά πλατφόρμα
Java ME	Java	Eclipse,LMA NetBeans Mobility Pack	Ναι, παρόλο που αρκετές υλοποιήσεις VM έχουν bugs σχετικά με τη συσκευή
Android	Java	Eclipse, Undroid (Plugin for Netbeans)	Μόνο Android
BlackBerry	Java	JDE - BlackBerry Java Development Environment	Μόνο BlackBerry
iPhone OS	Objective-C	Xcode	Μονο iPhone και iPod Touch
Windows Mobile	C, C++	Visual Studio 2008, 2005, eMbedded VC++ (free)	Windows Mobile, WindowsCE
Windows Mobile	Basic4ppc	Basic4ppc IDE	Windows Mobile, WindowsCE

Πίνακας 1: Περιβάλλοντα ανάπτυξης σε κινητά τηλέφωνα

11. Κρυπτογραφία

11.1 Κατηγορίες αλγορίθμων

11.1.1 Συμμετρικοί

Οι αλγόριθμοι Συμμετρικού-κλειδιού είναι μια κατηγορία αλγορίθμων κρυπτογράφησης που χρησιμοποιούν κοινότοπα συνδεδεμένα, συχνά ταυτόσημα, κρυπτογραφικά κλειδιά και για τιην αποκρυπτογράφηση και την κρυπτογράφηση.

Το κλειδί κρυπτογράφησης είναι κοινότοπα συσχετισμένο με το κλειδί αποκρυπτογράφησης, υπό την έννοια ότι μπορεί να ταυτίζεται ή υπάρχει μια απλή μετατροπή για να “πάει” μεταξύ των δύο κλειδιών. Τα κλειδιά, στην πράξη, αποτελούν

ένα κοινό μυστικό μεταξύ δύο ή περισσότερων ομάδων που μπορεί να χρησιμοποιηθεί για να διατηρηθεί μια ιδιωτική σύνδεση πληροφορίες.

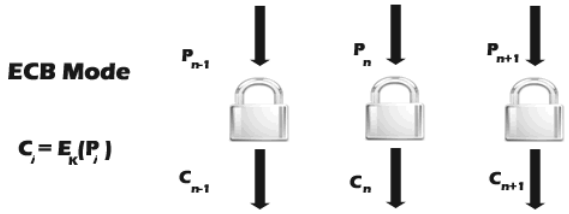
Τύποι των συμμετρικών-κλειδί αλγορίθμων

Αλγόριθμοι Συμμετρικών-κλειδιών μπορούν να διαιρεθούν σε αλγόριθμους κρυπτογραφήματος ρεύματος και μπλοκ κρυπτογραφήματος. Πιο κάτω εξηγούνται οι τρόποι λειτουργίας των κρυπτογραφημάτων:

Μερικά παραδείγματα δημοφιλών συμμετρικών αλγόριθμων περιλαμβάνουν τους Twofish, Serpent, AES (Rijndael), Blowfish, CAST5, RC4, TDES, και IDEA.

Block Cipher

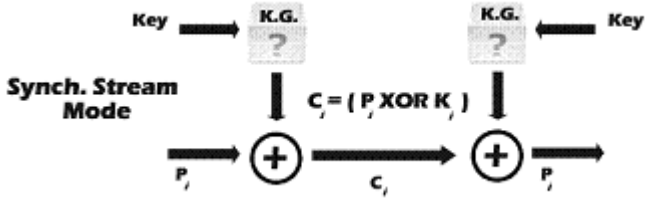
Σε αυτή τη μέθοδο τα δεδομένα κρυπτογραφούνται και αποκρυπτογραφούνται ανά μπλοκ. Στην πιο απλή λειτουργία του, διαιρείται το απλό κείμενο σε μπλοκ, τα οποία στη συνέχεια διοχετεύονται στο σύστημα κρυπτογράφησης για την παραγωγή μπλοκ του κρυπτογραφημένου κειμένου.



Σχήμα 4: Μπλοκ Κρυπτογράφηση

Stream Cipher

Ο Stream cipher λειτουργεί σε ένα ρεύμα δεδομένων το οποίο χειρίζονται λίγο λίγο. Stream cipher αποτελείται από δύο βασικά συστατικά: μια βασική γεννήτρια ρεύματος, καθώς και μια λειτουργία ανάμιξης. Η λειτουργία ανάμιξης είναι συνήθως μόνο μια λειτουργία XOR, ενώ η βασική γεννήτρια ρεύματος είναι η κύρια μονάδα σε αυτή την τεχνική κρυπτογράφησης. Για παράδειγμα, εάν το κλειδί παράγει μια σειρά από μηδενικά, στην έξοδο το αποτέλεσμα θα είναι πανομοιότυπο με το αρχικό απλό κείμενο.



Σχήμα 5: Κρυπτογράφηση ροής

11.1.2 Ασύμμετροι ή Δημοσίου κλειδιού (Public-Key)

Η κρυπτογραφία δημόσιου κλειδιού είναι μια κρυπτογραφική προσέγγιση η οποία συνεπάγεται τη χρήση των ασύμμετρων βασικών αλγορίθμων αντί ή παράλληλα με αλγορίθμους συμμετρικού-κλειδιού. Σε αντίθεση με τους αλγορίθμους συμμετρικού-κλειδιού, δεν απαιτείται μια ασφαλή πρώτη ανταλλαγή ενός ή περισσότερων μυστικών κλειδιών μεταξύ των αποστολέα και παραλήπτη. Οι αλγόριθμοι ασύμμετρου-κλειδιού χρησιμοποιούνται για τη δημιουργία ενός μαθηματικά συσχετισμένου ζεύγους κλειδιών: ένα μυστικό ιδιωτικό κλειδί και ένα δημόσιο κλειδί που δημοσιεύεται. Η χρήση αυτών των κλειδιών επιτρέπει την προστασία της αυθεντικότητας ενός μηνύματος, δημιουργώντας μια ψηφιακή υπογραφή ενός μηνύματος χρησιμοποιώντας το ιδιωτικό κλειδί, το οποίο μπορεί να ελεγχθεί χρησιμοποιώντας το δημόσιο κλειδί. Επιτρέπει, επίσης, την προστασία του απορρήτου και της ακεραιότητας ενός μηνύματος με το δημόσιο κλειδί κρυπτογράφησης, με κρυπτογράφηση του μηνύματος χρησιμοποιώντας το δημόσιο κλειδί, το οποίο μπορεί μόνο να αποκρυπτογραφηθεί με το ιδιωτικό κλειδί.

Στα πλαίσια της διπλωματικής εργασίας δεν ενδιαφερόμαστε για τους αλγόριθμους ασύμμετρου-κλειδιού.

11.2 Αλγόριθμοι

11.2.1 Data Encryption Standard (DES)

Ήταν το πρώτο πρότυπο κρυπτογράφησης που συνιστάται από τη NIST (National Institute of Standards and Technology – Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας). Είναι βασισμένο σε ένα αλγόριθμο προτεινόμενο από την IBM που ονομάζεται Lucifer. Το DES έγινε πρότυπο το 1974. Από τότε, πολλές επιθέσεις και τις μεθόδους που καταγράφονται που εκμεταλλεύονται τις αδυναμίες του DES. Ο DES κρυπτογραφεί με ένα μπλοκ κρυπτογράφημα 64-bit.

11.2.2 Triple Data Encryption Algorithm (Triple-DES)

Ως μια επέκταση του DES, το 3DES (Triple-DES) προτάθηκε ως πρότυπο κρυπτογράφησης. Σε αυτό το πρότυπο, η μέθοδος κρυπτογράφησης είναι παρόμοια με αυτήν στην αρχική του DES, αλλά εφαρμόζεται 3 φορές για να αυξηθεί το επίπεδο κρυπτογράφησης. Αλλά είναι γνωστό ότι 3DES είναι βραδύτερη από ότι άλλες μεθόδους μπλοκ κρυπτογράφησης. Ο αλγόριθμος αυτός χρησιμοποιεί μπλοκ κρυπτογράφημα 192-bit.

11.2.3 Advanced Encryption Standard (AES)

Είναι το νέο πρότυπο κρυπτογράφησης που συνιστάται από τη NIST να αντικαταστήσει DES. Ο αλγόριθμος Rijndael είχε επιλεγεί το 1997, μετά τη διεξαγωγή διαγωνισμού για την επιλογή των καλύτερων προτύπων κρυπτογράφησης. Η επίθεση ωμής βίας είναι η μόνη γνωστή αποτελεσματική επίθεση εναντίον του, κατά την οποία ο εισβολέας προσπαθεί να δοκιμάσει όλους τους συνδυασμούς τους χαρακτήρες για να ξεκλειδώσει την κρυπτογράφηση. Ο AES χρησιμοποιεί μπλοκ κρυπτογράφημα 128-bit.

Μέρος Β

12. Περίληψη Υλοποίησης

Ο σκοπός εργασίας είναι η διευκόλυνση του ιατρικού προσωπικού κατά την εξέταση του ασθενή. Ο ασθενής έχει μαζί του, στο κινητό του τηλέφωνο, τον ιατρικό του φάκελο σε ηλεκτρονική μορφή για άμεση πρόσβαση από οποιοδήποτε αρμόδιο άτομο. Η υλοποίηση, που παρουσιάζεται, έχει πραγματοποιηθεί στα πλαίσια της διπλωματικής εργασίας.

Ο ασθενής έχει τον ιατρικό του φάκελο αποθηκευμένο σε ηλεκτρονική μορφή στο κινητό του τηλέφωνο με δυνατότητα τροποποιήσης των προσωπικών του στοιχείων.

Η ασύρματη διασύνδεση του κινητού τηλεφώνου του ασθενή με τον υπολογιστή του ιατρού δίνει τη δυνατότητα τροποποίησης και συμπλήρωσης των περιεχομένων του ιατρικού φακέλου από τον ιατρό με απλό τρόπο.

13. Υπάρχουσες υλοποιήσεις

13.1 AllOne Mobile

Το AllOne Mobile είναι μια εφαρμογή κινητού τηλεφώνου που αναπτύχθηκε από το AllOne Health. Η εφαρμογή επιτρέπει τη σύνδεση μέσω διαδικτύου σε πληροφορίες υγείας του πελάτη. Παρακάτω αναφέρονται οι παρεχόμενες δυνατότητες της υπηρεσίας:

- Αποστολή οδηγιών ορθής λήψης φάρμακων του παιδιού στη νοσοκόμα του σχολείου.
- Αποθήκευση συνταγογραφήσεων.
- Πρόσβαση στο ιατρικό ιστορικό εξαρτωμένων ατόμων.
- Αποστολή ιατρικού φακέλου της οικογένειας σε ένα νέο ειδικό ιατρό πριν από την επίσκεψη.
- Επιβεβαίωση ασφαλιστικής κάλυψη για περίπτωση έκτακτης ανάγκης.
- Προβολή ιστορικού συνταγογραφήσεων.
- Προβολή πληροφοριών φαρμακείων.
- Αποστολή κάρτας ασφάλισης στον ιατρό.
- Αποθήκευση ιστορικού εμβολιασμών.

- Κατάλογος με άτομα που μπορούν να ενημερωθούν σε περίπτωση έκτακτης ανάγκης.
- Λήψη προειδοποιήσεων για επερχόμενες συναντήσεις υγείας και συνταγογραφήσεων.
- Εξατομικευμένες συμβουλές σχετικά με τη διαχείριση των χρόνιων παθήσεων
- Αποστολή ιατρικών μετρήσεων στον ιατρό.

Το AllOne Mobile, προσφέρει μεγάλες δυνατότης στους πελάτες μέσα από το κινητό τους τηλέφωνο. Αποτελεί μέρος ολοκληρωμένης λύσης ηλεκτρονικής διαχείρισης και της υγείας του πελάτη. Όμως εξαιτίας του γεγονότος ότι αποτελεί μέρος ευρύτερης λύσης το AllOne Mobile δεν είναι αυτόνομη υλοποίηση. Ο πελάτης της υπηρεσίας δεν έχει τη δυνατότητα τροποποίησης του ιατρικού του φακέλου κατά την επίσκεψή του σε ιατρό ο οποίος δε μετέχει στην συγκεκριμένη υπηρεσία. Επομένως το σύστημα είναι λειτουργικό μόνο όταν υπάρχει ιατρικός φορέας που να το υποστηρίζει.

13.2 MedicalRecord to Go

Το MedicalRecord to Go είναι μια αυτόνομη υλοποίηση ηλεκτρονικού Ιατρικού φακέλου σε κινητό τηλέφωνο και μνήμη USB. Ο ασθενής/ιατρός έχει πρόσβαση στον ιατρικό φάκελο από κινητό τηλέφωνο και υπολογιστή. Η διασύνδεση μεταξύ κινητού τηλεφώνου και υπολογιστή είναι ασύρματη. Σημαντικό πλεονέκτημα της υλοποίησης είναι η υποστήριξη του Clinical Document Architecture (CDA) του Οργανισμού Health Level 7 (HL7). Με αυτό τον τρόπο επιτρέπεται η επικοινωνία μεταξύ συστημάτων διαφορετικών κατασκευαστών.

Η υλοποίηση υποστηρίζει την αποθήκευση εγγράφων και εικόνων σε ηλεκτρονική μορφή, όπως κείμενα Microsoft Word και Excel, PDF, απλού κείμενου και εικόνες JPG, GIF, PNG, TIFF, BMP. Οι πληροφορίες αποθηκεύονται κρυπτογραφημένες. Για την ανάγνωση και τροποποίηση του ιατρικού φακέλου στον υπολογιστή απαιτείται οποιοδήποτε φυλλομετρητής ιστού.

14. Απαιτήσεις – Προδιαγραφές υψηλού επιπέδου

14.1 Κινητό Τηλέφωνο

Παρακάτω περιγράφονται αναλυτικά οι προδιαγραφές της υλοποίησης στο κινητό τηλέφωνο:

14.1.1 Λειτουργίες

14.1.1.1 Είσοδος στην εφαρμογή με αναγνωριστικό ορισμένο από τον χρήστη. Δυνατότητα αλλαγής του αναγνωριστικού

- Μόλις εκκινήσει η εφαρμογή θα εμφανίζεται μια οθόνη όπου ο χρήστης θα πρέπει να εισάγει αναγνωριστικό.
- Το αναγνωριστικό θα είναι ένας τετραψήφιος αριθμός.
- Σε περίπτωση εισαγωγής λανθασμένου αναγνωριστικού θα εμφανίζεται μήνυμα λάθους.
- Θα υπάρχει δυνατότητα εξόδου από την εφαρμογή από το μενού.
- Με την είσοδο του σωστού αναγνωριστικού η εφαρμογή θα μεταφέρεται στο κυρίως μενού όπου θα υπάρχει λίστα με επιλογές.
- Στη λίστα του κυρίως μενού θα υπάρχει επιλογή αλλαγής αναγνωριστικού. Ο χρήστης θα εισάγει το νέο αναγνωριστικό και θα εμφανίζεται μήνυμα επιτυχούς αλλαγής του.

14.1.1.2 Στοιχειά άμεσης ανάγκης στην αρχική οθόνη της εφαρμογής

- Στην αρχική οθόνη, δηλαδή στην οθόνη όπου εισάγεται το αναγνωριστικό εισόδου, εμφανίζονται στοιχεία που έχουν να κάνουν με την παροχής πρώτης βοήθειας στον ασθενή, πχ. Αλλεργίες ή ασθένειες που χρειάζονται άμεση περίθαλψη όπως ο διαβήτης.
- Τα στοιχεία αυτά δεν μπορούν να τροποποιηθούν από το κινητό τηλέφωνο, παρά μόνο από την εφαρμογή στον ηλεκτρονικό υπολογιστή.

14.1.1.3 Παρουσίαση και ανάγνωση του ιατρικού φακέλου από τον ασθενή

- Στη λίστα επιλογών του κυρίως μενού θα υπάρχει επιλογή ανάγνωσης του ιατρικού φακέλου.
- Ο ιατρικός φάκελος θα είναι χωρισμένος σε μέρη, τα οποία περιγράφονται αναλυτικά στην παράγραφο Δομή Ιατρικού Φακέλου και κάθε μέρος του ιατρικού φακέλου θα εμφανίζεται σε διαφορετική οθόνη.
- Ο χρήστης θα μπορεί να μεταβεί στην επόμενη/προηγούμενη οθόνη του ιατρικού φακέλου από το μενού της εφαρμογής.
- Θα υπάρχει δυνατότητα τροποποίησης το δημογραφικών στοιχείων του ασθενούς και του πλησιέστερου συγγενικού του προσώπου.
- Με την επιλογή αποθήκευσης του ιατρικού φακέλου θα εμφανίζεται μήνυμα επιτυχίας και ο χρήστης θα μεταφέρεται στο κυρίως μενού.

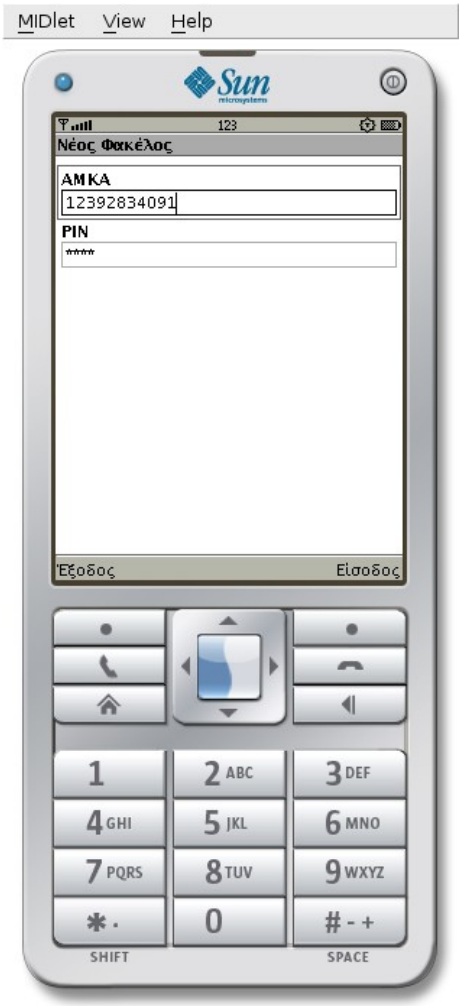
- Όλες οι πληροφορίες που είναι αποθηκευμένες στον ιατρικό φάκελο, με εξαίρεση των δημογραφικών στοιχείων του ασθενούς και του συγγενικού προσώπου, θα είναι μόνο προς ανάγνωση. Δεν υπάρχει δυνατότητα τροποποίησης τους από την εφαρμογή.

14.1.1.4 Ασύρματη σύνδεση με τον υπολογιστή του ιατρού και αποστολή ιατρικού φακέλου προς ανάγνωση και ενημέρωση

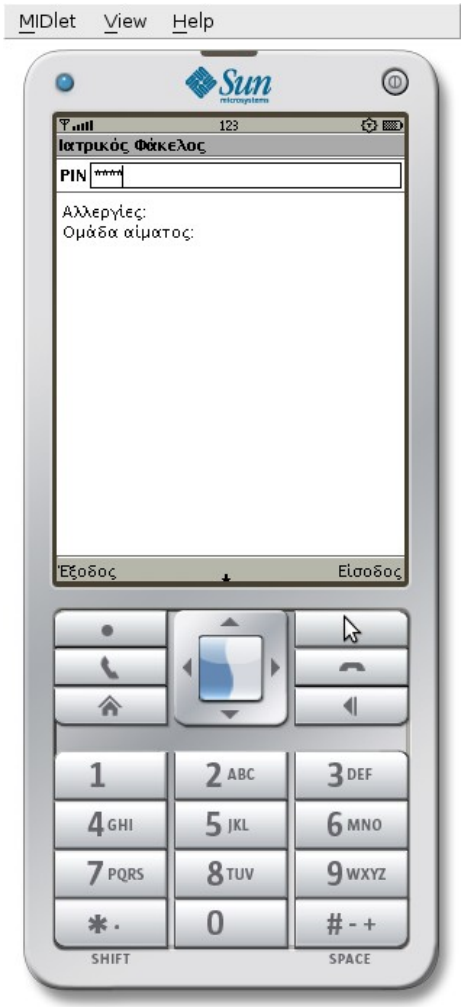
- Από το κυρίως μενού θα υπάρχει δυνατότητα ενημέρωσης του ιατρικού φακέλου. Με την επιλογή ενημέρωσης ο χρήστης θα μεταφέρεται σε οθόνη με τις δυνατές επιλογής ενημέρωσης.
- Στην οθόνη ενημέρωσης ο χρήστης επιλέγει την σύνδεση με υπολογιστή. Κατά την σύνδεσης και της αποστολής ή λήψης του ιατρικού φακέλου η εφαρμογή θα βρίσκεται σε κατάσταση αναμονής.
- Στην κατάσταση αναμονής η εφαρμογή θα εμφανίζει ένα πενταψήφιο αναγνωριστικό το οποίο πρέπει να εισαχθεί στον υπολογιστή του ιατρού έτσι ώστε να καταστεί δυνατή η επικοινωνία. Το αναγνωριστικό είναι διαφορετικό από αυτό που χρησιμοποιείται για την είσοδο στην εφαρμογή.
- Ένα μήνυμα θα ειδοποιεί τον χρήστη για την επιτυχή ή ανεπιτυχή αποστολή/λήψη του ιατρικού φακέλου.
- Με την ολοκλήρωση της διαδικασίας ο χρήστης θα μεταφέρεται στο κυρίως μενού.

14.1.2 Γραφικό περιβάλλον Graphical User Interface (GUI)

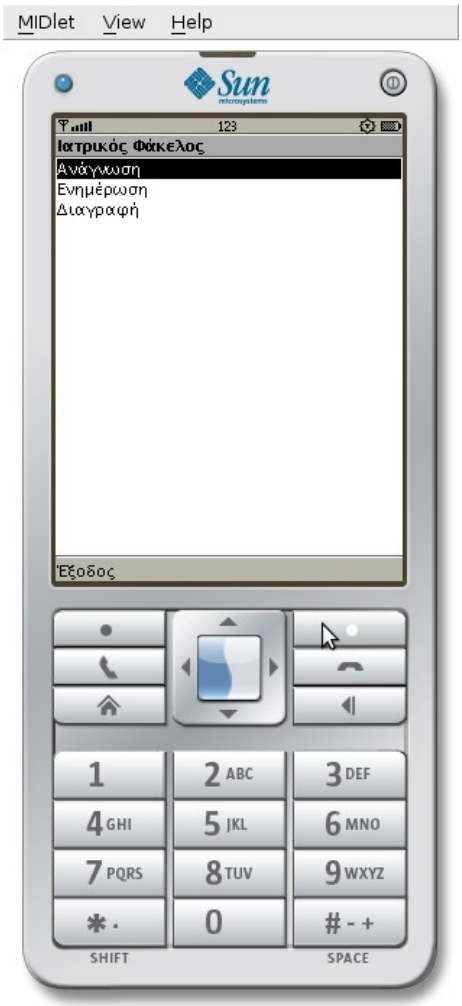
Περιγραφή διαφόρων οθονών της εφαρμογής.



Σχήμα 6: Η οθόνη εισόδου της εφαρμογής. Ο χρήστης πρέπει να εισάγει τον ΑΜΚΑ του και ένα αναγνωριστικό (τετραψήφιο αριθμό) για να δημιουργήσει καινούργιο ιατρικό φάκελο.



Σχήμα 7: Η αρχική οθόνη της εφαρμογής μετά τη δημιουργία του ιατρικού φακέλου. Ο χρήστης εισάγει το αναγνωριστικό του για να εισέλθει στην εφαρμογή. Στο υπόλοιπο μέρος της οθόνης εμφανίζονται άμεσης ανάγκης πληροφορίες σχετικά με τον ασθενή (πχ. Διαβητικός).

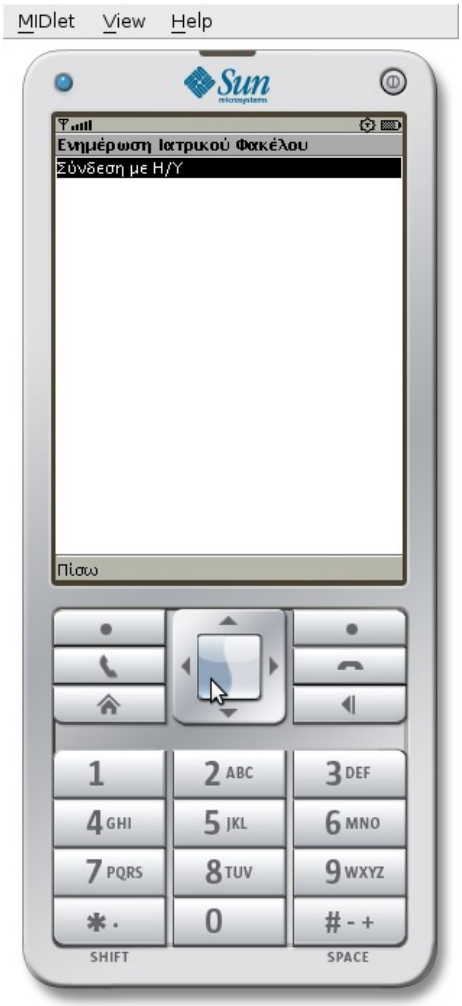


Σχήμα 8: Το κυρίως μενού. Ο χρήστης έχει τη μπορεί να διαβάσει τον ιατρικό φάκελο, με δυνατότητα τροποποίησης δημογραφικών στοιχείων, να ενημερώσει τον ιατρικό φάκελο και να τον διαγράψει.

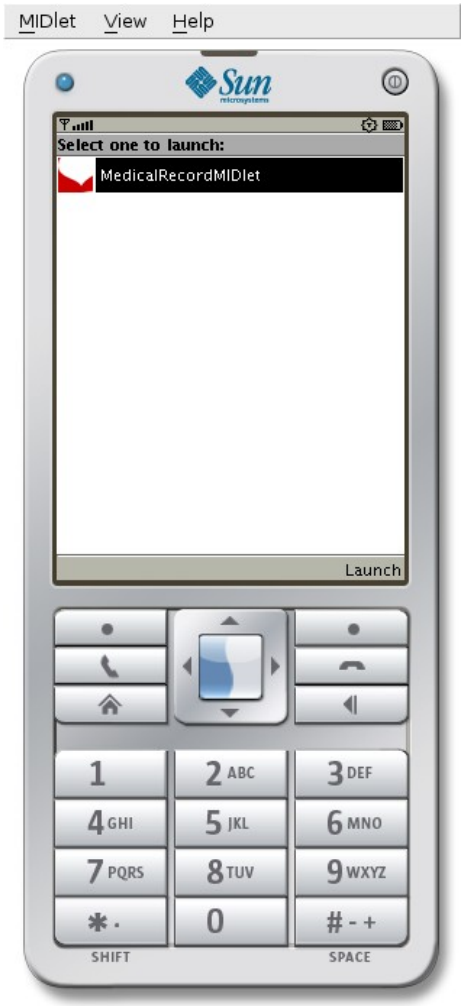
Σχήμα 9: Οθόνη από τον ιατρικό φάκελο που εμφανίζει τα δημογραφικά στοιχεία του ασθενή. Ο χρήστης έχει τη δυνατότητα τροποποίησης των στοιχείων αυτών.

Σχήμα 10: Οθόνη του ιατρικού φακέλου στην οποία εμφανίζονται τα αποτελέσματα των ιατρικών εξετάσεων. Δεν υπάρχει δυνατότητα τροποποίησης των στοιχείων αυτών.

Σχήμα 11: Οθόνη με τα στοιχεία συγγενή του ασθενή, για χρήση σε περίπτωση ανάγκης. Εμφανίζονται οι επιλογές στο μενού.



Σχήμα 12: Οθόνη ενημέρωσης του ιατρικού φακέλου. Δυνατότητα σύνδεσης με Η/Υ για την αποστολή/λήψη του ιατρικού φακέλου.



Σχήμα 13: Η εφαρμογή όπως εμφανίζεται στο κινητό τηλέφωνο μετά την εγκατάσταση.

14.1.3 Κρυπτογράφηση

14.1.3.1 JCE

Το Java Cryptography Extension (JCE) παρέχει ένα πλαίσιο και υλοποιήσεις για την κρυπτογράφηση, δημιουργία κλειδιών και βασικών συμφωνητικών, και Message Authentication Code (MAC) αλγόριθμους. Η υποστήριξη για κρυπτογράφηση περιλαμβάνει συμμετρική, ασύμμετρη, μπλοκ και τους αλγόριθμους κρυπτογράφησης

ρεύματος (stream ciphers). Το λογισμικό υποστηρίζει επίσης ασφαλή ρέματα και σφραγισμένα αντικείμενα. Το JCE, στο παρελθόν ήταν ένα προαιρετικό πακέτο (παράταση) προς την Java 2 SDK, Standard Edition (Java 2 SDK), οι εκδόσεις 1.2.x και 1.3.x. Το JCE έχει πλέον ενσωματωθεί στην Java 2 SDK v1.4 και τις μεταγενέστερες εκδόσεις Java 5 SDK και Java 6 SDK.

Το JCE βασίζεται στις ίδιες αρχές σχεδιασμού που βρέθηκαν σε άλλα σημεία του JCA: την ανεξαρτησία της εφαρμογής και, εφόσον είναι δυνατόν, την ανεξαρτησία από τον αλγόριθμο. Χρησιμοποιεί την ίδια αρχιτεκτονική "παρόχου". Οι πάροχοι που έχουν υπογραφή από έναν αξιόπιστο φορέα μπορούν να συνδεθούν στο πλαίσιο JCE, καθώς και νέοι αλγόριθμοι μπορούν να προστεθούν άνευ ραφής.

Το JCE API καλύπτει:

- Συμμετρική κρυπτογράφηση στοίβας, όπως τις DES, RC2 και IDEA.
- Συμμετρική κρυπτογράφηση ρεύματος, όπως RC4.
- Ασύμμετρη κρυπτογράφηση, όπως RSA.
- Κρυπτογράφηση βασισμένη σε κωδικό Password-based encryption (PBE).
- Συμφωνητικό κλειδιού.

14.1.3.2 SATSA

Το Security and Trust Services APIs (SATSA) ορίζεται στο Java Specification Request JSR177 και παρέχει πρόσβαση σε έξυπνες κάρτες και κρυπτογραφικές δυνατότητες για εφαρμογές που τρέχουν σε μικρές συσκευές. Το SATSA είναι βασισμένο στο JCE. Οι προδιαγραφές SATSA ορίζουν τέσσερα διαφορετικά APIs:

- SATSA-APDU επιτρέπει στις εφαρμογές να επικοινωνούν με εφαρμογές έξυπνων καρτών χρησιμοποιώντας ένα χαμηλού επιπέδου πρωτόκολλο.
- SATSA-JCRMI παρέχει μια εναλλακτική μέθοδο για την επικοινωνία με εφαρμογές έξυπνων καρτών χρησιμοποιώντας ένα απομακρυσμένου αντικειμένου πρωτόκολλο.
- SATSA-PKI επιτρέπει στις εφαρμογές να χρησιμοποιήσουν μια έξυπνη κάρτα για την ψηφιακή υπογραφή των δεδομένων και να διαχειρίζονται τα πιστοποιητικά χρήστη.
- SATSA-CRYPTO είναι ένα γενικού σκοπού κρυπτογράφησης API που υποστηρίζει πέψης μήνυμα, ψηφιακές υπογραφές, και αλγόριθμους κρυπτογράφησης.

Το SATSA έχει υλοποιήσεις σε Mobile Information Device Profile (MIDP) εφαρμογές, αν και η χρήση του σε άλλες πλατφόρμες Java ME είναι επίσης δυνατή.

14.1.3.3 Bouncy Castle

Το Bouncy Castle είναι μια συλλογή από APIs που χρησιμοποιούνται στην κρυπτογραφία Περιλαμβάνει APIs για τις γλώσσες προγραμματισμού Java και η C #.

Το χαμηλού επιπέδου, ή «ελαφρύ», API αποτελεί σύνολο APIs που εφαρμόζουν όλους τους βασικούς κρυπτογραφικούς αλγόριθμους. Η πρόθεση είναι να χρησιμοποιηθούν χαμηλού επιπέδου API στις περιορισμένων πόρων συσκευές κινητής τηλεφωνίας ή όπου δεν υπάρχει εύκολη πρόσβαση στις βιβλιοθήκες Java Cryptography Extension JCE.

14.1.3.4 Επιλογή παρόχου

Για την εφαρμογή στο κινητό τηλέφωνο έχει επιλεγεί το SATSA αντί του Bouncy Castle για τους παρακάτω λόγους:

- Συμβατότητα με περισσότερες συσκευές κινητής τηλεφωνίας.
- Πολύ μικρότερο μέγεθος βιβλιοθηκών.

14.1.3.5 Επιλογή αλγορίθμου

Το SATSA υποστηρίζει τρεις αλγόριθμους συμμετρικού κλειδιού, τους DES, Triple-DES και AES. Ο DES απορρίφθηκε λόγω των γνωστών αδυναμιών του. Για την υλοποίηση έχει επιλεγεί ο αλγόριθμος AES για τους λόγους που αναφέρονται παρακάτω:

- Μεγαλύτερη ασφάλεια σε σχέση με τον DES.
- Ταχύτερη κρυπτογράφηση σε σχέση με τον Triple-DES.
- Μικρότερο μέγεθος κρυπτογραφημένων δεδομένων σε σχέση με τον Triple-DES.

14.1.4 Διασυνδεσιμότητα (Bluetooth)

14.1.5 MIDlet

Το MIDlet είναι μια εφαρμογή που χρησιμοποιεί το MIDP του CLDC του περιβάλλοντος Java ME.

Κύκλος ζωής ενός MIDlet

Τα MIDlets διαχειρίζονται από το λογισμικό διαχείρισης εφαρμογών που είναι ενσωματωμένο μέσα στη συσκευή (κινητό τηλέφωνο). Αυτό συμβαίνει γιατί μια εφαρμογή εν εκτελέσει μπορεί να διακοπεί σε οποιοδήποτε σημείο από εξωτερικά γεγονότα, όπως εισερχόμενες κλήσεις.

Ο διαχειριστής εφαρμογών ελέγχει τις δραστηριότητες των πολλαπλών MIDlets στο περιβάλλον Java ME σε μια συσκευή και μπορεί να επιλέγει ποια MIDlets είναι ενεργά μια συγκεκριμένη χρονική στιγμή με το να εκκινεί και να διακόπτει το κάθε ένα ξεχωριστά.

Υπάρχουν τρεις δυνατές καταστάσεις στον κύκλο ζωής ενός MIDlet:

- *paused* – Το αρχικοποιημένο MIDlet έχει κατασκευαστεί και είναι ανενεργό.
- *active* – Το MIDlet είναι ενεργό.
- *destroyed* - Το MIDlet έχει τερματιστεί και είναι έτοιμο για ανάκτηση από το συλλέκτη σκουπιδιών (grabage collector).

Ο διαχειριστής εφαρμογων δημιουργεί MIDlets, συνήθως σε απάντηση στο αίτημα των χρηστών, σε κατάσταση παύσης. Σε κάποιο σημείο μετά από τη δημιουργία, η κατάσταση του MIDlet αλλάζει από παυμένο (paused) σε ενεργό (active). Όταν ένα MIDlet τερματιστεί αλλάζει η κατάσταση σε καταστρεμμένο (*destroyed*).

Τα MIDlets πακετάρονται σε ένα αρχείο .jar το οποίο περιέχει ένα αρχείο εκδήλωσης (Manifest-file) που υποδεικνύει ποιες κλάσεις υλοποιούν πιο MIDlet. Το αρχείο .jar εκτός από κλάσεις μπορεί να περιέχει και άλλα αρχεία όπως φωτογραφίες. Ένα δεύτερο αρχείο .jad περιέχει πληροφορίες για την τοποθεσία του .jar καθώς επίσης και μια λίστα με τα MIDlets της σουίτας και άλλες ιδιότητες.

Το .jad μπορεί να εξαπλωθεί (deploy) με δύο τρόπους.

- Εξάπλωση Over the Air (OTA), όπου τα αρχεία .jad και .jar έχουν αναμορφωθεί (upload) σε ένα εξυπηρετητή ιστού (web server) ο οποίος είναι προσβάσιμος στη συσκευή μέσω HTTP. Ο χρήστης κατεβάζει το αρχείο .jad και εγκαθιστά τα αναγκαία MIDlets.
- Τοπική εξάπλωση, όπου το MIDlet μεταφέρεται στην συσκευή σε μια σύνδεση εκτός δικτύου, όπως είναι το Bluetooth και το IrDa και μπορεί να προϋποθέτει τη χρήση υλικού συγκεκριμένου για τη συσκευή.

14.1.6 RMS

Για την αποθήκευση των δεδομένων στο κινητό τηλέφωνο χρησιμοποιείται το Record Management System (RMS) των MIDlets για την έμμονη (persistence) αποθήκευση δεδομένων. Το RMS είναι μια απλή βάση δεδομένων προσανατολισμένη σε εγγραφές.

14.1.6.1 RecordStore

Τα Record Stores είναι δυαδικά αρχεία που εξαρτάται από την πλατφόρμα, επειδή έχουν δημιουργηθεί σε τοποθεσίες εξαρτώμενες της πλατφόρμας. Τα MIDlets μπορούν να δημιουργήσουν πολλαπλά RecordStores (αρχεία δεδομένων), με διαφορετικά ονόματα.

Το RMS APIs παρέχουν τις ακόλουθες λειτουργίες:

- Επιτρέπουν στα MIDlets να χειραγωγήσουν (προσθέτουν και αφαιρούν) τις εγγραφές σε ένα RecordStore.
- Επιτρέπουν σε διαφορετικά MIDlets της ίδιας εφαρμογής να μοιράζονται εγγραφές με απευθείας πρόσβαση των RecordStores των άλλων MIDlets.
- Δεν παρέχουν μηχανισμό ανταλλαγής εγγραφών μεταξύ MIDlets διαφορετικών εφαρμοφών.

Ονόματα RecordStores

Το όνομα ενός Record Store είναι ευαίσθητο έναντι είδους των στοιχείων {κεφαλαίων-πεζών} (case-sensitive), και δεν μπορεί να είναι μεγαλύτερο από 32 χαρακτήρες. Επίσης, ένα MIDlet δεν μπορεί να δημιουργήσει δύο Record Stores με το ίδιο όνομα στην ίδια εφαρμογή, αλλά μπορεί να δημιουργήσει ένα Record Store με το ίδιο όνομα ως ένα MIDlet σε κάποια άλλη εφαρμογή. Όταν δημιουργείτε ένα νέο Record Store, αποθηκεύεται σε ένα ευρετήριο που ονομάζεται *NOJAM*. Παραδείγματος χάριν, ας υποθέσουμε ότι χρησιμοποιείται το Wireless Toolkit και ότι έχει εγκατασταθεί στο /opt/j2me/WTK/. Αν το όνομα του έργου είναι RecordStoreExample και το όνομα του Record Store είναι myrecordstore, το RecordStore δημιουργείται κάτω από το ευρετήριο /opt/j2me/NOJAM και έχει όνομα myrecordstore.db.

14.2 Ηλεκτρονικός Υπολογιστής

Η εφαρμογή στον ηλεκτρονικό υπολογιστή περιγράφεται αναλυτικά παρακάτω. Οι γενικές λειτουργίες της εφαρμογής είναι η σύνδεση με το κινητό τηλέφωνο και λήψη του ιατρικού φακέλου, η ανάγνωση του ληφθέντος ιατρικού φακέλου από τον υπολογιστή, η τροποποίηση των στοιχείων του φακέλου και η αποστολή του ενημερωμένου ιατρικού φακέλου στο κινητό τηλέφωνο.

Να σημειωθεί ότι ο ιατρικός φάκελος δεν μπορεί να αποθηκευτεί στον ηλεκτρονικό υπολογιστή. Με το κλείσιμο της εφαρμογής όλα τα στοιχεία του φακέλου χάνονται.

14.2.1 Λειτουργίες

14.2.1.1 Ασύρματη σύνδεση με κινητό τηλέφωνο του ασθενή. Λήψη του ιατρικού φακέλου

- Ο ιατρός αφού εκκινήσει την εφαρμογή στον ηλεκτρονικό υπολογιστή θα επιλέξει την λήψη του ιατρικού φακέλου από το μενού επιλογών.
- Συμπληρώνει ένα 5ψήφιο αναγνωριστικό μιας χρήσης, το οποίο θα εμφανίζεται στην εφαρμογή του κινητού τηλεφώνου και πατάει το κουμπί σύνδεσης.
- Την πρώτη φορά που θα γίνει η σύνδεση θα χρειαστεί να γίνει ζευγοποίηση των δύο συσκευών (κινητού τηλεφώνου ασθενή και ηλεκτρονικού υπολογιστή ιατρού), διαδικασία που είναι τυποποιημένη για το πρωτόκολλο Bluetooth. Να σημειωθεί ότι η διαδικασία ζευγοποίησης ενδέχεται να διαφέρει ανάλογα με την υλοποίηση της στοίβας Bluetooth που χρησιμοποιείται στον ηλεκτρονικό υπολογιστή του ιατρού.
- Με την ολοκλήρωση της διαδικασίας ζευγοποίησης, αρχίζει η σύνδεση των δύο εφαρμογών, κινητού τηλεφώνου και ηλεκτρονικού υπολογιστή, μέσω του πρωτοκόλλου OBEX που υποστηρίζεται από το Bluetooth.
- Η εφαρμογή στον ηλεκτρονικό υπολογιστή ζητάει και λαμβάνει τον ιατρικό φάκελο από την εφαρμογή στο κινητό τηλέφωνο.
- Με τη λήψη του ιατρικού φακέλου η σύνδεση τερματίζεται.
- Τα στοιχεία του ιατρικού φακέλου φορτώνονται στις αντίστοιχες καρτέλες της εφαρμογής και η διαδικασία τερματίζεται.

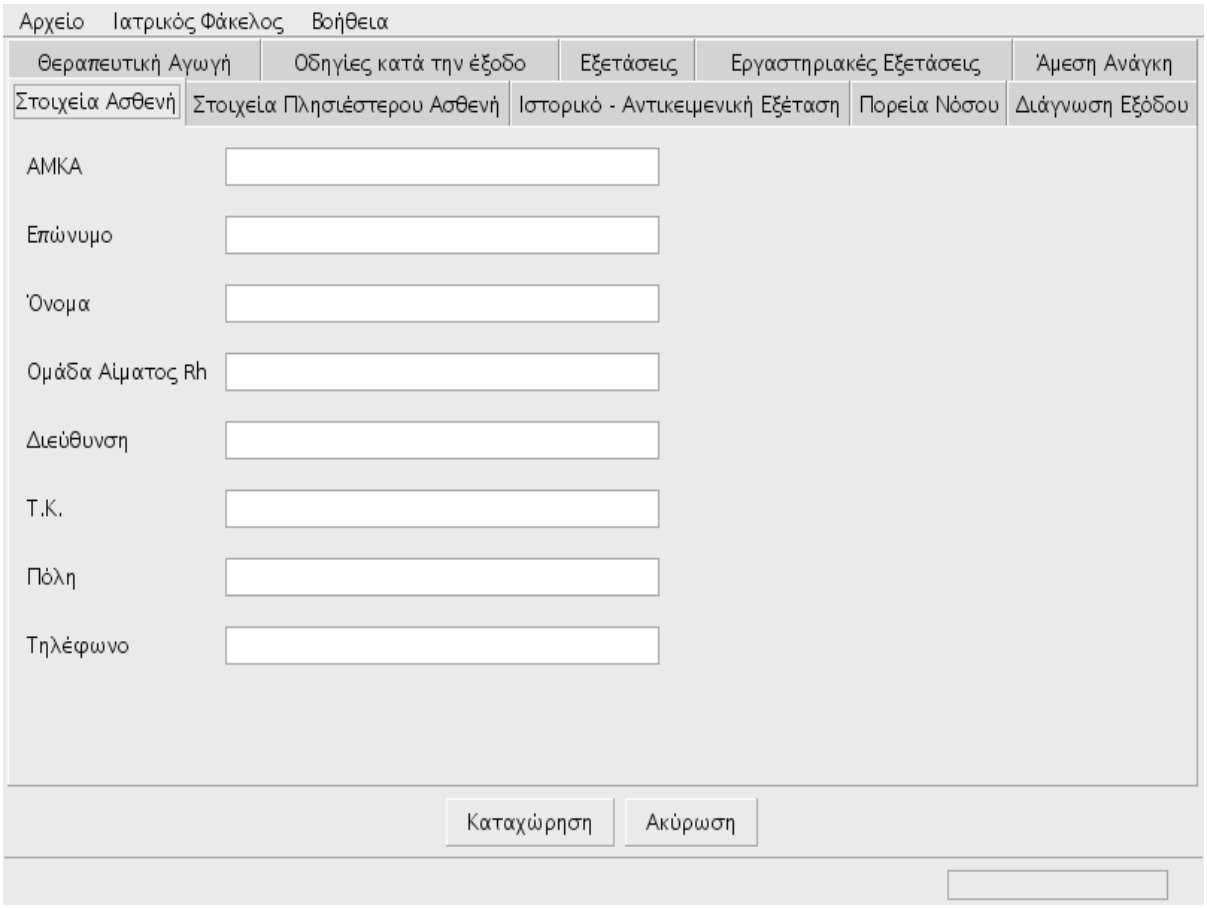
14.2.1.2 Τροποποίηση/Προσθήκη στοιχείων στον ιατρικό φάκελο

- Ο ιατρικός φάκελος είναι χωρισμένος σε διάφορες καρτέλες, με τον ίδιο τρόπο που είναι χωρισμένος στην εφαρμογή του κινητού τηλεφώνου σε διαφορετικές οθόνες.
- Η κάθε καρτέλα είναι απλή και περιέχει μικρό αριθμό πεδίων για να είναι παρόμοια με αυτή στο κινητό τηλέφωνο. Δεν υπάρχουν επιπλέον δυνατότητες στην εφαρμογή του ηλεκτρονικού υπολογιστή σε σχέση με αυτή του κινητού τηλεφώνου.
- Ο ιατρός πέραν της δυνατότητας ανάγνωσης του ιατρικού φακέλου του ασθενή, έχει την δυνατότητα να τροποποιήσει οποιοδήποτε στοιχείο του φακέλου.

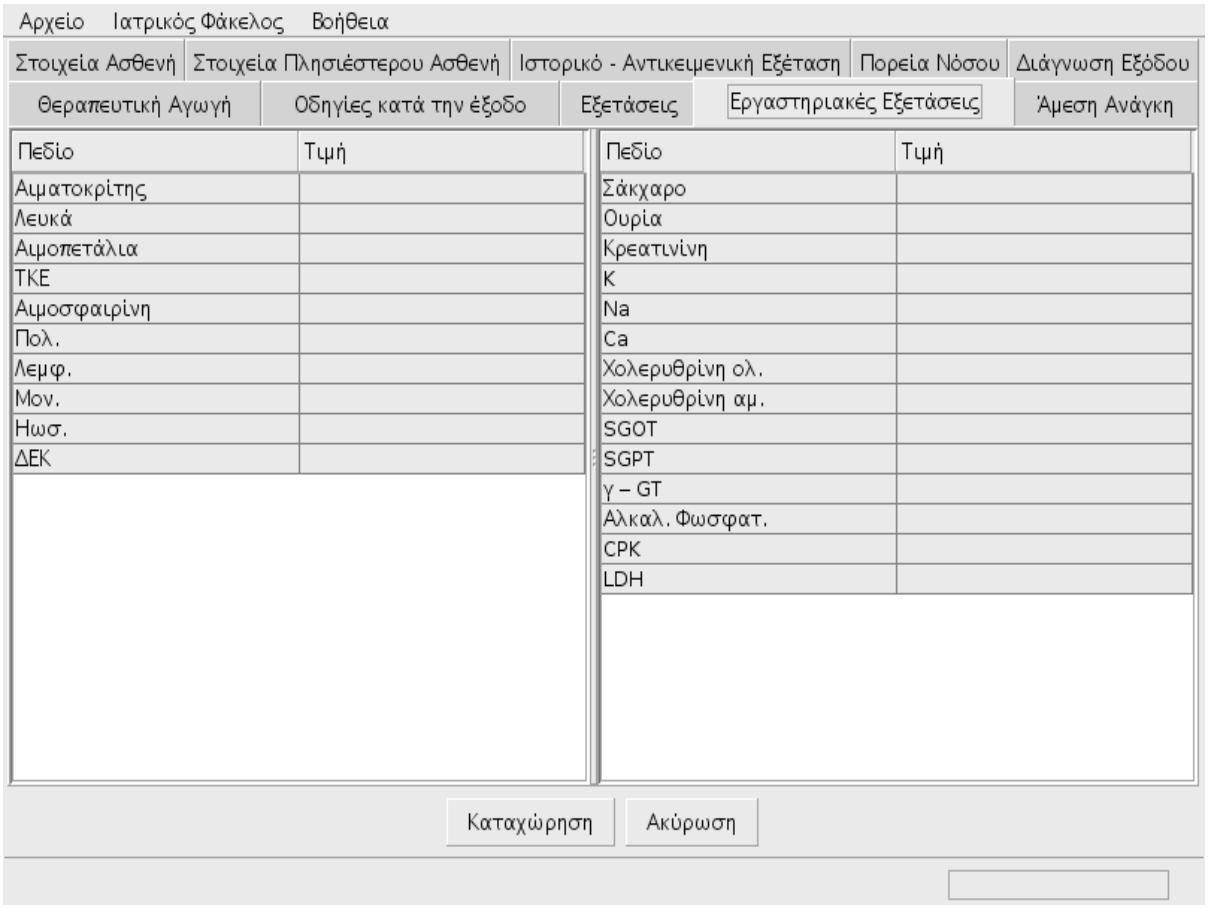
14.2.1.3 Αποστολή του ενημερωμένου ιατρικού φακέλου στο κινητό τηλέφωνο του ασθενή

- Μετά την ολοκλήρωση των τροποποιήσεων ο ιατρός επιλέγει την αποστολή του ιατρικού φακέλου, στο κινητό τηλέφωνο του ασθενή, από το μενού επιλογών της εφαρμογής.
- Για την αποστολή του ενημερωμένου ιατρικού φακέλου χρησιμοποιείται ο ίδιος 5ψήφιος κωδικός που είχε χρησιμοποιηθεί στη λήψη του ιατρικού φακέλου.
- Η διαδικασία σύνδεσης πρέπει να εκκινηθεί και πάλι από το κινητό τηλέφωνο.
- Μήνυμα επιτυχίας/αποτυχίας ενημερώνει τον ιατρό για την έκβαση της αποστολής.

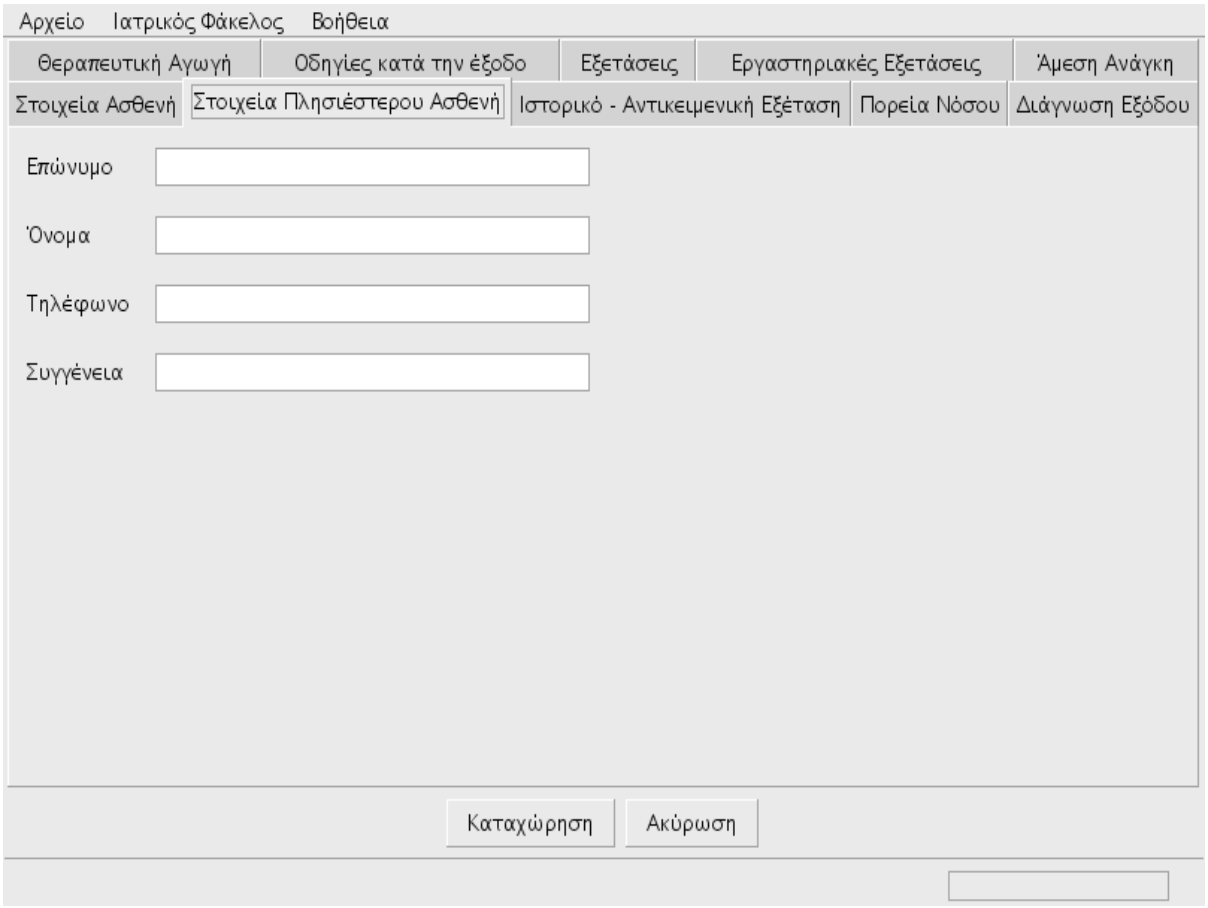
14.2.2 Γραφικό περιβάλλον Graphical User Interface (GUI)



Σχήμα 14: Η καρτέλα με τα δημογραφικά στοιχεία του ασθενή.



Σχήμα 15: Η καρτέλα με τα αποτελέσματα των εργαστηριακών εξετάσεων.



Σχήμα 16: Η καρτέλα με τα στοιχεία του πλησιέστερου συγγενικού προσώπου του ασθενή.

14.3 Java Swing

Το Swing είναι ένα εργαλειοσύνολο GUI για Java. Είναι μέρος των Java Foundation Classes (JFC) της Sun Microsystems - ένα API για παροχή μια γραφικής διεπαφής χρήστη (GUI) για τα προγράμματα Java.

Το Swing αναπτύχθηκε για να παρέχει ένα πιο εξελιγμένο σύνολο από GUI στοιχεία από το προηγούμενο Abstract Window Toolkit. Το Swing παρέχει μια εγγενής εμφάνιση και αίσθηση (look and feel) η οποία προσομοιώνει την εμφάνιση και και την αίσθηση των διαφόρων πλατφορμών. Υποστηρίζει επίσης μια βυσματώσιμη (pluggable) εμφάνιση και αίσθηση που επιτρέπει στις εφαρμογές να έχουν μια εμφάνιση και αίσθηση ανεξάρτητη προς την υποκείμενη πλατφόρμα.

14.4 Διαδίκτυο

- Σύνδεση του προσωπικού του κλινικού εργαστηρίου (πχ αιματολογικό) στην ιστοσελίδα.
- Είσοδος στην σελίδα με αναγνωριστικά.
- Συμπλήρωση των αποτελεσμάτων των εργαστηριακών εξετάσεων.
- Έγκριση και καταχώρηση των αποτελεσμάτων.
- Αυτόματη αποστολή αποτελεσμάτων στο κινητό τηλέφωνο του ασθενή.

15. ΑΜΚΑ

Ο ΑΜΚΑ (Αριθμός Μητρώου Κοινωνικής Ασφάλισης) είναι ουσιαστικά η ταυτότητα εργασίας και ασφάλισης κάθε εργαζόμενου, συνταξιούχου και προστατευόμενου μέλους της οικογένειάς τους στη χώρα μας. Με τον ΑΜΚΑ θα εξυπηρετήστε πιο εύκολα και γρήγορα σε όλες σας τις συναλλαγές που αφορούν την εργασία και την ασφάλιση όπως: να προχωρήσετε σε έναρξη απασχόλησης και ασφάλισης, να καταβάλετε τις ασφαλιστικές σας εισφορές, να εκδώσετε ή να ανανεώσετε το βιβλιάριο ασθένειας, να πάρετε τη σύνταξή σας ή τις οποιεσδήποτε παροχές, επιδόματα και βοηθήματα και όλα αυτά με πολύ λιγότερες καθυστερήσεις και γραφειοκρατικές διαδικασίες.

Ο ΑΜΚΑ αντικαθιστά τον Αριθμό Μητρώου (ΑΜ) που μέχρι σήμερα χορηγούν οι Ασφαλιστικοί Φορείς στους ασφαλιζόμενους και στους συνταξιούχους, κάνοντας τη ζωή σας ακόμα πιο εύκολη.

Θα είναι απαραίτητος από τον Οκτώβριο του 2009. Η απόδοσή του γίνεται σε όλα τα ΚΕΠ και τα γραφεία ΑΜΚΑ που λειτουργούν σε Ασφαλιστικά Ταμεία.

Γιατί είναι απαραίτητος ο ΑΜΚΑ;

Με τον ΑΜΚΑ και τους 13 νέους φορείς κοινωνικής ασφάλισης μπαίνουν οι βάσεις μιας νέας, λειτουργικής δομής του ασφαλιστικού συστήματος, προωθώντας και ισχυροποιώντας την ασφαλιστική μεταρρύθμιση. Έτσι:

- Μπαίνουν τα θεμέλια για σύγχρονες, απλοποιημένες παροχές σε βασικούς τομείς της καθημερινότητας του πολίτη, όπως:
    - Στον τομέα της ιατροφαρμακευτικής περίθαλψης, δρώντας άμεσα ενάντια στην κατασπατάληση των πόρων των Ταμείων, ώστε να μπορούν να βελτιώνονται οι παροχές.
    - Στον τομέα της εργασίας και της ασφάλισης του εργαζόμενου, αφού εξασφαλίζονται τα δικαιώματά του, καταπολεμώντας την εισφοροδιαφυγή.

- Στον συνταξιοδοτικό τομέα, μειώνοντας αισθητά το χρόνο έκδοσης και απονομής της σύνταξης

❍ Εκσυγχρονίζονται οι αρμόδιοι φορείς και βελτιώνονται οι παρεχόμενες υπηρεσίες, διευκολύνοντας τις συναλλαγές όλων μας

❍ Μειώνεται ο χρόνος απονομής της σύνταξης, ιδιαίτερα όταν πρόκειται για περίπτωση διαδοχικής ασφάλισης

❍ Τέλος, υποστηρίζοντας τα πρότυπα της Ευρωπαϊκής Ένωσης διευκολύνεται η ασφάλιση, η ιατροφαρμακευτική περίθαλψη και η συνταξιοδότηση των πολιτών που εργάζονται σε κάποια άλλη χώρα της Ευρωπαϊκής Ένωσης.

16. Δομή Ιατρικού Φακέλου

16.1.1 Προτυποποίηση εντύπων ενιαίας λειτουργίας των νοσοκομείων Υπουργείου Υγείας και Κοινωνικής Αλληλεγγύης ΥΥΚΑ

Η σημασία των χρησιμοποιούμενων εντύπων στους μεγάλους δημόσιους οργανισμούς, όπως είναι τα Νοσοκομεία, είναι ιδιαίτερα σημαντική. Τα χρησιμοποιούμενα έντυπα έρχονται να υποστηρίξουν συγκεκριμένες αρμοδιότητες, λειτουργίες και δραστηριότητες, αποτελώντας τα εργαλεία υλοποίησης των λειτουργιών αυτών. Οι οικονομικές και λογιστικές λειτουργίες, η προμήθεια και διακίνηση εξοπλισμού, υλικών και υπηρεσιών, η διεκπεραίωση διοικητικών λειτουργιών, η διακίνηση και διαχείριση των ασθενών, η παραγγελία εργαστηριακών εξετάσεων, η διεκπεραίωση του ιατρικού και νοσηλευτικού έργου, η τήρηση ιατρικών αρχείων κλπ αποτελούν λειτουργίες η υλοποίηση των οποίων δεν είναι νοητή χωρίς τη χρήση συγκεκριμένων εντύπων ή “φορμών”.

Σε πολλές περιπτώσεις, η ποιότητα μιας λειτουργίας καθορίζεται σε σημαντικό βαθμό από τα ίδια τα έντυπα που την υποστηρίζουν. Είναι επίσης φανερό, ότι ο σχεδιασμός και η οργάνωση των εντύπων ενός οργανισμού αντανακλά άμεσα και το επίπεδο οργάνωσης των αντίστοιχων λειτουργιών.

Στα δημόσια Νοσοκομεία της χώρας μας χρησιμοποιείται σήμερα ένας μεγάλος αριθμός διαφορετικών εντύπων για τη διεκπεραίωση των κάθε είδους εργασιών τους. Ορισμένα από τα έντυπα αυτά προβλέπονται από την υπάρχουσα νομοθεσία και το περιεχόμενό τους προσδιορίζεται με υπουργικές αποφάσεις ή εγκυκλίους. Αρκετά άλλα έντυπα είναι μεν επιβεβλημένα από τη νομοθεσία ή από τις ασκούμενες αρμοδιότητες, αλλά η μορφή και το περιεχόμενό τους δεν έχουν καθορισθεί με συγκεκριμένες αποφάσεις ή εγκυκλίους. Στις περιπτώσεις αυτές, κάθε νοσοκομείο διαμορφώνει δικά

του πρότυπα εντύπων, με αποτέλεσμα, για την ίδια λειτουργία, να χρησιμοποιούνται τελείως διαφορετικά έντυπα, πολλά από τα οποία παρουσιάζουν εμφανείς αδυναμίες και ανεπάρκειες στην υποστήριξη των αντίστοιχων λειτουργιών. Σε πολλές μάλιστα περιπτώσεις, μέσα στο ίδιο νοσοκομείο, για την ίδια λειτουργία, είναι δυνατόν να χρησιμοποιούνται περισσότερα του ενός, αναλόγως της οργανικής μονάδας που κάνει τη χρήση του εντύπου. Τέλος, σε κάθε νοσοκομείο υπάρχει ένας επιπλέον τεράστιος αριθμός εντύπων τα οποία διαμορφώνονται ad-hoc από οποιοδήποτε μέλος του προσωπικού τους, για να καλύψουν πολυποίκιλες ανάγκες, όπως τις αντιλαμβάνεται κάθε μέλος του προσωπικού. Η εικόνα αυτή υποδηλώνει την απουσία μελέτης ή προετοιμασίας για τον καθορισμό και τη διαμόρφωση των εντύπων που χρησιμοποιούνται για την υποστήριξη των νοσοκομειακών λειτουργιών.

Η μεγάλη αυτή ποικιλομορφία και η έλλειψη προτυποποίησης έχει σοβαρές αρνητικές παρενέργειες σε ότι αφορά:

- Τη δυνατότητα ενιαίας λογιστικής και μηχανογραφικής οργάνωσης των νοσοκομείων.
- Τη δυνατότητα εφαρμογής κωδικοποιήσεων.
- Τη δυνατότητα κατάρτισης τμηματικών προϋπολογισμών.
- Τη δυνατότητα τήρησης ορθών και ενημερωμένων ιατρικών και υπηρεσιακών αρχείων.
- Τη δυνατότητα επικοινωνίας μεταξύ των μονάδων.
- Τη δυνατότητα ανάπτυξης ενιαίων διαδικασιών και λειτουργιών μέσα στα νοσοκομεία.
- Τη δυνατότητα εφαρμογής ενιαίων διαδικασιών ελέγχου και διασφάλισης της ποιότητας. Όλα τα παραπάνω καθιστούν αναγκαία την ανάληψη μιας προσπάθειας για την προτυποποίηση των χρησιμοποιούμενων εντύπων, με τελικό σκοπό την εισαγωγή ενιαίων προτύπων για όλες τις θεσμοθετημένες λειτουργίες στα Νοσοκομεία του ΕΣΥ.

Σκοπός της παρούσας έκδοσης είναι η παρουσίαση των προτύπων εντύπων τα οποία αναπτύχθηκαν μέσω του προγράμματος αυτού, με σκοπό να χρησιμοποιηθούν κατ' αρχήν πιλοτικά και στη συνέχεια οριστικά, από όλα τα Νοσοκομεία του ΕΣΥ.

Οι στόχοι του προγράμματος ήταν:

- Ο καθορισμός των εντύπων που απαιτούνται για την υποστήριξη κάθε συγκεκριμένης λειτουργίας του Νοσοκομείου, με βάση τις αρμοδιότητες, τη λειτουργία και τον οργανισμό του Νοσοκομείου.

- Η διαμόρφωση προτύπων εντύπων για όλες τις βασικές λειτουργίες του Νοσοκομείου, με παράλληλη ενοποίηση και συγχώνευση των πολλαπλών εντύπων που χρησιμοποιούνται σήμερα για παραπλήσιες ή παρεμφερείς λειτουργίες.
- Η εξασφάλιση της δυνατότητας ενσωμάτωσης στα υπό ανάπτυξη έντυπα συστημάτων κωδικοποιήσεων.
- Η εξασφάλιση της δυνατότητας διεκπεραίωσης προϋπολογιστικών και λογιστικών λειτουργιών μέσω των εντύπων.
- Η χρωματική και αριθμητική κωδικοποίηση των προτύπων εντύπων.
- Η περιγραφή των βασικών μορφολογικών και λειτουργικών χαρακτηριστικών κάθε εντύπου και η αποσαφήνιση της διαδικασίας αρχειοθέτησης κάθε εντύπου.
- Η εξασφάλιση της δυνατότητας τοπικών προσαρμογών κάθε εντύπου στις ιδιαιτερότητες κάθε Νοσοκομείου
- Η παραγωγή και διάθεση των εντύπων τόσο σε γραπτή όσο και σε ηλεκτρονική μορφή.

17. Πλεονεκτήματα

- Διαρκής κατοχή και μεταφορά του ιατρικού φακέλου από τον ασθενή, χωρίς επιπλέον κόστος, εφόσον αυτός αποθηκεύεται στην υπάρχουσα συσκευή του ασθενή.
- Δυνατότητα σε διαφορετικούς ιατρούς που παρακολουθούν τον ασθενή να γνωρίζουν το πλήρες ιστορικό του.
- Άμεση πρόσβαση στον ιατρικό φάκελο από το αρμόδιο ιατρικό προσωπικό, είτε από το κινητό τηλέφωνο του ασθενή είτε από τον υπολογιστή του ιατρού.
- Εύκολη ανάγνωση και ενημέρωση του ιατρικού φακέλου, χωρίς τεχνικές γνώσεις υλοποίησης της εφαρμογής.
- Διαμόρφωση του ιατρικού φακέλου με βάση την προτυποποίηση του Υπουργείου Υγείας & Κοινωνικής Αλληλεγγύης.
- Αποθήκευση των στοιχείων του ιατρικού φακέλου σε κρυπτογραφημένη μορφή.
- Εγκατάσταση εφαρμογής στον υπολογιστή του ιατρού κατευθείαν από το διαδίκτυο, άμεσα και γρήγορα, ακόμα και κατά τη διάρκεια της συνάντησης με τον ασθενή.
- Υποστήριξη της συντριπτικής πλειοψηφίας των κινητών τηλεφώνων. Απλή και σύντομη διαδικασία εγκατάστασης της εφαρμογής στο κινητό τηλέφωνο.

- Υποστήριξη οποιουδήποτε υπολογιστή και λειτουργικού συστήματος.

Μέρος Γ

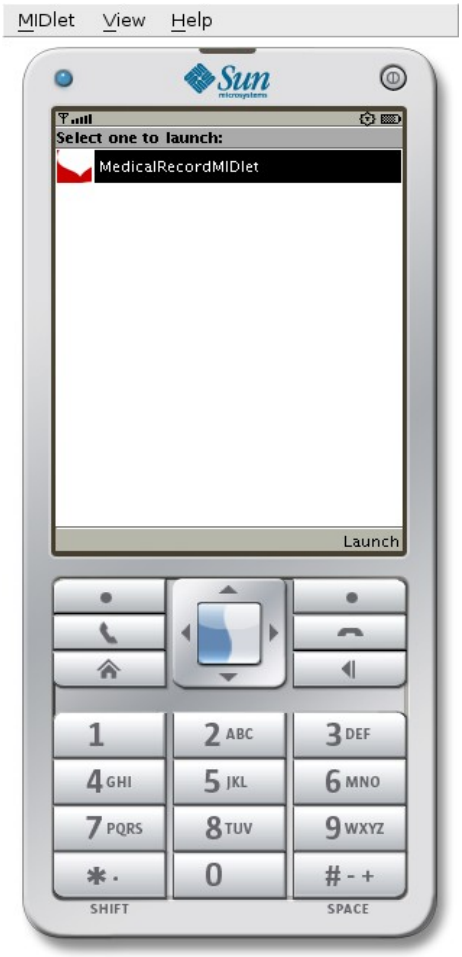
18. Εργαλεία Ανάπτυξης

18.1 Sun Java Wireless Toolkit for CLDC

Το Sun Java Wireless Toolkit είναι ένα σύνολο εργαλείων για τη δημιουργία Java εφαρμογών που τρέχουν σε συσκευές που συμφωνούν με τις Java Technology for the Wireless Industry (JTWI), JSR 185 προδιαγραφές και τις Mobile Service Architecture (MSA, JSR 248) προδιαγραφές. Αποτελείται από εργαλεία ανάπτυξης, βοηθήματα, καθώς και έναν εξομοιωτή συσκευής κινητής τηλεφωνίας.

Είναι πολύ χρήσιμο στην ανάπτυξη εφαρμογών για κινητά τηλέφωνα και δοκιμή αυτών στον εξομοιωτή.

Στα πλαίσια της διπλωματικής χρησιμοποιήθηκε ευρέως η πλατφόρμα αυτή και συγκεκριμένα ο εξομοιωτής ο οποίος επιτρέπει τον έλεγχο της λειτουργίας μιας υλοποίησης χωρίς να χρειάζεται να μεταφέρεται κάθε φορά στο κινητό τηλέφωνο.



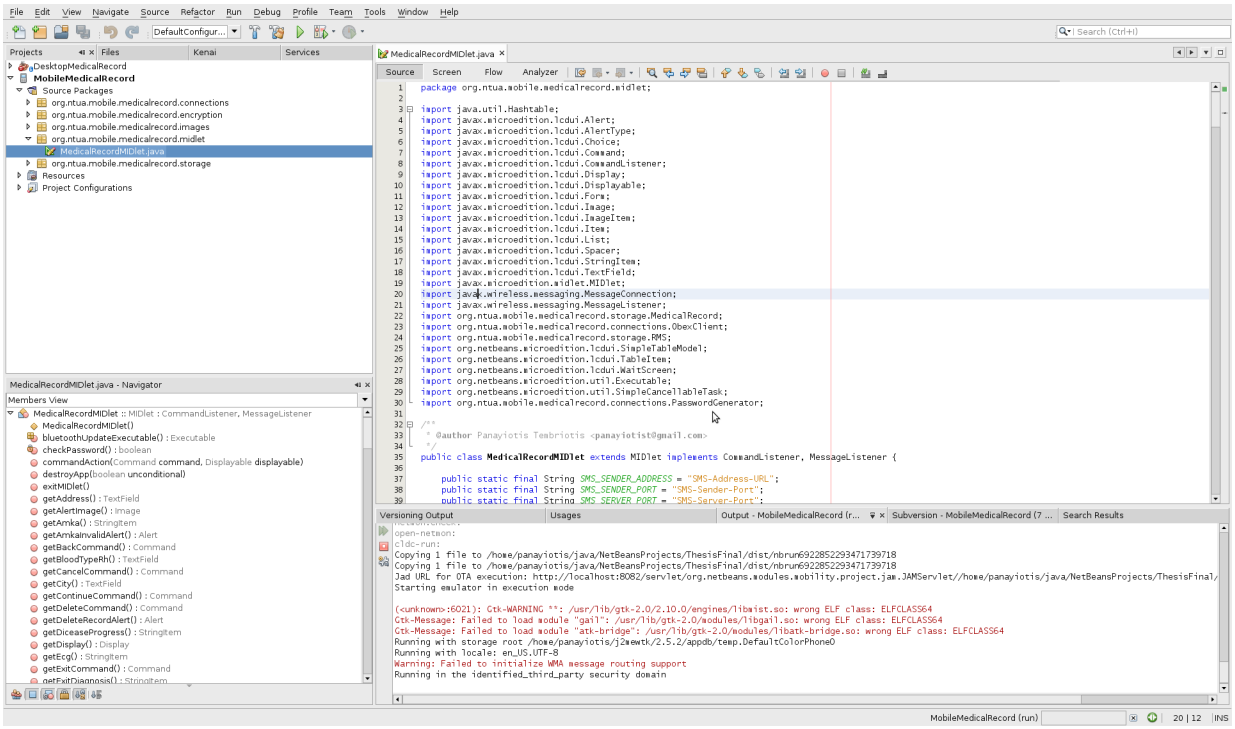
Σχήμα 17: Sun Java Wireless Toolki for CLDC

18.2 Netbeans

Το **NetBeans** αναφέρεται τόσο στην πλατφόρμα πλαισίου για τις εφαρμογές Java, και ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη με Java, JavaScript, PHP, Python, Ruby, Groovy, C, C + + , Scala, Clojure και πολύ περισσότερο.

Το NetBeans IDE είναι γραμμένο σε Java και τρέχει παντού όπου μια εικονική μηχανή Java JVM είναι εγκατεστημένη, συμπεριλαμβανομένων των Windows, Mac OS, Linux, και Solaris. Ένα Java Development Kit (JDK) απαιτείται για τη λειτουργία της ανάπτυξης Java.

Το NetBeans πλατφόρμα επιτρέπει στις εφαρμογές που θα αναπτυχθούν από μια σειρά αρθρωτών στοιχείων λογισμικού που ονομάζονται modules. Εφαρμογές που βασίζονται στην πλατφόρμα NetBeans (συμπεριλαμβανομένου του NetBeans IDE) μπορεί να επεκταθούν από τρίτους.



Σχήμα 18: Netbeans

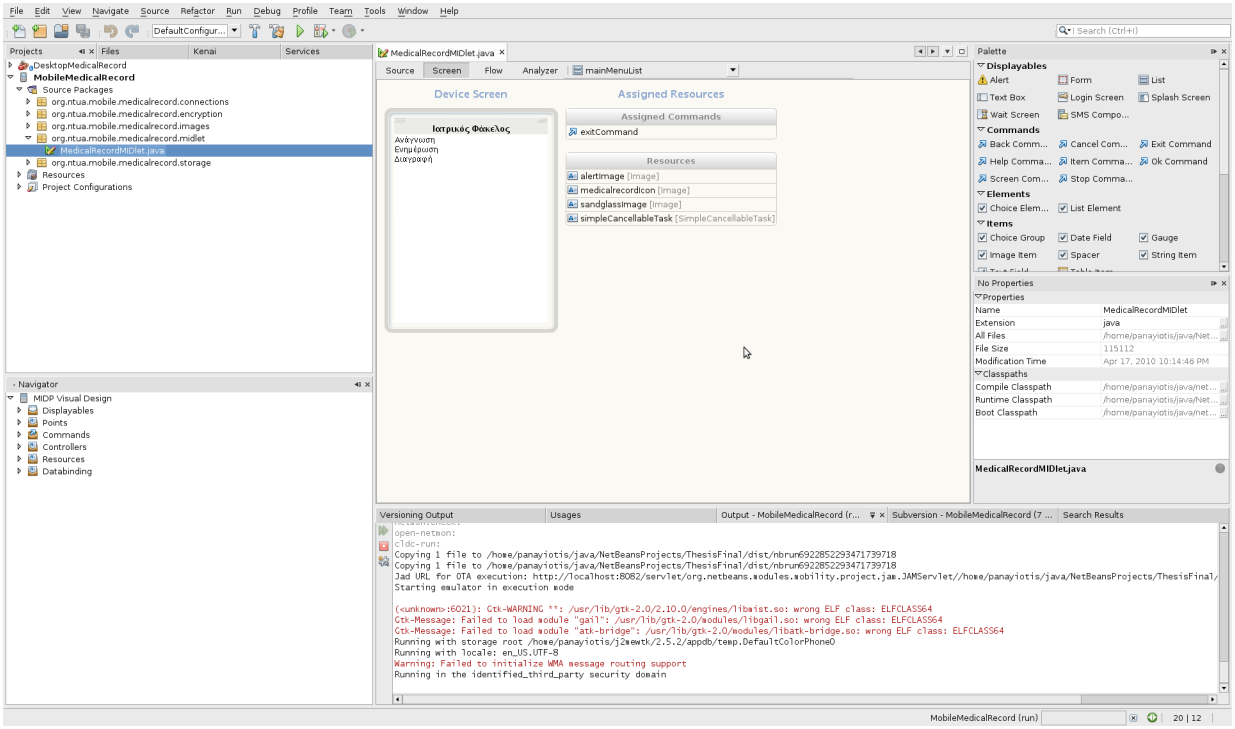
18.2.1 Java ME

Το Netbeans παρέχει εργαλεία για την ευκολότερη ανάπτυξη εφαρμογών Java ME. Τα εργαλεία αυτά περιγράφονται πιο κάτω:

Visual Mobile Designer (VSD)

Εργαλείο για την γραφική ανάπτυξη εφαρμογών MIDlet για κινητά τηλέφωνα. Τα MIDlets αποτελούν το σημείο εκκίνησης μιας υλοποίησης. Είναι το γραφικό περιβάλλον το οποίο βλέπει αρχικά ο χρήστης που εκκινεί την εφαρμογή.

Το VSD είναι ένα εξαιρετικό εργαλείο στην ανάπτυξη εφαρμογών MIDlet.



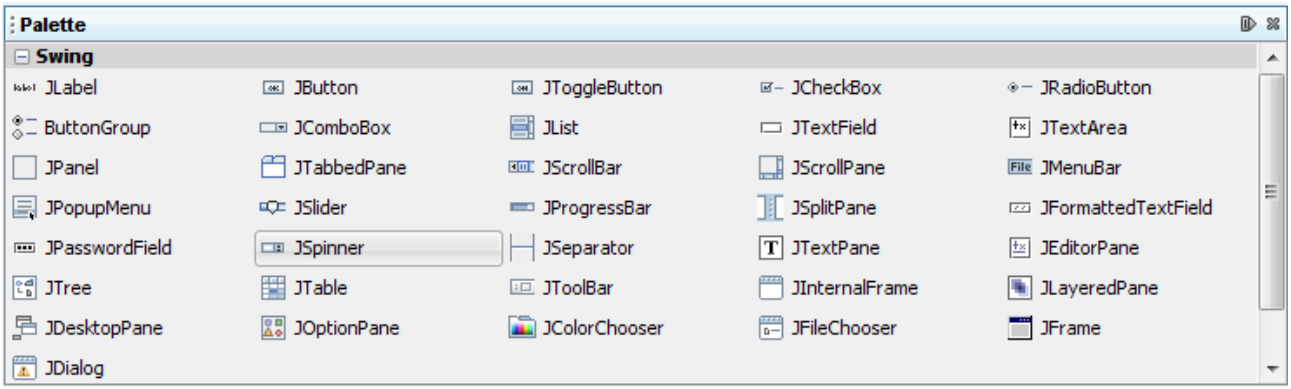
Σχήμα 19: Visual Mobile Designer

18.3 Swing

Το Netbeans περιέχει τις παρακάτω δυνατότητες ανάπτυξης μιας Java Swing εφαρμογής .

18.4 Παλέτα (Palette)

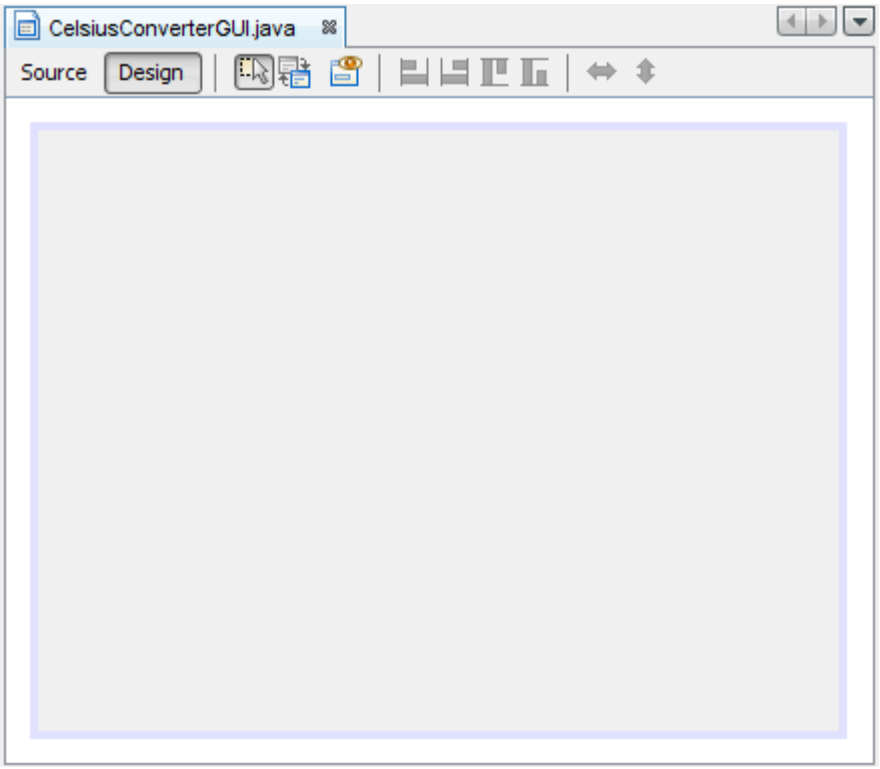
Η παλέτα περιέχει όλα τα στοιχεία που προσφέρονται από το Swing API.



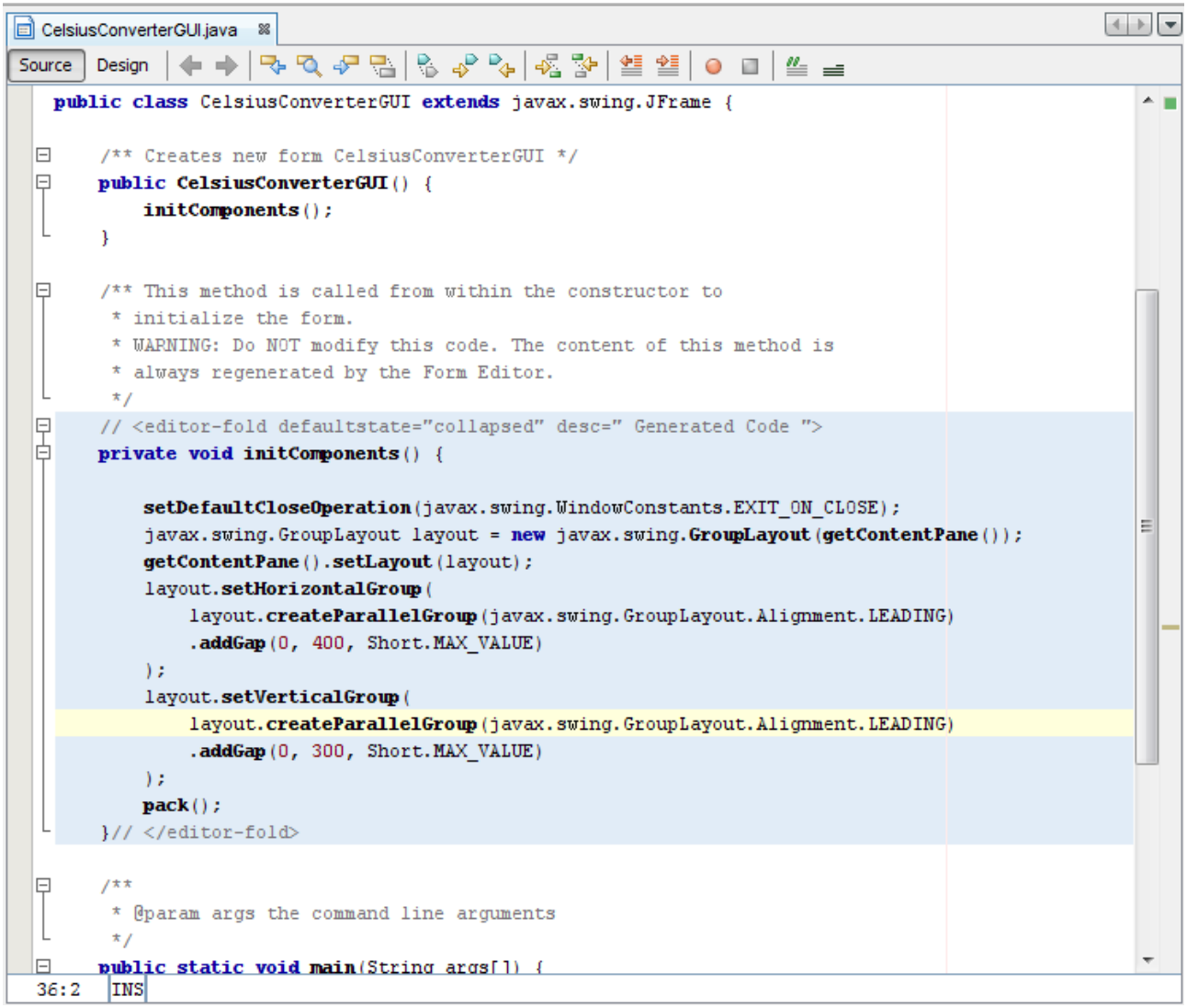
Μερικά από τα στοιχεία που είναι διαθέσιμα στην παλέτα.

18.5 Περιοχή Σχεδιασμού (Design Area)

Η Περιοχή Σχεδιασμού είναι ο χώρος που γίνεται η γραφική κατασκευή του GUI. Διαθέτει δύο όψεις: προβολή του πηγαίου κώδικα και όψη γραφικού σχεδιασμού.



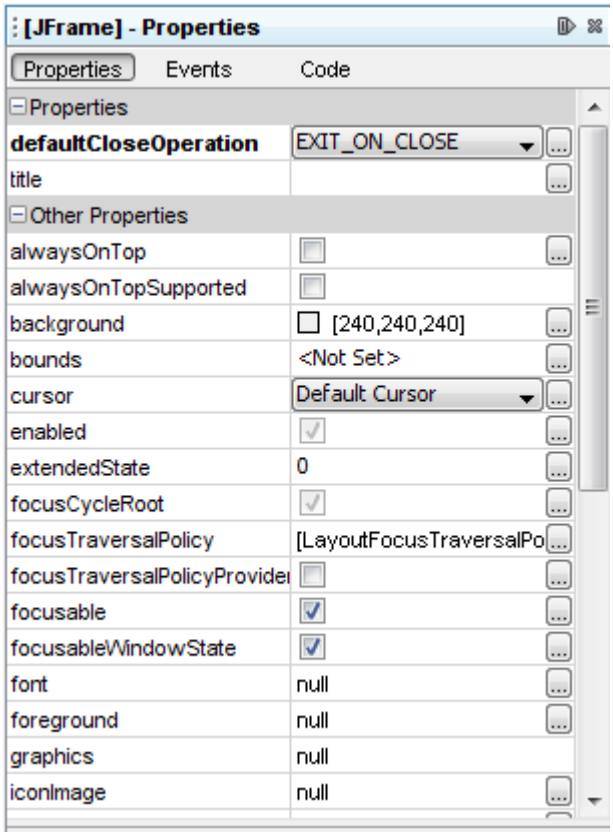
Η παραπάνω εικόνα δείχνει ένα ενιαίο `JFrame` αντικείμενο, όπως εμφανίζεται από το μεγάλο μπλε σκιασμένο ορθογώνιο περίγραμμα. Η συνήθως αναμενόμενη συμπεριφορά (όπως η έξοδος, όταν ο χρήστης κάνει κλικ στο πλήκτρο "Κλείσιμο") παράγεται αυτόματα από το IDE και εμφανίζεται στην όψη του πηγαίου κώδικα μεταξύ μη επεξεργάσιμων, μπλε τμημάτων του κώδικα γνωστά ως φυλασσόμενο μπλοκ.



Μια γρήγορη ματιά στον πηγαίο κώδικα αποκαλύπτει ότι το IDE έχει δημιουργήσει μια ιδιωτική μέθοδο που ονομάζεται `initComponents`, η οποία αρχικοποιεί τα διάφορα συστατικά στοιχεία του GUI. Λέει, επίσης, στην εφαρμογή να τερματίσει κατά το κλείσιμο “exit on close”, εκτελεί κάποια συγκεκριμένες διεργασίες σχετικές με τη διάταξη των συστατικών και πακετάρει τα συστατικά μαζί στην οθόνη.

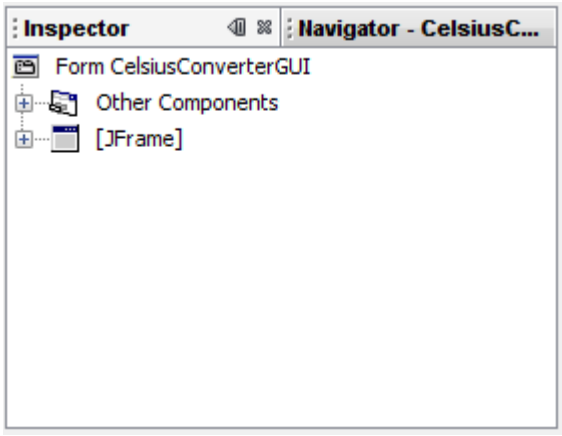
18.6 Επεξεργαστής Ιδιοκτησίας (Property Editor)

Ο Επεξεργαστής Ιδιοκτησίας επιτρέπει την επεξεργασία των ιδιοτήτων του κάθε συστατικού.



Η ανωτέρω εικόνα δείχνει τις διάφορες ιδιότητες αυτού του αντικειμένου, όπως το χρώμα φόντου, το χρώμα των νέων στοιχείων, τη γραμματοσειρά και τον δρομέα.

18.7 Επιθεωρητής (Inspector)



Ο Επιθεωρητής παρέχει μια γραφική αναπαράσταση των στοιχείων της εφαρμογής.

19. Προδιαγραφή χαμηλού επιπέδου (Low Level Specification)

19.1 Περιγραφή

Το πρόγραμμα *medicalRecord* αποτελείται από δυο πακέτα, *mobile* και *desktop*, τα οποία περιέχουν τις υλοποιήσεις για το κινητό τηλέφωνο και τον υπολογιστή αντίστοιχα. Παρακάτω θα εξετάσουμε αναλυτικά την δομή και την λειτουργία για κάθε ένα από τα παραπάνω πακέτα.

19.1.1 mobile

Η υλοποίηση αποτελείται από τέσσερα πακέτα, *connections, encryption, midlet, storage.* Παρακάτω περιγράφονται αναλυτικά τα πακέτα αυτά.

19.1.1.1 connections

Το πακέτο περιέχει τις κλάσεις που αφορούν στην διασυνδεσιμότητα του κινητού τηλεφώνου με τον υπολογιστή. Οι περιεχόμενες κλάσεις και οι κυριότερες μέθοδοί τους παραθέτονται παρακάτω.

CodeGenerator

Βοηθητική κλάση για τη δημιουργία ενός ψευδοτυχαίου 5ψήφιου αριθμού που χρησιμοποιείται κατά τη σύνδεση μέσω Bluetooth.

Η κλάση επεκτείνει την *java.util.Random.*

CodeGenerator (Constructor)

Αρχικοποιεί την κλάση και καλεί ιδιωτική μέθοδο που παράγει ένα πενταψήφιο ψευδοτυχαίο αριθμό.

getCode

Καλεί ιδιωτική μέθοδο, η οποία επιστρέφει ένα μονοψήφιο αριθμό, πέντε φορές και προσθέτει το αποτέλεσμα σε μια στοιχειοσειρά.

getInt

Ορίσματα: *int n* – Ο μέγιστος επιτρεπόμενος παραγόμενος αριθμός.

Υλοποίηση της αντίστοιχης μεθόδου της κλάσης java.util.Random του Java SDK, η οποία δεν υπάρχει στο Java ME.

OBEXServer

Η κλάση χρησιμοποιείται για τη σύνδεση του κινητού τηλεφώνου με τον υπολογιστή του ιατρού για την ανταλλαγή του ιατρικού φακέλου. Η σύνδεση γίνεται μέσω Bluetooth με τη χρήση του πρωτοκόλλου OBEX. Το κινητό τηλέφωνο λειτουργεί ως εξυπηρετητής. Η κλάση παρέχει μια μέδοθο για την έναρξη του εξυπηρετητή από το κινητό τηλέφωνο και δύο μεθόδους που καλούνται από την εφαρμογή στον ηλεκτρονικό υπολογιστή για τη λήψη και αποστολή του φακέλου.

Για την επαλήθευση της ταυτότητας του πελάτη η κλάση εφαρμόζει την διεπαφή *Authenticator*. Ο κωδικός σύνδεσης που εμφανίζεται στην οθόνη του κινητού κατά την αναμονή σύνδεσης από τον πελάτη χρησιμοποιείται για την επαλήθευση.

Παρακάτω αναφέρονται οι μέθοδοι της κλάσης και η χρήση αυτών:

OBEXServer (Constructor)

Ορίσματα: *MedicalRecordMIDlet parent* – Ο μέγιστος επιτρεπόμενος παραγόμενος αριθμός.

Αρχικοποιεί την κλάση και καλεί την *CodeGenerator* για δημιουργία κωδικού σύνδεσης.

startServer

Εκκινεί τον εξυπηρετητή OBEX στο κινητό τηλέφωνο για τη σύνδεση με τον υπολογιστή. Ο εξυπηρετητής παραμένει σε αναμονή μέχρι να γίνει σύνδεση. Μετά το κλείσιμο της πρώτης σύνδεσης ο εξυπηρετητής τερματίζει. Χρησιμοποιείται για την αποστολή και λήψη του ιατρικού φακέλου.

onConnect

Ορίσματα*:*

HeaderSet request – Περιέχει τις επικεφαλίδες που έχουν σταλεί από τον πελάτη.

HeaderSet reply – Περιέχει τις επικεφαλίδες που θα σταλούν από τον εξυπηρετητή.

Επιστρέφει*: int* – Τον κωδικό απόκρισης (Response Code) που θα απσταλεί στον πελάτη.

Αποτελεί την πρώτη μέθοδο που καλείται όταν ένας πελάτης επιχειρήσει να συνδεθεί με τον εξυπηρετητή. Στην μέθοδο αυτή ο εξυπηρετητής ζητά από τον πελάτη να εξακριβώσει την αυθεντικότητά περνώντας ένα authentication challenge στις επικεφαλίδες απάντησης. Ο πελάτης πρέπει να απαντήσει με ένα username και password τα οποία θα χρησιμοποιηθούν από τον εξυπηρετητή για την αναγνώριση του πελάτη.

OnDisconnect

Ορίσματα:

HeaderSet request – Περιέχει τις επικεφαλίδες που έχουν σταλεί από τον πελάτη.

HeaderSet reply – Περιέχει τις επικεφαλίδες που θα σταλούν από τον εξυπηρετητή.

Καλείται από τον πελάτη για την αποσύνδεση με τον εξυπηρετητή.

OnGet

Ορίσματα*: Operation op* – Περιέχει τις επικεφαλίδες που έχουν σταλεί από τον πελάτη και επιτρέπει τη δημιουργία νέων επικεφαλίδων από τον εξυπηρετητή.

Επιστρέφει*: int* – Τον κωδικό απόκρισης (Response Code) που θα αποσταλεί στον πελάτη.

Η μέθοδος διαβάζει τις επικεφαλίδες και γράφει σε ένα εξερχόμενο δεδομενορεύμα το αντικείμενο που έχει ζητηθεί από τον πελάτη. Τα δυνατά αντικείμενα είναι ο ιατρικός φάκελος του ασθενή και ο φάκελος άμεσης ανάγκης του ασθενή. Για την αναγνώριση του αντικειμένου που έχει ζητηθεί χρησιμοποιείται η επικεφαλίδα *NAME*.

OnPut

Ορίσματα*: Operation op* – Περιέχει τις επικεφαλίδες που έχουν σταλεί από τον πελάτη και επιτρέπει τη δημιουργία νέων επικεφαλίδων από τον εξυπηρετητή.

Επιστρέφει*: int* – Τον κωδικό απόκρισης (Response Code) που θα αποσταλεί στον πελάτη.

Η μέθοδος διαβάζει τις επικεφαλίδες και διαβάζει από ένα εισερχόμενο δεδομενορεύμα το αντικείμενο που θα αποσταλεί από τον πελάτη. Τα δυνατά αντικείμενα είναι ο ιατρικός φάκελος του ασθενή και ο φάκελος άμεσης ανάγκης του ασθενή. Για την αναγνώριση του αντικειμένου χρησιμοποιείται η επικεφαλίδα *NAME*.

19.1.1.2 encryption

Το πακέτο περιέχει τις κλάσεις που αφορούν στην κρυπτογράφηση του ηλεκτρονικού ιατρικού φακέλου.

Οι περιεχόμενες κλάσεις και οι κυριότερες μέθοδοί τους παραθέτονται παρακάτω.

MyCipher

Είναι υπεύθυνη για την κρυπτογράφηση και αποκρυπτογράφηση των στοιχείων του ιατρικού φακέλου. Για την κρυπτογράφηση χρησιμοποιείται ο αλγόριθμος AES.

MyCipher (Constructor):

Ορίσματα*: String password* – Ο 4ψήφιος αριθμός αναγνωριστικό του χρήστη.

Αρχικοποιεί την κλάση και δημιουργεί το 16σύλλαβο κλειδί που θα χρησιμοποιηθεί για την κρυπτογράφηση. Επίσης αρχικοποιεί το κρυπτογράφημα για τον αλγόριθμο AES.

encrypt

Ορίσματα*: byte[] plainText* – Ο ψηφιακός ιατρικός φάκελος.

Επιστρέφει: *byte[] – Tov* ιατρικό φάκελο κρυπτογραφημένο.

Θέτει τη λειτουργία του κρυπτογραφήματος σε ENCRYPT και καλεί ιδιωτική μέθοδο της που επιστρέφει τον ιατρικό φάκελο κρυπτογραφημένο.

decrypt

Ορίσματα: *byte[] encryptedText* – Ο ψηφιακός ιατρικός φάκελος κρυπτογραφημένος.

Επιστρέφει: *byte[]* – Τον ιατρικό φάκελο κρυπτογραφημένο.

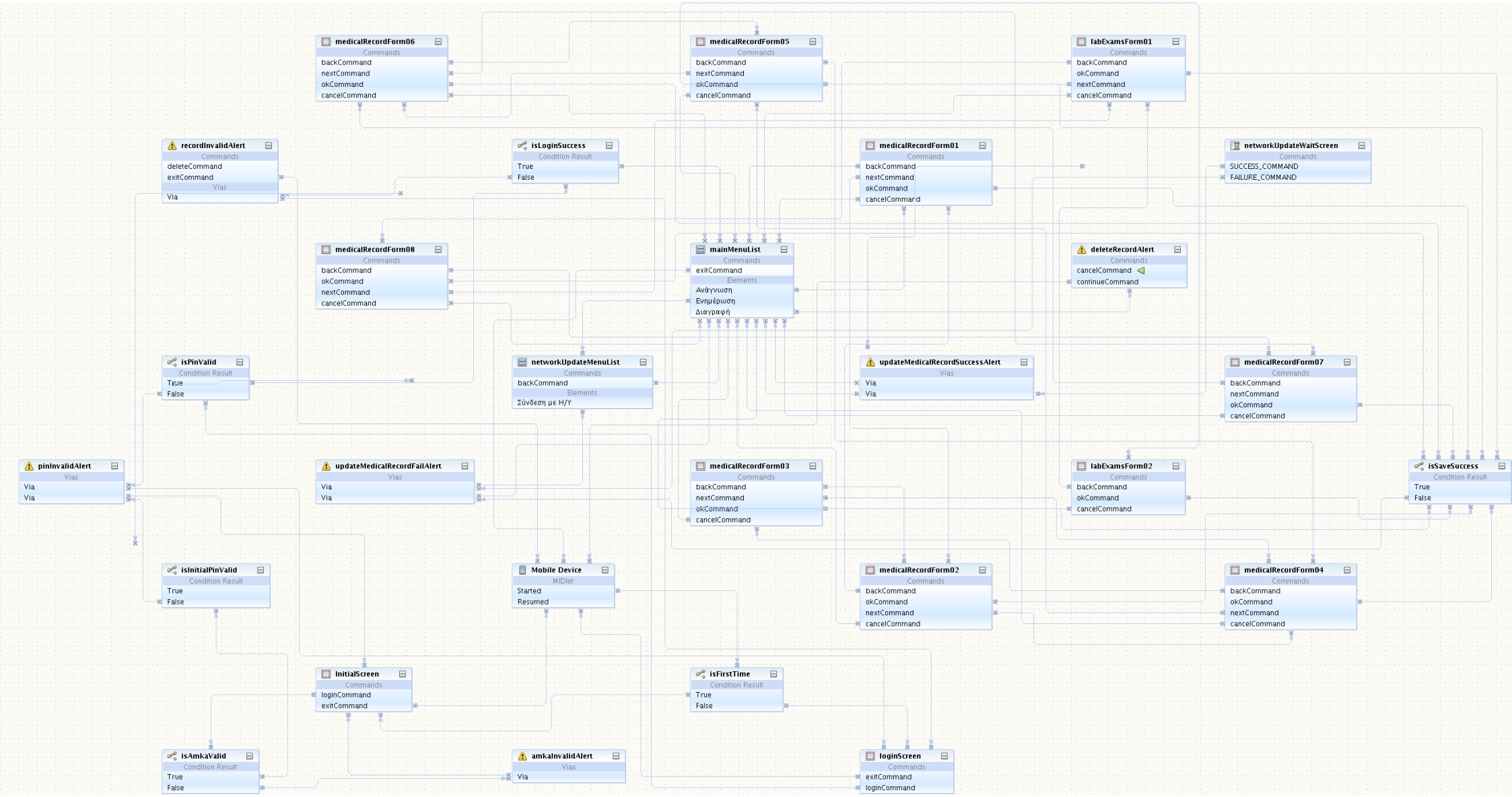
Θέτει τη λειτουργία του κρυπτογραφήματος σε ENCRYPT και καλεί ιδιωτική μέθοδο της κλάσης που επιστρέφει τον ιατρικό φάκελο κρυπτογραφημένο.

19.1.1.3 midlet

Το πακέτο περιέχει τις κλάσεις που αφορούν στο γραφικό περιβάλλον της εφαρμογής. Οι περιεχόμενες κλάσεις και οι κυριότερες μέθοδοί τους παραθέτονται παρακάτω.

MedicalRecordMIDlet

Αποτελέι το γραφικό περιβάλλον της εφαρμογής και το σύνδεσμο με όλες τις υπόλοιπες κλάσεις. Η ροή της εφαρμογής συνοψίζεται στο σχήμα που ακολουθεί.


medicalRecordForm06
Commands
backCommand
nextCommand
okCommand
cancelCommand
medicalRecordForm05
Commands
backCommand
nextCommand
okCommand
cancelCommand
labExamsForm01
Commands
backCommand
okCommand
nextCommand
cancelCommand
recordInvalidAlert
Commands
deleteCommand
exitCommand
Vias
Via
isLoginSuccess
Condition Result
True
False
medicalRecordForm01
Commands
backCommand
nextCommand
okCommand
cancelCommand
networkUpdateWaitScreen
Commands
SUCCESS_COMMAND
FAILURE_COMMAND
medicalRecordForm08
Commands
backCommand
okCommand
nextCommand
cancelCommand
mainMenuList
Commands
exitCommand
Elements
Ανάγνωση
Ενημέρωση
Διαγραφή
deleteRecordAlert
Commands
cancelCommand
continueCommand
isPinValid
Condition Result
True
False
networkUpdateMenuList
Commands
backCommand
Elements
Σύνδεση με Η/Υ
updateMedicalRecordSuccessAlert
Vias
Via
Via
medicalRecordForm07
Commands
backCommand
nextCommand
okCommand
cancelCommand
pinInvalidAlert
Vias
Via
Via
updateMedicalRecordFailAlert
Vias
Via
Via
medicalRecordForm03
Commands
backCommand
nextCommand
okCommand
cancelCommand
labExamsForm02
Commands
backCommand
okCommand
cancelCommand
isSaveSuccess
Condition Result
True
False
isInitialPinValid
Condition Result
True
False
Mobile Device
MIDlet
Started
Resumed
medicalRecordForm02
Commands
backCommand
okCommand
nextCommand
cancelCommand
medicalRecordForm04
Commands
backCommand
okCommand
nextCommand
cancelCommand
InitialScreen
Commands
loginCommand
exitCommand
isFirstTime
Condition Result
True
False
isAmkaValid
Condition Result
True
False
amkaInvalidAlert
Vias
Via
loginScreen
Commands
exitCommand
loginCommand

19.1.1.4 storage

Το πακέτο περιέχει τις κλάσεις που αφορούν στην αποθήκευση των δεδομένων του ιατρικού φακέλου. Οι περιεχόμενες κλάσεις και οι κυριότερες μέθοδοί τους παραθέτονται παρακάτω.

GenericRecord

Η κλάση αυτή ορίζει τη γενική δομή ενός αντικειμένου που μπορεί να αποηθκεύεται από την εφαρμογή. Παρέχει μεδόθους για την ευκολότερη αποθήκευση και ανάσυρσει των στοιχείων που είναι αποθηκευμένα σε *RecordStore*. Όλα τα δεδομένα του αντικειμένου αποθηκεύονται υπό δομή *Hashtable* για εύκολα επεκτάσιμα.

GenericRecord(Constructor)

Η μέδοθος αυτή αρχικοποιεί την κλάση χωρίς δεδομένα.

GenericRecord(Constructor)

Ορίσματα: *Hashtable dataTable* – Τα δεδομένα προς αποθήκευση.

Η μέδοθος αρχικοποιεί την κλάση με κάποια δεδομένα και καλεί ιδιωτική μέθοδο για να μεταφράσει σε κενές στοιχειοσειρές τυχόν *null* τιμές δεδομένων. Αυτό γίνεται για να μην έχουμε *NullPointerException* όταν ανακαλούνται τα δεδομένα για εμφάνιση από την εφαρμογή ή όταν αποθηκεύονται σε ένα *RecordStore*.

getDateL

Επιστρέφει την ημερομηνία πρώτης αποθήκευσης του αντικειμένου στο RecordStore.

setDateL

Ορίσματα: *long dateL* – Η ημερομηνία της δημιουργίας του ιατρικού φακέλου.

Καλείται εσωτερικά από την εφαρμογή για να ορίσει την ημερομηνία της δημιουργίας του ιατρικού φακέλου. Στον παρόν στάδιο η πληροφορία αυτή είναι διάφανη στον χρήστη της εφαρμογής.

getRecordId

Επιστρέφει: *int* – το RecordId του αντικειμένου.

Χρησιμοποιείται για να βρεθεί η θέση στην οποία το αντικείμενο ήταν αποθηκευμένο στο RecordStore.

setRecordId

Ορίσματα: *int RecordId* – Το RecordId του αντικειμένου.

Χρησιμοποιείται για να ορίσει τη θέση του αντικειμένου στο RecordStore.

getDataTable

Επιστρέφει: *Hashtable* – Τα δεδομένα του αντικειμένου.

Χρησιμοποιείται για την ανάσυρση όλων των στοιχείων του ιατρικού φακέλου είτε για εμφάνιση από την εφαρμογή είτε για την αποθήκευση τους σε *RecordStore*.

setDataTable

Ορίσματα: *Hashtable dataTable* – Τα δεδομένα του αντικειμένου.

Χρησιμοποιείται για την αρχικοποίηση/τροποποίηση των δεδομένων του ιατρικού φακελου σε περίπτωση αλλαγής τους από τον χρήστη καθώς και κατά την περίπτωση ανάσυρσης τους από *RecordStore*.

toByteArray

Επιστρέφει: *byte[]* – Τα δεδομένα του αντικειμένου υπό δομή array.

Για να αποθηκευτούν τα στοιχεία του ιατρικού φακέλου σε *RecordStore* πρέπει να βρίσκονται σε αυτή τη δομή. Αυτή η μέθοδος αυτοματοποιεί την μετατροπή των δεδομένων του ιατρικού φακέλου από *Hashtable* σε *bye[].*

fromByteArray

Όταν ανασύρονται τα στοιχεία του ιατρικού φακέλου από ένα *RecordStore* βρίσκονται σε δομή *byte[]*. Η μέδοθος αυτή τα μετατρέπει σε δομή *Hashtable* για να είναι πιο διαχειρίσιμα.

initializeHashTable

Ιδιωτική μέδοθος. Διαβάζει όλα τα δεδομένα του ιατρικού φακέλου. Αν κάποιο από αυτά είναι *null* τότε το αρχικοποιεί σαν κενή γραμματοσειρά.

MedicalRecord

Η κλάση αυτή κάνει επέκταση της κλάση *GenericRecord*.

Περιέχει τα όλα τα πεδία του ιατρικού φακέλου εκτός των εργαστηριακών εξετάσεων. Το αντικείμενο αυτό χρησιμοποίειται για να κρατάει τις πληροφορίες του ιατρικού φακέλου ενόσω τρέχει η εφαρμογή.

Ορίζει μεθόδους *getters* και *setters* για τα όλα τα πεδία του ιατρικού φακέλου.

MedicalRecord(Constructor)

Ορίσματα*: Hashtable dataTable* – Τα δεδομένα του αντικειμένου*.*

Χρησιμοποιεί το constructor της κλάσης *GenericRecord* για την αρχικοποίηση της κλάσης. Η κλάση αρχικοποιείται με τις τιμές των στοίχειων του ιατρικού φακέλου που έχουν ανακτηθεί από το *RecordStore*. Σε περίπτωση δημιουργίας νέου φακέλου αρχικοποιείται με την τιμή του ΑΜΚΑ μόνο.

Στην κλάση αυτή υπάρχουν και μια σειρά από getters και setters για την ευκολότερη ανάκτηση των στοιχείων του αντικειμένου. Κάθε πεδίο, όπως αυτά εμφανίζονται κατά την ανάγνωση του ιατρικού φακέλου, έχει ένα *setter* και ένα *getter*.

setPatientAttrs

Ορίσματα:

String surname – Το επίθετο του ασθενή

String name – Το όνομα του ασθενή

String blo odTypeRh – Η ομάδα αίματος του ασθενή

String address – Η διεύθυνση του ασθενή

String postalCode – Ο ταχυδρομικός κώδικας του ασθενή

String city – Η πόλη του ασθενή

String phoneNumber – Το σταθερό τηλέφωνο του ασθενή

String nearestRelativeSurname – Το επίθετου του πλησιέστερου συγγενή

String nearestRelativeName – Το όνομα του πλησιέστερου συγγενή

String nearestRelativePhoneNumber –Το τηλέφωνο του πλησιέστερου συγγενή

String nearestRelativeRelationship – Η συγγένια με τον πλησιέστερου συγγενή

Ο χρήστης μπορεί να τροποποιήσει τα στοιχεία του φακέλου και να αποθηκεύσει τις αλλαγές. Η μέθοδος αυτή χρησιμοποιείται για να ενημερώσει το αντικείμενο με τις αλλαγές στα στοιχεία του ασθενή/συγγενή που έχουν γίνει.

RMS

Η κλάση αυτή είναι υπεύθυνη για την οποιαδήποτε επικοινωνία της εφαρμογής με το *RecordStore.* Το RecordStore είναι το αντικείμενο στο οποίο αποθηκεύεται ο ιατρικός φάκελος. Τα στοιχεία του φακέλου αποθηκεύονται από ένα *DataOutputStream* και ανακτώνται με τη χρήση ενός *DataInputStream.* Υπάρχει μόνο ένα *RecordStore* για την αποθήκευση στοιχείων του ιατρικού φακέλου.

Όταν δημιουργείται μια εγγραφή σε ένα *RecordStore* επιστρέφεται ένα *recordId* το οποίο αποτελεί το πρωτεύων κλειδί του αντικειμένου. Για την ανάκτηση του ιατρικού φακέλου που είναι αποθηκευμένος στο *RecordStore* χρειάζεται το *record.* Το ίδιο ισχύει και κατά την ενημέρωση μιας υπάρχουσας εγγραφής. Το *RecordStore* υποστηρίζει την ανάκτηση εγγραφών με κάποιο φίλτρο, αλλά επειδή τα δεδομένα είναι κρυπτογραφημένα η χρησιμοποίηση αυτής της μεθόδου δεν είναι δυνατή. Κατά συνέπεια η υλοποίηση δεν υποστηρίζει περισσότερους από έναν ιατρικό φάκελο.

RMS (Constructor)

Ορίσματα: *String password* – Το αναγνωριστικό του χρήστη.

Όταν αρχικοποιείται η κλάση, περνιέται το αναγνωριστικό του χρήστη για να μπορεί να αποκρυπτογραφηθεί/κρυπτογραφηθεί ο ιατρικός φάκελος.

setPassword(Costructor)

Ορίσματα: *String password* – Το αναγνωριστικό του χρήστη.

Θέτει την τιμή του αναγνωριστικού μετά τη αρχικοποίηση της κλάσης.

getRecordIds

Ορίσματα: *String recordDataType* – Ο τύπος των περιεχομένων του *RecordStore*.

Επιστρέφει: *int[]* – Συστοιχεία με τα *recordIds* του συγκεκριμένου *RecordStore*.

Χρησιμοποιείται για να επιστρέψει το *recordId* της εγγραφής του ιατρικού φακέλου. Το *recordId* χρησιμοποιείται μετέπειτα για την ανάκτηση αποθήκευση του ιατρικού φακέλου.

Αν δε βρεθεί κανένα *recordId* τότε σημαίνει ότι είναι η πρώτη χρήση της εφαρμογής και δίνεται η επιλογή στον χρήστη να δημιουργήσει καινούργιο ιατρικό φάκελο.

Αν βρεθούν περισσότερα από ένα *recordIds* τότε έχει προκύψει κάποιο σφάλμα με την εφαρμογή.

getRecord

Ορίσματα: *int recordId* – Το recordid της εγγραφής στο *RecordStore*.

Επιστρέφει: *byte[]* – το περιεχόμενο της εγγραφής.

Χρησιμοποιείται για την ανάκτηση του ιατρικού φακέλου. Έχει προηγηθεί η κλήση της *getRecordIds* για την ανάκτηση του *recordId* του ιατρικού φακέλου.

addRecord

Ορίσματα:

String RecordDataType – Ο τύπος το δεδομένων του *RecordStore*.

byte[] recordDataBs – Τα δεδομένα προς αποθήκευση.

Επιστρέφει: int – Το recordId της εγγραφής στο *RecordStore*.

Αποθηκεύει τα δεδομένα του ιατρικού φακέλου στο *RecordStore*. Πριν την αποθήκευση καλεί τη μέθοδο *callEncrypt* για καρυπτογράγηση των δεδομένων.

updateRecord

Ορίσματα:

int recordId – Το recordid της εγγραφής στο *RecordStore*.

byte[] recordDataBs – το περιεχόμενο της εγγραφής.

Χρησιμοποιείται για την αποθήκευση της ενημερωμένης έκδοσης του ιατρικού φακέλου.

deleteRecord

Ορίσματα: *String RecordDataType* – Ο τύπος το δεδομένων του *RecordStore*.

Διαγράφει την εγγραφή του ιατρικού φακέλου.

closeRecordStores

Κλείνει τα *RecordStores* μετά τη χρήση. Είναι υποχρεωτικό το κλείσιμο πριν τον τερματισμό της εφαρμογής.

deleteRecordStore

Διαγράφει το *RecordStore*, δηλαδή το αρχείο αποθήκευσης της βάσης δεδομένων.

CallEncrypt

Ορίσματα: *byte[] bs* – Τα δεδομένα του ιατρικού φακέλου.

Επιστρέφει: *byte[]* – κρυπτογραφημένα τα δεδομένα του ιατρικού φακέλου.

Ιδιωτική μέθοδος για την κρυπτογράφηση των δεδομένων του ιατρικού φακέλου πριν την αποθήκευση τους σε εγγραφή του *RecordStore*.

CallDecrypt

Ορίσματα: *byte[] bs* – Τα κρυπτογραφημένα δεδομένα του ιατρικού φακέλου.

Επιστρέφει: *byte[]* – τα δεδομένα του ιατρικού φακέλου αποκρυπτογραφημένα.

Ιδιωτική μέθοδος για την αποκρυπτογράφηση των δεδομένων του ιατρικού φακέλου μετά την ανάκτηση τους από εγγραφή του *RecordStore*.

openRecordStore

Επιστρέφει: *RecordStore* – Αντικείμενο για την χρήση του *RecordStore*.

Για να χρησιμοποιηθεί μια βάση δεδομένων *RecordStore* πρέπει πρώτα να ανοιχθεί. Αυτό γίνεται με τη χρήση αυτής της ιδιωτικής μεθόδου.

19.1.2 desktop

Το πακέτο αυτό περιέχει τις κλάσεις της υλοποίησης στον ηλεκτρονικό υπολογιστή. Χρησιμοποιείται κατά την επίσκεψη του ασθενή στον ιατρό για την ανάγνωση και ενημέρωση του ιατρικού φακέλου.

Παρακάτω περιγράφονται αναλυτικά τα πακέτα αυτά.

19.1.3 desktopmedicalrecord

Το πακέτο αυτό περιέχει τις κλάσεις της γραφικής διεπαφής του χρήστη (GUI).

DesktopMedicalRecord

Η κύρια κλάση της εφαρμογής.

Αρχικοποιεί την κλάση **DesktopMedicalRecordView** που αποτελεί το κυρίως πλαίσιο της εφαρμογής.

Περιέχει μέθοδο *getter* για την εύκολη πρόσβαση στο κυρίως πλαίσιο.

DesktopMedicalRecordAboutBox

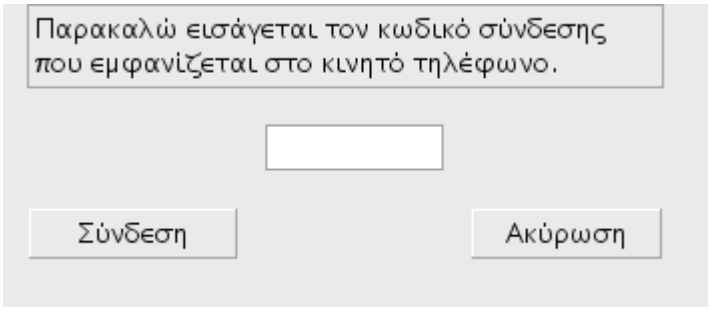
Κλάση που υλοποιεί το παράθυρο με γενικές για την εφαρμογή.

Η εφαρμογή χρησιμοποιείται σε συνδυασμό με αντίστοιχη εφαρμογή στο κινητό τηλέφωνο για την ανάγνωση και τροποποίηση ιατρικού φακέλου.

Σχήμα 20: Εφαρμογή Η/Υ: Πέρι

DesktopMedicalRecordConnecitonDialog

Κλάση που υλοποιεί το παράθυρο εισαγωγής του κωδικού για την σύνδεση με το κινητό τηλέφωνο. Καλείται κατά τη λήψη του ιατρικού φακέλου. Ο κωδικός σύνδεσης αποθηκεύεται και επαναχρησιμοποιείται κατά την αποστολή του ιατρικού φακέλου.



Σχήμα 21: Εφαρμογή Η/Υ: Σύνδεση

DesktopMedicalRecordView

Το κυρίως πλαίσιο της εφαρμογής. Περιέχει καρτέλες με τα στοιχεία του ιατρικού φακέλου και τα στοιχεία άμεσης ανάγκης.

Για την καταχώρηση των αλλαγών στα στοιχεία του ιατρικού φακέλου χρησιμοποιείται το πλήκτρο “Καταχώρηση”.

Για την ακύρωση των τροποποιήσεων του ιατρικού φακέλου και την επαναφορά των δεδομένων όπως αυτά έχουν ληφθεί από το κινητό τηλέφωνο χρησιμοποιείται το πλήκτρο “Ακύρωση”.

Η σύνδεση με το κινητό τηλέφωνο τρέχει σε μίτο (thread) για να μην παγώσει η λειτουργία της εφαρμογής.

19.1.4 obex

Περιέχει τις κλάσεις για την σύνδεση μέσω Bluetooth με το κινητό τηλέφωνο. Συγκεκριμένα περιέχει την κλάση ανεύρεσης Bluetooth συσκευών και αναζήτησης διαθέσιμων υπηρεσιών και την κλάση πελάτη OBEX για την λήψη/αποστολή του ιατρικού φακέλου από το κινητό τηλέφωνο.

OBEXClient

Η κλάση περιέχει μεθόδους λήψης (*get*) και αποστολής (*put*) του ιατρικού φακέλου. Επίσης περιέχει μέθοδο σύνδεσης με το κινητό τηλέφωνο. Τέλος υλοποιεί την κλάση Authenticator για την επαλήθευση, με τον κωδικό σύνδεσης, της ταυτότητας του ηλεκτρονικού υπολογιστή κατά την σύνδεση.

get

Ορίσματα:

byte[] username – Δεν χρησιμοποιείται.

byte[] password – Ο κωδικός σύνδεσης με το κινητό τηλέφωνο.

String objName – Το όνομα του αντικειμένου προς λήψη.

Επιστρέφει: *byte[]* – Το αντικείμενο που έχει ζητηθεί.

Η μέθοδος επιστρέφει τον ιατρικό φάκελο από το κινητό τηλέφωνο.

Η παράμετρος *ObjName* ορίζει το όνομα του αντικειμένου που ζητείται. Μπορεί να είναι είτε ο ιατρικός φάκελος του ασθενή είτε τα στοιχεία άμεσης ανάγκης.

Ορίζει τις επικεφαλίδες και ανοίγει ένα εισερχόμενο δεδομενορεύμα για την λήψη του αντικειμένου.

Μετά τη λήψη αποσυνδέεται από τον εξυπηρετητή.

put

Ορίσματα:

byte[] username – Δεν χρησιμοποιείται.

byte[] password – Ο κωδικός σύνδεσης με το κινητό τηλέφωνο.

byte[] obj – Το αντικείμενο που θα αποσταλεί στον εξυπηρετητή.

String objName – Το όνομα του αντικειμένου προς λήψη.

Η μέθοδος αποστέλλει τον ιατρικό φάκελο στο κινητό τηλέφωνο.

Η παράμετρος *ObjName* ορίζει το όνομα του αντικειμένου που αποστέλλεται. Μπορεί να είναι είτε ο ιατρικός φάκελος του ασθενή είτε τα στοιχεία άμεσης ανάγκης.

Ορίζει τις επικεφαλίδες και ανοίγει ένα εξερχόμενο δεδομενορεύμα για την αποστολή του αντικειμένου. Στις επικεφαλίδες πέραν του ονόματος του αντικειμένου ορίζει και το μέγεθος αυτού.

Μετά τη αποστολή αποσυνδέεται από τον εξυπηρετητή.

connectToObexServer

Μέθοδος που αρχικοποιεί την σύνδεση με το κινητό τηλέφωνο.

Αρχικά καλεί την κλάση *ServicesSearch* για την αναζήτηση του κινητού τηλεφώνου και ορίζει τις γενικές επικεφαλίδες σύνδεσης, ανεξάρτητα του αν η σύνδεση χρησιμοποιηθεί για την λήψη ή την αποστολή αντικειμένου. Οι επικεφαλίδες περιέχουν και τον κωδικό σύνδεσης.

ServicesSearch

Η κλάση ανεύρεσης Bluetooth συσκευών και αναζήτησης της υπηρεσίας ιατρικού φακέλου. Η συσκευές που αποκαλυφθέντες συσκευές προστίθενται σε ένα διάνυσμα. Οι υπηρεσίες που ανευρίσκονται προστίθενται σε ένα διάνυσμα.

Η κλάση υλοποιεί την διεπαφή *DiscoveryListener* για την αναζήτηση Bluetooth συσκευών.

main

Ορίσματα: *String[] args* – Δεν χρησιμοποιείται.

Η μέθοδος ανεύρεσης Bluetooth συσκευών και αναζήτησης Bluetooth υπηρεσιών.

Οι συσκευές που ανακαλύπτονται προστίθενται σε ένα διάνυσμα *Vector*. Για να προστεθεί όμως μια συσκευή πρέπει να έχει μείζονα κλάση συσκευής (Major Device Class) ίσο με 0x0200. Αυτό επιτυγχάνει την προσθήκη στο διάνυσμα μόνο των συσκευών κινητής τηλεφωνίας.

Οι υπηρεσίες που ανακαλύπτονται προστίθενται σε ένα διάνυσμα *Vector*. Για να προστεθεί όμως μια συσκευή πρέπει να έχει μοναδικό καθολικό αναγνωριστικό (UUID) 0x7105 και όνομα υπηρεσίας MedicalRecord. Αυτό επιτυγχάνει την προσθήκη στο διάνυσμα μόνο των υπηρεσιών ιατρικού φακέλου μόνο.

deviceDiscovered

RemoteDevice btDevice – Η συσκευή που έχει ανακαλυφθεί κατά την διερεύνηση.

DeviceClass cod – Οι κλάσεις υπηρεσιών, η μείζονα κλάση και η ελάσσονα κλάση της συσκευής που έχει ανακαλυφθεί.

Καλείται όταν ανεβρεθεί μια συσκευή κατά τη διάρκεια της διερεύνησης. Η διερεύνηση ψάχνει μόνο για συσκευές που είναι ανακαλύψιμες.

InquiryCompleted

int discType – Το είδος της διερεύνησης που έχει ολοκληρωθεί.

Καλείται όταν η διερεύνηση ολοκληρωθεί. Η διερεύνηση ολοκληρώνεται είτε με επιτυχία είτε με τερματισμό είτε με σφάλμα.

ServicesDiscovered

int transID – Δεν χρησιμοποιείται.

ServiceRecord[] servRecord – Λίστα με τις ανευρεθείσες υπηρεσίες.

Καλείται όταν βρεθούν υπηρεσίες κατά την αναζήτηση.

Βρόχος ψαξίματος της υπηρεσίας “MedicalRecord” στη λίστα με τις υπηρεσίες που έχουν ανευρεθεί και προσθήκη αυτής στο διάνυσμα ανευρισκομένων υπηρεσιών.

serviceSearchCompleted

int transID – Δεν χρησιμοποιείται.

int respCode – Ο κωδικός απόκρισης που επιδεικνύει την κατάσταση ολοκλήρωσης της αναζήτησης.

Η αναζήτηση μπορεί να ολοκληρωθεί με τις εξής καταστάσεις: ολοκληρωμένη, μη ανευρεθείσα συσκευή, σφάλμα, καμία εγγραφή και τερματισμένη.

Μόνο όταν η κατάσταση είναι “ολοκλήρωσης” η αναζήτηση έγινε με επιτυχία.

19.2 Δοκιμαστικά σενάρια

19.2.1 Κινητό τηλέφωνο

19.2.1.1 Δημιουργία νέου φακέλου με χρήση μη αποδεκτού ΑΜΚΑ

Βήματα:

1. Εισαγωγή λανθασμένου ΑΜΚΑ.
2. Εισαγωγή 4ψήφιου κωδικού.
3. Είσοδος.

Αναμενόμενο αποτέλεσμα:

Μήνυμα λάθους.

Δοκιμή Α:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **04790** στο πεδίο ΑΜΚΑ.
3. Εισαγωγή **1234** στο πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού.
5. Μήνυμα “**Μη αποδεκτός ΑΜΚΑ**”.

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

Δοκιμή Β:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Κενό πεδίο ΑΜΚΑ.
3. Εισαγωγή **1234** στο πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού.
5. Μήνυμα “**Μη αποδεκτός ΑΜΚΑ**”.

Αποτέλεσμα**:**

ΕΠΙΤΥΧΙΑ

19.2.1.2 Δημιουργία νέου φακέλου με χρήση μη αποδεκτού κωδικού

Βήματα:

1. Εισαγωγή ΑΜΚΑ.
2. Εισαγωγή μη 4ψήφιου κωδικού.
3. Είσοδος.

Αναμενόμενο αποτέλεσμα:

Μήνυμα λάθους.

Δοκιμή Α:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **12058104790** στο πεδίο ΑΜΚΑ.
3. Εισαγωγή **123** στο πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού.
5. Μήνυμα “**Μη αποδεκτός κωδικός**”.

Αποτέλεσμα**:**

ΕΠΙΤΥΧΙΑ

Δοκιμή Β:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **12058104790** στο πεδίο ΑΜΚΑ.
3. Κενό πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού.
5. Μήνυμα “**Μη αποδεκτός κωδικός**”.

Αποτέλεσμα**:**

ΕΠΙΤΥΧΙΑ

19.2.1.3 Δημιουργία νέου φακέλου

Βήματα:

1. Εισαγωγή ΑΜΚΑ.
2. Εισαγωγή 4ψήφιου κωδικού.
3. Είσοδος.

Αναμενόμενο αποτέλεσμα:

Επιτυχής είσοδος στην εφαρμογή.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **12058104790** στο πεδίο ΑΜΚΑ.
3. Εισαγωγή **1234** στο πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

19.2.1.4 Αλλαγή δημογραφικών στοιχείων ασθενούς και συγγενούς προσώπου

Βήματα:

1. Εισαγωγή κωδικού.
2. Επιλογή “Ανάγνωση” από το κυρίως μενού.
3. Αλλαγή στοιχείων ασθενούς.
4. Μεταφορά στην επόμενη οθόνη.
5. Αλλαγή στοιχείων συγγενούς προσώπου.
6. Αποθήκευση ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Επιτυχής ενημέρωση ιατρικού φακέλου.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).

2. Εισαγωγή **1234** στο πεδίο κωδικός.
3. Επιλογή “**Είσοδος**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).
4. Επιλογή “**Ανάγνωση**” από τη λίστα (Μεταφορά στην οθόνη “**Στοιχειά Ασθενή**”).
5. Μετακίνηση στο πεδίο “**Επίθετο**”.
6. Εισαγωγή “**ΤΕΜΠΡΙΩΤΗΣ**”.
7. Επιλογή “**Επόμενο**” από το μενού (Μεταφορά στην οθόνη “**Στοιχειά Πλησιέστερου Συγγενή**”).
8. Μετακίνηση στο πεδίο “**Όνομα**”.
9. Εισαγωγή “**ΝΙΚΟΛΑΟΣ**”.
10. Επιλογή “**Αποθήκευση**” από το μενού.
11. Μήνυμα “**Ο φάκελος ενημερώθηκε**”.
12. Επιλογή “**ΟΚ**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).
13. Επιλογή “**Ανάγνωση**” από τη λίστα (Μεταφορά στην οθόνη “**Στοιχειά Ασθενή**”).
14. Μετακίνηση στο πεδίο “**Επίθετο**” – *Το πεδίο περιέχει την τιμή **ΤΕΜΠΡΙΩΤΗΣ***.
15. Επιλογή “**Επόμενο**” από το μενού (Μεταφορά στην οθόνη “**Στοιχειά Πλησιέστερου Συγγενή**”).
16. Μετακίνηση στο πεδίο “**Όνομα**” – *Το πεδίο περιέχει την τιμή **ΝΙΚΟΛΑΟΣ***.

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

19.2.1.5 Αλλαγή ιατρικών στοιχείων φακέλου (αποτυχία)

Βήματα:

1. Εισαγωγή κωδικού.
2. Επιλογή “Ανάγνωση” από το κυρίως μενού.
3. Μεταφορά στην οθόνη με την πορεία της νόσου του ασθενούς.
4. Αλλαγή περιεχομένου του πεδίου πεδία νόσου.

Αναμενόμενο αποτέλεσμα:

Αδυναμία αλλαγής περιεχομένου. Το πεδίο μπορεί να διαβαστεί μόνο.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **1234** στο πεδίο κωδικός.
3. Επιλογή “**Είσοδος**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).

4. Επιλογή “**Ανάγνωση**” από τη λίστα (Μεταφορά στην οθόνη “**Στοιχεία Ασθενή**”).
5. Επιλογή “**Επόμενο**” από το μενού (Μεταφορά στην οθόνη “**Στοιχεία Πλησιέστερου Συγγενή**”).
6. Εισαγωγή “**Όλα καλώς**” στο πεδίο “**Πορεία Νόσου**” – *Αδυναμία εισαγωγής κειμένου*.

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

19.2.1.6 Σύνδεση με υπολογιστή με χρήση λανθασμένου ΑΜΚΑ (αποτυχία)

Βήματα:

1. Εισαγωγή κωδικού.
2. (Εκκίνηση εφαρμογής στον υπολογιστή και σύνδεση με Bluetooth).
3. Επιλογή “Ενημέρωση” από το κυρίως μενού.
4. Επιλογή “Bluetooth” από το μενού ενημέρωσης.
5. Αναμονή.
6. (Εισαγωγή λανθασμένου ΑΜΚΑ από την εφαρμογή στον υπολογιστή).

Αναμενόμενο αποτέλεσμα:

Μήνυμα Λάθους.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **1234** στο πεδίο κωδικός.
3. Επιλογή “**Είσοδος**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).
4. Επιλογή “**Ενημέρωση**” από τη λίστα (Μεταφορά στην οθόνη “**Ενημέρωση**”).
5. Επιλογή “**Bluetooth**” από τη λίστα (Μεταφορά σε οθόνη αναζήτησης).
6. Μήνυμα “**Λάθος ΑΜΚΑ!**”.

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

19.2.1.7 Σύνδεση με υπολογιστή και αποστολή/λήψη ιατρικού φακέλου

Βήματα:

1. Εισαγωγή κωδικού.
2. (Εκκίνηση εφαρμογής στον υπολογιστή και σύνδεση με Bluetooth).
3. Επιλογή “Ενημέρωση” από το κυρίως μενού.
4. Επιλογή “Bluetooth” από το μενού ενημέρωσης.
5. Αναμονή.

6. (Εισαγωγή ΑΜΚΑ από την εφαρμογή στον υπολογιστή).

Αναμενόμενο αποτέλεσμα:

Μήνυμα επιτυχούς αποστολής/λήψης ιατρικού φακέλου από το κινητό.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **1234** στο πεδίο κωδικός.
3. Επιλογή "**Είσοδος**" από το μενού (Μεταφορά στην οθόνη "**Κυρίως Μενού**").
4. Επιλογή "**Ενημέρωση**" από τη λίστα (Μεταφορά στην οθόνη "**Ενημέρωση**").
5. Επιλογή "**Bluetooth**" από τη λίστα (Μεταφορά σε οθόνη αναζήτησης).
6. Μήνυμα "**Ο φάκελος έχει αποσταλεί/ενημερωθεί**".

Αποτέλεσμα:

ΕΠΙΤΥΧΙΑ

19.2.1.8 Είσοδος στην εφαρμογή με λανθασμένο κωδικό (αποτυχία)

Βήματα:

1. Εισαγωγή μη έγκυρου 4ψήφιου κωδικού.
2. Είσοδος.

Αναμενόμενο αποτέλεσμα:

Μήνυμα λάθους.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **1455** στο πεδίο κωδικός.
3. Επιλογή "**Είσοδος**" από το μενού.
4. Μήνυμα "**ΠΡΟΣΟΧΗ: Λάθος PIN ή μη αποδεκτή κατάσταση φακέλου!**".

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.1.9 Διαγραφή ιατρικού φακέλου μετά από είσοδο με λανθασμένο κωδικό

Βήματα:

1. Εισαγωγή 4ψήφιου κωδικού.
2. Είσοδος.
3. Επιλογή διαγραφής ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Επιτυχής διαγραφή ιατρικού φακέλου.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **1234** στο πεδίο κωδικός.

3. Επιλογή “**Είσοδος**” από το μενού.
4. Μήνυμα “**ΠΡΟΣΟΧΗ: Λάθος PIN ή μη αποδεκτή κατάσταση φακέλου!**”.
5. Επιλογή “**Διαγραφή**” από το μενού.
6. Μήνυμα “**Προσοχή: Με τη διαγραφή του φακέλου θα χαθούν όλα τα δεδομένα**”.
7. Επιλογή “**Συνέχεια**” από το μενού (Έξοδος από την εφαρμογή).

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.1.10 Διαγραφή ιατρικού φακέλου (Μετά την είσοδο στην εφαρμογή)

Βήματα:

1. Εισαγωγή 4ψήφιου κωδικού.
2. Είσοδος.
3. Επιλογή διαγραφής ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Επιτυχής διαγραφή ιατρικού φακέλου.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Εισαγωγή **12058104790** στο πεδίο ΑΜΚΑ.
3. Εισαγωγή **1234** στο πεδίο κωδικός.
4. Επιλογή “**Είσοδος**” από το μενού (Μεταφορά στην οθόνη “**Κυρίως Μενού**”).
5. Επιλογή “**Διαγραφή**” από τη λίστα.
6. Μήνυμα “**Προσοχή: Με τη διαγραφή του φακέλου θα χαθούν όλα τα δεδομένα**”.
7. Επιλογή “**Συνέχεια**” από το μενού (Έξοδος από την εφαρμογή).

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.2 Υπολογιστής

19.2.2.1 Έναρξη εφαρμογής από το διαδίκτυο (JNLP)

Βήματα:

1. Σύνδεση με τον εξυπηρετητή ιστού (web server).
2. Κλικ στον σύνδεσμο έναρξης της εφαρμογής.
3. Άνοιγμα του αρχείου launch.jnlp με το Java Web Start.

Αναμενόμενο αποτέλεσμα:

Εκκίνηση της εφαρμογής στον υπολογιστή.

Δοκιμή:

1. Άνοιγμα φυλλομετρητή ιστού.

2. Εισαγωγή *http://localhost/medicalrecord/launch.html* στην διεύθυνση.
3. Κλικ στον αρχείο *launch.jnlp*.
4. Εισαγωγή της εντολής "*javaws launch.jnlp*"

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.2.2 Σύνδεση με κινητό τηλέφωνο με χρήση λανθασμένου κωδικού σύνδεσης (αποτυχία)

Βήματα:

1. Άνοιγμα του μενού Ιατρικός Φάκελος.
2. Επιλογή λήψης ιατρικού φακέλου.
3. Εισαγωγή λανθασμένου κωδικού.

Αναμενόμενο αποτέλεσμα:

Μήνυμα αποτυχίας λήψης ιατρικού φακέλου.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Κλικ στο μενού **Ιατρικός Φάκελος**.
3. Κλικ στην επιλογή **Λήψη** του μενού (Άνοιγμα νέου παραθύρου για εισαγωγή κωδικού σύνδεσης).
4. Εισαγωγή **12321** στο πεδίο κωδικός.
5. Κλικ στο πλήκτρο **Σύνδεση**.
6. Μήνυμα "**Αποτυχία λήψης ιατρικού φακέλου**".

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.2.3 Σύνδεση με κινητό τηλέφωνο και λήψη ιατρικού φακέλου

Βήματα:

1. Άνοιγμα του μενού Ιατρικός Φάκελος.
2. Επιλογή λήψης ιατρικού φακέλου.
3. Εισαγωγή κωδικού που εμφανίζεται στο κινητό τηλέφωνο.
4. Μήνυμα επιτυχούς λήψης ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Φόρτωμα των δεδομένων του ιατρικού φακέλου στον υπολογιστή.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Κλικ στο μενού **Ιατρικός Φάκελος**.
3. Κλικ στην επιλογή **Λήψη** του μενού (Άνοιγμα νέου παραθύρου για εισαγωγή κωδικού σύνδεσης).

4. Εισαγωγή **12345** στο πεδίο κωδικός.
5. Κλικ στο πλήκτρο **Σύνδεση**.
6. Μήνυμα “**Ολοκλήρωση λήψης ιατρικού φακέλου**”.

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.2.4 Τροποποίηση ιατρικού φακέλου

Βήματα:

1. Άνοιγμα της εφαρμογής.
2. Λήψη ιατρικού φακέλου.
3. Τροποποίηση των στοιχείων του ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Δυνατότητα τροποποίησης των στοιχείων.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Κλικ στο μενού **Ιατρικός Φάκελος**.
3. Κλικ στην επιλογή **Λήψη** του μενού (Άνοιγμα νέου παραθύρου για εισαγωγή κωδικού σύνδεσης).
4. Εισαγωγή **12345** στο πεδίο κωδικός.
5. Κλικ στο πλήκτρο **Σύνδεση**.
6. Μήνυμα “**Ολοκλήρωση λήψης ιατρικού φακέλου**”.
7. Κλικ στην καρτέλα **Άμεση Ανάγκη**.
8. Εισαγωγή κειμένου “**διαβητικός**” στο πεδίο κειμένου.

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

19.2.2.5 Αποστολή ενημερωμένου ιατρικού φακέλου

Βήματα:

1. Άνοιγμα της εφαρμογής.
2. Λήψη ιατρικού φακέλου.
3. Τροποποίηση των στοιχείων του ιατρικού φακέλου.
4. Επιλογή αποστολής του ιατρικού φακέλου.

Αναμενόμενο αποτέλεσμα:

Επιτυχής αποστολή ενημερωμένου ιατρικού φακέλου.

Δοκιμή:

1. Έναρξη εφαρμογής (Αρχική οθόνη).
2. Κλικ στο μενού **Ιατρικός Φάκελος**.

3. Κλικ στην επιλογή **Λήψη** του μενού (Άνοιγμα νέου παραθύρου για εισαγωγή κωδικού σύνδεσης).
4. Εισαγωγή **12345** στο πεδίο κωδικός.
5. Κλικ στο πλήκτρο **Σύνδεση**.
6. Μήνυμα "**Ολοκλήρωση λήψης ιατρικού φακέλου**".
7. Κλικ στην καρτέλα **Άμεση Ανάγκη**.
8. Εισαγωγή κειμένου "**διαβητικός**" στο πεδίο κειμένου.
9. Κλικ στο μενού **Ιατρικός Φάκελος**.
10. Κλικ στην επιλογή **Αποστολή** του μενού.
11. Μήνυμα "**Ολοκλήρωση αποστολής του ιατρικού φακέλου**".

Αποτέλεσμα: **ΕΠΙΤΥΧΙΑ**

20. Προτάσεις – Μελλοντικές Επεκτάσεις

Ασύγχρονη ενημέρωση του ιατρικού φακέλου του ασθενή από το αρμόδιο ιατρικό προσωπικό. Η πληροφορία θα αποθηκεύεται στο φορέα υποστήριξης της υπηρεσίας, η οποία είναι υπεύθυνη για την ενημέρωση του ιατρικού φακέλου στα κινητά τηλέφωνα των εγγεγραμμένων ασθενών.

- Αποθήκευση των συνταγογραφήσεων και της οδηγίας χρήσης του φαρμάκου στο κινητό τηλέφωνο του ασθενή. Προγραμματισμός ειδοποιήσεων από το κινητό τηλέφωνο για την ορθή λήψη των φαρμάκων.
- Αντίστοιχη δυνατότητα για τις συναντήσεις του ασθενή με το ιατρό.
- Υποστήριξη αποθήκευσης και επεξεργασίας της κάρτας εμβολιασμών του ασθενή.
- Συγχρονισμός στοιχείων ιατρικού φακέλου μεταξύ κινητού τηλεφώνου και έξυπνης κάρτας (smartcard).
- Υποστήριξη του προτύπου Clinical Document Architecture – CDA του Οργανισμού Health Level 7 – HL7.
- Πρόσβαση στον ιατρικό φάκελο του κινητού τηλεφώνου από εξουσιοδοτημένο ιατρικό προσωπικό με την εισαγωγή αναγνωριστικών.
- Κλήση άμεσης βοήθειας (πχ Ε.ΚΕ.Π.Υ., Ε.Κ.Α.Β) με ταυτόχρονη αποστολή ιατρικού φακέλου στον φορέα υποστήριξης του συστήματος, Η υπηρεσία ενεργοποιείται με το πάτημα ενός πλήκτρου στο κινητό.
- Αποστόλη του γεωγραφικού στίγματος (GPS) του ασθενή κατά την κλήση άμεσης βοήθειας.

- Ασύρματη διασύνδεση με τον εγκέφαλο του αυτοκινήτου του ασθενή με συνεχή παρακολούθηση των δεδομένων και της κατάστασης του αυτοκινήτου. Αυτόματη κλήση άμεσης βοήθειας με βάση τα στοιχεία που λαμβάνονται από το αυτοκίνητο. Ηλεκτρονικός Ιατρικός Φάκελος σε Κινητό Τηλέφωνο και Εφαρμογές του

20.1 Health Level 7

Το Health Level 7 (HL7), είναι μια πλήρως εθελοντική, μη κερδοσκοπική οργάνωση που εμπλέκεται στην ανάπτυξη των διεθνών προτύπων υγειονομικής περίθαλψης. Μερικά πρότυπα που δημιουργήθηκαν από τον οργανισμό είναι το HL7 v2.x, v3.0 και HL7 RIM.

Το HL7 και τα μέλη της παρέχουν ένα πλαίσιο (και τα σχετικά πρότυπα) για την ανταλλαγή, την ολοκλήρωση, το διαμοίρασμα και ανάκτηση των ηλεκτρονικών πληροφοριών για την υγεία. Το v2.x των προτύπων, το οποίο υποστηρίζει την κλινική πρακτική και τη διαχείριση, την παράδοση, και την αξιολόγηση των υπηρεσιών υγείας , είναι το πιο συχνά χρησιμοποιούμενο στον κόσμο.

Βιβλιογραφία

[1] "Bluetooth traveler". www.hoovers.com. http://www.hoovers.com/business-information/--pageid__13751--/global-hoov-index.xhtml

[2] Monson, Heidi (1999-12-14). "Bluetooth Technology and Implications". SysOpt.com. http://www.sysopt.com/features/network/article.php/3532506.

[3] "About the Bluetooth SIG". Bluetooth SIG. http://www.bluetooth.com/Bluetooth/SIG/

[4] Kardach, Jim (2008-05-03). "How Bluetooth got its name". http://www.eetimes.eu/scandinavia/206902019?cid=RSSfeed_eetimesEU_scandinavia.

[5] Newton, Harold. (2007). Newton’s telecom dictionary. New York: Flatiron Publishing.

[6] "How Bluetooth Technology Works". Bluetooth SIG. http://www.bluetooth.com/Bluetooth/Technology/Works/

[7] http://www.bluetooth.com/English/Technology/Works/Pages/Profiles_Overview.aspx

[8] Hypertag.com

[9] "Wii Controller". Bluetooth SIG. http://bluetooth.com/Bluetooth/Products/Products/Product_Details.htm?ProductID=2951.

[10] Telemedicine.jp

[11] http://www.bluetooth.com/English/Products/Pages/Watch.aspx

[12] Apple (2002-07-17). "Apple Introduces "Jaguar," the Next Major Release of Mac OS X". Press release. http://www.apple.com/pr/library/2002/jul/17jaguar.html.

[13] BlueZ - Official Linux Bluetooth protocol stack

[14] OMTP.org

[15] http://www.bluetooth.com/English/Experience/Pages/On_Your_Phone.aspx

[16] "The Bluetooth Blues". Information Age. 2001-05-24. Archived from the original on 2007-12-22. http://web.archive.org/web/20071222231740/http://www.information-age.com/article/2001/may/the_bluetooth_blues.

[17] IEEE Std 802.15.1-2002 – IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)

[18] IEEE Std 802.15.1-2005 – IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)

[19] Guy Kewney (2004-11-16). "High speed Bluetooth comes a step closer: enhanced data rate approved". Newswireless.net. http://www.newswireless.net/index.cfm/article/629

[20] "Specification Documents". Bluetooth SIG. http://www.bluetooth.com/Bluetooth/Technology/Building/Specifications/.

[21] "HTC TyTN Specification" (PDF). HTC. http://www.europe.htc.com/z/pdf/products/1766_TyTN_LFLT_OUT.PDF.

[22] Simple Pairing Whitepaper. Version V10r00. Bluetooth SIG. 2006-08-03. http://bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf.

[23] David Meyer (2009-04-22). "Bluetooth 3.0 released without ultrawideband". zdnet.co.uk. http://news.zdnet.co.uk/communications/0,1000000085,39643174,00.htm.

[24] Bluetooth.com

[25] Nokia (2007-06-12). "Wibree forum merges with Bluetooth SIG"

[26] http://www.wibree.com/press/Wibree_pressrelease_final_1206.pdf.

[27] Wimedia.org

[28] Bluetooth.com

[29] USB.org

[30] Incisor.tv

[31] Bluetooth group drops ultrawideband, eyes 60 GHz, Report: Ultrawideband dies by 2013, Incisor Magazine November 2009

[32] Stallings, William. (2005). Wireless communications & networks.'=' Upper Saddle River, NJ: Pearson Prentice Hall.

[33] D. Chomienne, M. Eftimakis (2010-10-20). "Bluetooth Tutorial" http://www.newlogic.com/products/Bluetooth-Tutorial-2001.pdf.

[34] Juha T. Vainio (2000-05-25). "Bluetooth Security". Helsinki University of Technology. http://www.iki.fi/jiitv/bluesec.pdf

[35] Andreas Becker (2007-08-16) (PDF). Bluetooth Security & Hacks. Ruhr-Universität Bochum. http://gsyc.es/~anto/ubicuos2/bluetooth_security_and_hacks.pdf

[36] Scarfone, K., and Padgette, J. (September 2008) (PDF). Guide to Bluetooth Security. National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf.

[37] "What is bluejacking?". Helsinki University of Technology. http://www.bluejackq.com/what-is-bluejacking.shtml.

[38] "Security Weaknesses in Bluetooth". RSA Security Conf. – Cryptographer's Track. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.7357.

[39] "Bluetooth". The Bunker. http://www.thebunker.net/resources/bluetooth.

[40] "BlueBug". Trifinite.org. http://trifinite.org/trifinite_stuff_bluebug.html.

[41] John Oates (2004-06-15). "Virus attacks mobiles via Bluetooth". The Register. http://www.theregister.co.uk/2004/06/15/symbian_virus/.

[42] "Long Distance Snarf". Trifinite.org. http://trifinite.org/trifinite_stuff_lds.html.

[43] "F-Secure Malware Information Pages: Lasco.A". F-Secure.com. http://www.f-secure.com/v-descs/lasco_a.shtml.

[44] Ford-Long Wong, Frank Stajano, Jolyon Clulow (2005-04) (PDF). Repairing the Bluetooth pairing protocol. University of Cambridge Computer Laboratory. http://www.cl.cam.ac.uk/~fw242/publications/2005-WongStaClu-bluetooth.pdf.

[45] Yaniv Shaked, Avishai Wool (2005-05-02). Cracking the Bluetooth PIN. School of Electrical Engineering Systems, Tel Aviv University. http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/.

[46] "Phone pirates in seek and steal mission". Cambridge Evening News http://web.archive.org/web/20070717035938/http://www.cambridge-news.co.uk/news/region_wide/2005/08/17/06967453-8002-45f8-b520-66b9bed6f29f.lpf.

[47] Going Around with Bluetooth in Full Safety. F-Secure. 2006-05. http://www.securenetwork.it/bluebag_brochure.pdf.

[48] M. Hietanen, T. Alanko (2005-10). "Occupational Exposure Related to Radiofrequency Fields from Wireless Communication Systems" (PDF). XXVIIIth General Assembly of URSI - Proceedings. Union Radio-Scientifique Internationale. http://www.ursi.org/Proceedings/ProcGA05/pdf/K03.7(01682).pdf.

[49]

[50] Ortiz, C. Enrique; Giguere, Eric (January 15, 2001), Mobile Information Device Profile for Java 2 Micro Edition (1st ed.), John Wiley & Sons, pp. 352, ISBN 0471034657

[51] Hayun, Roy Ben (March 30, 2009). Java ME on Symbian OS: Inside the Smartphone Model (1st ed.). Wiley. pp. 482. ISBN 0470743182. http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470743182.html.

[52] Knudsen, Jonathan (January 8, 2008). Kicking Butt with MIDP and MSA: Creating Great Mobile Applications (1st ed.). Prentice Hall. pp. 432. ISBN 0321463420. http://www.informit.com/store/product.aspx?isbn=9780321463425.

[53] Li, Sing; Knudsen, Jonathan (April 25, 2005). Beginning J2ME: From Novice to Professional (3rd ed.). Apress. pp. 480. ISBN 1590594797. http://www.apress.com/book/view/9781590594797.

[54] Java Web Start 1.0 press release

[55] Pack200 and Compression for Network Deployment

[56] Bug ID 4802695, Support 64-bit Java Plug-in and Java webstart on Windows/Linux on AMD64

[57] Nicolas Courtois, Josef Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations". pp267–287, ASIACRYPT 2002.

[58] Joan Daemen, Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer, 2002. ISBN 3-540-42580-2.

[59] Christof Paar, Jan Pelzl, "The Advanced Encryption Standard", Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". Springer, 2009.

[60] Biham, Eli and Adi Shamir (1991). "Differential Cryptanalysis of DES-like Cryptosystems". Journal of Cryptology 4 (1): 3–72. doi:10.1007/BF00630563. http://www.springerlink.com/content/k54h077np8714058/.

[61] Biham, Eli and Adi Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer Verlag, 1993. ISBN 0-387-97930-1, ISBN 3-540-97930-1.

[62] Biham, Eli and Alex Biryukov: An Improvement of Davies' Attack on DES. J. Cryptology 10(3): 195–206 (1997)

[63] Biham, Eli, Orr Dunkelman, Nathan Keller: Enhancing Differential-Linear Cryptanalysis. ASIACRYPT 2002: pp254–266

[64] Biham, Eli. A Fast New DES Implementation in Software Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design, Electronic Frontier Foundation

[65] Biryukov, A, C. De Canniere and M. Quisquater (2004). "On Multiple Linear Approximations". Lecture Notes in Computer Science 3152: 1–22. doi:10.1007/b99099. http://www.springerlink.com/content/16udaqwwl9ffrtxt/.

[66] Campbell, Keith W., Michael J. Wiener: DES is not a Group. CRYPTO 1992: pp512–520

[67] Coppersmith, Don. (1994). The data encryption standard (DES) and its strength against attacks. IBM Journal of Research and Development, 38(3), 243–250.

[68] Diffie, Whitfield and Martin Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard" IEEE Computer 10(6), June 1977, pp74–84

[69] Ehrsam et al., Product Block Cipher System for Data Security, U.S. Patent 3,962,539, Filed February 24, 1975

[70] Gilmore, John, "Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design", 1998, O'Reilly, ISBN 1-56592-520-3.

[71] Junod, Pascal. "On the Complexity of Matsui's Attack." Selected Areas in Cryptography, 2001, pp199–211.

[72] Kaliski, Burton S., Matt Robshaw: Linear Cryptanalysis Using Multiple Approximations. CRYPTO 1994: pp26–39

[73] Knudsen, Lars, John Erik Mathiassen: A Chosen-Plaintext Linear Attack on DES. Fast Software Encryption - FSE 2000: pp262–272

[74] Langford, Susan K., Martin E. Hellman: Differential-Linear Cryptanalysis. CRYPTO 1994: 17–25

[75] Levy, Steven, Crypto: How the Code Rebels Beat the Government—Saving Privacy in the Digital Age, 2001, ISBN 0-14-024432-8.

[76] Matsui, Mitsuru (1994). "Linear Cryptanalysis Method for DES Cipher". Lecture Notes in Computer Science 765: 386–397. doi:10.1007/3-540-48285-7. http://www.springerlink.com/content/92509p5l4ravyn62/.

[77] Mitsuru Matsui (1994). "The First Experimental Cryptanalysis of the Data Encryption Standard". Lecture Notes in Computer Science 839: 1–11. doi:10.1007/3-540-48658-5_1. http://www.springerlink.com/content/vrteugmt7erqqbw1/.

[78] National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[79] GSM Doc 28/85 "Services and Facilities to be provided in the GSM System" rev2, June 1985

[80] LA Times: Why text messages are limited to 160 characters

[81] GSM 03.40 Technical realization of the Short Message Service (SMS).

[82] see GSM document 02/82 available in the ETSI archive

[83] These Message Handling Systems had been standardized in the ITU, see specifications X.400 series

[84] See the book Hillebrand, Trosby, Holley, Harris: SMS the creation of Personal Global Text Messaging, Wiley 2010

[85] See GSM document 28/85rev.2 of June 85 and GSM WP1 document 66/86 available in the ETSI archive

[86] See also Friedhelm Hillebrand "GSM and UMTS, the creation of Global Mobile Communication", Wiley 2002, chapters 10 and 16, ISBN 0470 84322 5

[87] GSM document 19/85, available in the ETSI archive

[88] GSM document 28/85r2, available in the ETSI archive

[89] GSM TS 02.03, Teleservices Supported by a GSM Public Land Mobile Network (PLMN).

[90] Document GSM IDEG 79/87r3, available in the ETSI archive

[91] GSM 03.40, WP4 document 152/87, available in the ETSI archive

[92] Finn Trosby, "the strange duckling of GSM SMS", Telektronikk Vol.3 2004.

[93] MAP phase 1 specification, available from the 3GPP web site.

[94] MAP phase 2 specification, available from the 3GPP web site.

[95] CAMEL Phase 3 specification, available from the 3GPP web site.

[96] CAMEL Phase 4 specification, also available from the 3GPP specification page.

[97] Hppy bthdy txt! December 2002, BBC News.

[98] UK hails 10th birthday of SMS, December 2002, The Times of India.

[99] False dawn of the photo phone boom, Jan 2003, The Scotsman.

[100] First commercial deployment of Text Messaging (SMS)

[101] GSM World press release

[102] Crystal, David (2008-07-05). "2b or not 2b?". Guardian Unlimited. http://books.guardian.co.uk/departments/referenceandlanguages/story/0,,2289259,00.html

[103] ITU Internet Report 2006: digital.life, Chapter 3

[104] http://www.dslreports.com/shownews/91379

[105] GSM 03.41, Technical Realization of Short Message Service Cell Broadcast (SMSCB).

[106] Gil Held: "Data over Wireless Networks". page 105-111, 137-138. Wiley, 2001.

[107] 3GPP TS 23.038, Alphabets and language-specific information.

[108] Ian Groves: "Mobile Systems", page 70, 79, 163-166. Chapman & Hall, 1998.

[109] "t-zones text messaging: send and receive messages with mobile text messaging". T-mobile.com. http://www.t-mobile.com/shop/addons/services/TzonesDetail.aspx?tp=Svc_Tab_TZones&tsp=Svc_Sub_Messaging&tssp=Svc_Sub_TextMessaging&oscid=4CD51BA7-B5AF-4AB2-85E0-50EC0AF141F9.

[110] "Support - How do I compose and send a text message to a Sprint or Nextel customer from email?". http://support.sprintpcs.com/doc/sp7648.xml?selectedDeviceId=5707&related=y&Referring%20Related%20DocID%20List%20Index=5&docid=7434&navtypeid=10&pagetypeid=7&prevPageIndex=10.

[111] "Answers to FAQs - Verizon Wireless Support". http://support.vzw.com/faqs/Picture%20Messaging/faq_pixmessaging.html#item3.

[112] BT trials mobile SMS to voice landline, January 2004, The Register.

[113] [1], September 2006, SMStextnews

[114] SMS Tutorial: Introduction to AT Commands, Basic Commands and Extended Commands

[115] Solutions to the GSM Security Weaknesses, Proceedings of the 2nd IEEE International Conference on Next Generation Mobile Applications, Services, and Technologies (NGMAST2008), pp.576-581, Cardiff, UK, September 2008

[116] SSMS - A Secure SMS Messaging Protocol for the M-Payment Systems, Proceedings of the 13th IEEE Symposium on Computers and Communications (ISCC'08), pp. 700-705, July 2008

[117] An Analysis of Vulnerabilities in SMS-Capable Cellular Networks: Exploiting Open Functionality in SMS-Capable Cellular Networks (Website)

[118] An overview on how to stop SMS Spoofing in mobile operator networks (September 9, 2008).

Παραρτήματα

Παράρτημα Α – Προτυποποίηση ΥΥΚΑ

Για την δημιουργία του ιατρικού φακέλου χρησιμοποιήθηκε η απόσπασμα της παραγράφου “**I4**” του εντύπου “**4. Έντυπα Ιατρικής Υπηρεσίας**”. Το απόσπασμα εμφανίζεται στην παρακάτω εικόνα.

Σχήμα 22: Έντυπο Ιατρικής Υπηρεσίας I4

Παράρτημα Β – Κώδικας

DesktopMedicalRecord

```
/*
 * DesktopMedicalRecord.java
 */

package desktopmedicalrecord;

import org.jdesktop.application.Application;
import org.jdesktop.application.SingleFrameApplication;

/**
 * The main class of the application.
 */
public class DesktopMedicalRecord extends SingleFrameApplication {

    /**
     * At startup create and show the main frame of the application.
     */
    @Override protected void startup() {
        show(new DesktopMedicalRecordView(this));
    }

    /**
        * This method is to initialize the specified window by injecting
resources.
     * Windows shown in our application come fully initialized from the GUI
     * builder, so this additional configuration is not needed.
     */
    @Override protected void configureWindow(java.awt.Window root) {
    }

    /**
     * A convenient static getter for the application instance.
     * @return the instance of DesktopMedicalRecord
     */
    public static DesktopMedicalRecord getApplication() {
        return Application.getInstance(DesktopMedicalRecord.class);
    }

    /**
     * Main method launching the application.
     */
    public static void main(String[] args) {
        launch(DesktopMedicalRecord.class, args);
    }
}
```

```
/*
 * DesktopMedicalRecordAboutBox.java
 */
package desktopmedicalrecord;

import org.jdesktop.application.Action;

public class DesktopMedicalRecordAboutBox extends javax.swing.JDialog {

    public DesktopMedicalRecordAboutBox(java.awt.Frame parent) {
        super(parent);
        initComponents();
        getRootPane().setDefaultButton(closeButton);
    }

    @Action
    public void closeAboutBox() {
        dispose();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        closeButton = new javax.swing.JButton();
        javax.swing.JLabel appTitleLabel = new javax.swing.JLabel();
        javax.swing.JLabel versionLabel = new javax.swing.JLabel();
        javax.swing.JLabel appVersionLabel = new javax.swing.JLabel();
        javax.swing.JLabel vendorLabel = new javax.swing.JLabel();
        javax.swing.JLabel appVendorLabel = new javax.swing.JLabel();
        javax.swing.JLabel homepageLabel = new javax.swing.JLabel();
        javax.swing.JLabel appHomepageLabel = new javax.swing.JLabel();
        javax.swing.JLabel appDescLabel = new javax.swing.JLabel();
        javax.swing.JLabel imageLabel = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
                        org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
icalRecord.class).getContext().getResourceMap(DesktopMedicalRecordAboutBox.class
);
        setTitle(resourceMap.getString("title")); // NOI18N
        setModal(true);
        setName("aboutBox"); // NOI18N
        setResizable(false);

                                    javax.swing.ActionMap actionMap =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
icalRecord.class).getContext().getActionMap(DesktopMedicalRecordAboutBox.class,
this);
        closeButton.setAction(actionMap.get("closeAboutBox")); // NOI18N
            closeButton.setText(resourceMap.getString("closeButton.text")); //
NOI18N
        closeButton.setName("closeButton"); // NOI18N
```

```
          appTitleLabel.setFont(appTitleLabel.getFont().deriveFont(appTitleLabel
.getFont().getStyle()  |  java.awt.Font.BOLD,  appTitleLabel.getFont().getSize()
+4));
           appTitleLabel.setText(resourceMap.getString("Application.title")); //
NOI18N
          appTitleLabel.setName("appTitleLabel"); // NOI18N

          versionLabel.setFont(versionLabel.getFont().deriveFont(versionLabel.ge
tFont().getStyle() | java.awt.Font.BOLD));
            versionLabel.setText(resourceMap.getString("versionLabel.text")); //
NOI18N
          versionLabel.setName("versionLabel"); // NOI18N

          appVersionLabel.setText(resourceMap.getString("Application.version"));
// NOI18N
          appVersionLabel.setName("appVersionLabel"); // NOI18N

          vendorLabel.setFont(vendorLabel.getFont().deriveFont(vendorLabel.getFo
nt().getStyle() | java.awt.Font.BOLD));
             vendorLabel.setText(resourceMap.getString("vendorLabel.text")); //
NOI18N
          vendorLabel.setName("vendorLabel"); // NOI18N

          appVendorLabel.setText(resourceMap.getString("Application.vendor")); /
/ NOI18N
          appVendorLabel.setName("appVendorLabel"); // NOI18N

          homepageLabel.setFont(homepageLabel.getFont().deriveFont(homepageLabel
.getFont().getStyle() | java.awt.Font.BOLD));
          homepageLabel.setText(resourceMap.getString("homepageLabel.text")); //
NOI18N
          homepageLabel.setName("homepageLabel"); // NOI18N

          appHomepageLabel.setText(resourceMap.getString("Application.homepage")
); // NOI18N
          appHomepageLabel.setName("appHomepageLabel"); // NOI18N

            appDescLabel.setText(resourceMap.getString("appDescLabel.text")); //
NOI18N
          appDescLabel.setName("appDescLabel"); // NOI18N

          imageLabel.setIcon(resourceMap.getIcon("imageLabel.icon")); // NOI18N
          imageLabel.setName("imageLabel"); // NOI18N

                                      javax.swing.GroupLayout    layout    =    new
javax.swing.GroupLayout(getContentPane());
          getContentPane().setLayout(layout);
          layout.setHorizontalGroup(
               layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
              .addGroup(layout.createSequentialGroup()
                  .addComponent(imageLabel)
                  .addGap(18, 18, 18)
                   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.TRAILING)
                             .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                               .addComponent(versionLabel)
                               .addComponent(vendorLabel)
```

```
                                .addComponent(homepageLabel))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.RELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.Group
Layout.Alignment.LEADING)
                                .addComponent(appVersionLabel)
                                .addComponent(appVendorLabel)
                                .addComponent(appHomepageLabel)))
                                                        .addComponent(appTitleLabel,
javax.swing.GroupLayout.Alignment.LEADING)
                                                        .addComponent(appDescLabel,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
253, Short.MAX_VALUE)
                        .addComponent(closeButton))
                    .addContainerGap())
          );
          layout.setVerticalGroup(
               layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                .addComponent(imageLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
237, Short.MAX_VALUE)
               .addGroup(layout.createSequentialGroup()
                   .addContainerGap()
                   .addComponent(appTitleLabel)
                   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                                                        .addComponent(appDescLabel,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                       .addComponent(versionLabel)
                       .addComponent(appVersionLabel))
                   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                       .addComponent(vendorLabel)
                       .addComponent(appVendorLabel))
                   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                   .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.A
lignment.BASELINE)
                       .addComponent(homepageLabel)
                       .addComponent(appHomepageLabel))
                   .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                   .addComponent(closeButton)
                   .addContainerGap())
          );

          pack();
      }// </editor-fold>
      // Variables declaration - do not modify
      private javax.swing.JButton closeButton;
      // End of variables declaration
  }
```

```
/*
 * DesktopMedicalRecordView.java
 */
package desktopmedicalrecord;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.bluetooth.BluetoothStateException;
import org.jdesktop.application.Action;
import org.jdesktop.application.ResourceMap;
import org.jdesktop.application.SingleFrameApplication;
import org.jdesktop.application.FrameView;
import org.jdesktop.application.Task;
import org.jdesktop.application.TaskMonitor;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.lang.reflect.Field;
import java.util.Hashtable;
import javax.swing.Timer;
import javax.swing.Icon;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JTable;
import javax.swing.table.TableColumn;
import javax.swing.table.TableColumnModel;
import javax.swing.text.JTextComponent;
import obex.OBEXClient;
import org.ntua.mobile.medicalrecord.storage.EmergencyRecord;
import org.ntua.mobile.medicalrecord.storage.GenericRecord;
import org.ntua.mobile.medicalrecord.storage.MedicalRecord;

/**
 * The application's main frame.
 */
public class DesktopMedicalRecordView extends FrameView {

    private OBEXClient obexclient;
    private JDialog dialogBox;

    public DesktopMedicalRecordView(SingleFrameApplication app) {
        super(app);

        initComponents();

          // status bar initialization - message timeout, idle icon and busy
animation, etc
        ResourceMap resourceMap = getResourceMap();
                                                    int      messageTimeout      =
resourceMap.getInteger("StatusBar.messageTimeout");
        messageTimer = new Timer(messageTimeout, new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                statusMessageLabel.setText("");
            }
        });
        messageTimer.setRepeats(false);
                                                int      busyAnimationRate      =
resourceMap.getInteger("StatusBar.busyAnimationRate");
        for (int i = 0; i < busyIcons.length; i++) {
```

```
                busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i +
"]");
            }
            busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent e) {
                    busyIconIndex = (busyIconIndex + 1) % busyIcons.length;
                    statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);
                }
            });
            idleIcon = resourceMap.getIcon("StatusBar.idleIcon");
            statusAnimationLabel.setIcon(idleIcon);
            progressBar.setVisible(false);

            // connecting action tasks to status bar via TaskMonitor
                                        TaskMonitor     taskMonitor     =     new
TaskMonitor(getApplication().getContext());
                                       taskMonitor.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {

                @Override
                public void propertyChange(java.beans.PropertyChangeEvent evt) {
                    String propertyName = evt.getPropertyName();
                    if ("started".equals(propertyName)) {
                        if (!busyIconTimer.isRunning()) {
                            statusAnimationLabel.setIcon(busyIcons[0]);
                            busyIconIndex = 0;
                            busyIconTimer.start();
                        }
                        progressBar.setVisible(true);
                        progressBar.setIndeterminate(true);
                    } else if ("done".equals(propertyName)) {
                        busyIconTimer.stop();
                        statusAnimationLabel.setIcon(idleIcon);
                        progressBar.setVisible(false);
                        progressBar.setValue(0);
                    } else if ("message".equals(propertyName)) {
                        String text = (String) (evt.getNewValue());
                        statusMessageLabel.setText((text == null) ? "" : text);
                        messageTimer.restart();
                    } else if ("progress".equals(propertyName)) {
                        int value = (Integer) (evt.getNewValue());
                        progressBar.setVisible(true);
                        progressBar.setIndeterminate(false);
                        progressBar.setValue(value);
                    }
                }
            });
        }

        @Action
        public void showAboutBox() {
            if (aboutBox == null) {
                                                            JFrame     mainFrame     =
DesktopMedicalRecord.getApplication().getMainFrame();
                aboutBox = new DesktopMedicalRecordAboutBox(mainFrame);
                aboutBox.setLocationRelativeTo(mainFrame);
            }
            DesktopMedicalRecord.getApplication().show(aboutBox);
        }
```

```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    mainPanel = new javax.swing.JPanel();
    jTabbedPane1 = new javax.swing.JTabbedPane();
    jPanel1 = new javax.swing.JPanel();
    amkaLabel = new javax.swing.JLabel();
    surnameLabel = new javax.swing.JLabel();
    nameLabel = new javax.swing.JLabel();
    bloodTypeRhLabel = new javax.swing.JLabel();
    addressLabel = new javax.swing.JLabel();
    postalCodeLabel = new javax.swing.JLabel();
    cityLabel = new javax.swing.JLabel();
    phoneNumberLabel = new javax.swing.JLabel();
    amka = new javax.swing.JTextField();
    surname = new javax.swing.JTextField();
    name = new javax.swing.JTextField();
    bloodTypeRh = new javax.swing.JTextField();
    address = new javax.swing.JTextField();
    postalCode = new javax.swing.JTextField();
    city = new javax.swing.JTextField();
    phoneNumber = new javax.swing.JTextField();
    jPanel8 = new javax.swing.JPanel();
    nrSurnameLabel = new javax.swing.JLabel();
    nrNameLabel = new javax.swing.JLabel();
    nrPhoneNumberLabel = new javax.swing.JLabel();
    nearestRelativeSurname = new javax.swing.JTextField();
    nearestRelativeName = new javax.swing.JTextField();
    nearestRelativePhoneNumber = new javax.swing.JTextField();
    nrRelationshipLabel = new javax.swing.JLabel();
    nearestRelativeRelationship = new javax.swing.JTextField();
    jPanel2 = new javax.swing.JPanel();
    jScrollPane1 = new javax.swing.JScrollPane();
    historyExamEvaluation = new javax.swing.JTextArea();
    jPanel3 = new javax.swing.JPanel();
    jScrollPane2 = new javax.swing.JScrollPane();
    diceaseProgress = new javax.swing.JTextArea();
    jPanel4 = new javax.swing.JPanel();
    jScrollPane3 = new javax.swing.JScrollPane();
    jTextArea3 = new javax.swing.JTextArea();
    jPanel5 = new javax.swing.JPanel();
    jScrollPane4 = new javax.swing.JScrollPane();
    exitDiagnosis = new javax.swing.JTextArea();
    jPanel6 = new javax.swing.JPanel();
    JScrollPanel5 = new javax.swing.JScrollPane();
    treatmentIntervention = new javax.swing.JTextArea();
    jPanel7 = new javax.swing.JPanel();
    jLabel9 = new javax.swing.JLabel();
    jScrollPane6 = new javax.swing.JScrollPane();
    ecg = new javax.swing.JTextArea();
    jLabel10 = new javax.swing.JLabel();
    jScrollPane8 = new javax.swing.JScrollPane();
    radiography = new javax.swing.JTextArea();
    jLabel11 = new javax.swing.JLabel();
```

```
        jScrollPane9 = new javax.swing.JScrollPane();
        otherExams = new javax.swing.JTextArea();
        jPanel10 = new javax.swing.JPanel();
        jSplitPane1 = new javax.swing.JSplitPane();
        jScrollPane5 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        jScrollPane7 = new javax.swing.JScrollPane();
        jTable2 = new javax.swing.JTable();
        jPanel11 = new javax.swing.JPanel();
        JScrollPanel6 = new javax.swing.JScrollPane();
        treatmentIntervention1 = new javax.swing.JTextArea();
        jPanel9 = new javax.swing.JPanel();
        submitButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();
        menuBar = new javax.swing.JMenuBar();
        javax.swing.JMenu fileMenu = new javax.swing.JMenu();
        javax.swing.JMenuItem exitMenuItem = new javax.swing.JMenuItem();
        medicalRecordMenu = new javax.swing.JMenu();
        getRecordMenuItem = new javax.swing.JMenuItem();
        putRecordMenuItem = new javax.swing.JMenuItem();
        javax.swing.JMenu helpMenu = new javax.swing.JMenu();
        aboutMenuItem = new javax.swing.JMenuItem();
        statusPanel = new javax.swing.JPanel();
                        javax.swing.JSeparator statusPanelSeparator = new
javax.swing.JSeparator();
        statusMessageLabel = new javax.swing.JLabel();
        statusAnimationLabel = new javax.swing.JLabel();
        progressBar = new javax.swing.JProgressBar();

        mainPanel.setMaximumSize(new java.awt.Dimension(800, 600));
        mainPanel.setMinimumSize(new java.awt.Dimension(200, 400));
        mainPanel.setName("mainPanel"); // NOI18N
        mainPanel.setPreferredSize(new java.awt.Dimension(600, 400));
                    mainPanel.setLayout(new javax.swing.BoxLayout(mainPanel,
javax.swing.BoxLayout.PAGE_AXIS));

        jTabbedPane1.setMaximumSize(new java.awt.Dimension(800, 600));
        jTabbedPane1.setMinimumSize(new java.awt.Dimension(600, 300));
        jTabbedPane1.setName("jTabbedPane1"); // NOI18N
        jTabbedPane1.setPreferredSize(new java.awt.Dimension(600, 300));

        jPanel1.setName("jPanel1"); // NOI18N
        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

                        org.jdesktop.application.ResourceMap resourceMap =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
icalRecord.class).getContext().getResourceMap(DesktopMedicalRecordView.class);
        amkaLabel.setText(resourceMap.getString("amkaLabel.text")); // NOI18N
        amkaLabel.setName("amkaLabel"); // NOI18N
                                            jPanel1.add(amkaLabel,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 15, 121, -1));

          surnameLabel.setText(resourceMap.getString("surnameLabel.text")); //
NOI18N
        surnameLabel.setName("surnameLabel"); // NOI18N
                                          jPanel1.add(surnameLabel,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 56, 121, -1));

        nameLabel.setText(resourceMap.getString("nameLabel.text")); // NOI18N
        nameLabel.setName("nameLabel"); // NOI18N
```

```
                                                    jPanel1.add(nameLabel,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 97, 121, -1));

          bloodTypeRhLabel.setText(resourceMap.getString("bloodTypeRhLabel.text"
)); // NOI18N
          bloodTypeRhLabel.setName("bloodTypeRhLabel"); // NOI18N
                                                jPanel1.add(bloodTypeRhLabel,     new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 138, -1, -1));

            addressLabel.setText(resourceMap.getString("addressLabel.text")); //
NOI18N
          addressLabel.setName("addressLabel"); // NOI18N
                                                  jPanel1.add(addressLabel,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 179, 121, -1));

          postalCodeLabel.setText(resourceMap.getString("postalCodeLabel.text"))
; // NOI18N
          postalCodeLabel.setName("postalCodeLabel"); // NOI18N
                                                   jPanel1.add(postalCodeLabel,     new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 220, 121, -1));

          cityLabel.setText(resourceMap.getString("cityLabel.text")); // NOI18N
          cityLabel.setName("cityLabel"); // NOI18N
                                                    jPanel1.add(cityLabel,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 261, 121, -1));

          phoneNumberLabel.setText(resourceMap.getString("phoneNumberLabel.text"
)); // NOI18N
          phoneNumberLabel.setName("phoneNumberLabel"); // NOI18N
                                                jPanel1.add(phoneNumberLabel,     new
org.netbeans.lib.awtextra.AbsoluteConstraints(12, 302, 121, -1));

          amka.setText(resourceMap.getString("amka.text")); // NOI18N
          amka.setName("amka"); // NOI18N
                                                       jPanel1.add(amka,       new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 12, 291, -1));

          surname.setText(resourceMap.getString("surname.text")); // NOI18N
          surname.setName("surname"); // NOI18N
                                                     jPanel1.add(surname,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 53, 291, -1));

          name.setText(resourceMap.getString("name.text")); // NOI18N
          name.setName("name"); // NOI18N
                                                       jPanel1.add(name,       new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 94, 291, -1));

              bloodTypeRh.setText(resourceMap.getString("bloodTypeRh.text")); //
NOI18N
          bloodTypeRh.setName("bloodTypeRh"); // NOI18N
                                                   jPanel1.add(bloodTypeRh,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 135, 291, -1));

          address.setText(resourceMap.getString("address.text")); // NOI18N
          address.setName("address"); // NOI18N
                                                       jPanel1.add(address,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 176, 291, -1));

                postalCode.setText(resourceMap.getString("postalCode.text")); //
NOI18N
          postalCode.setName("postalCode"); // NOI18N
```

```
                                                    jPanel1.add(postalCode,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 217, 291, -1));

          city.setName("city"); // NOI18N
                                                            jPanel1.add(city,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 258, 291, -1));

          phoneNumber.setName("phoneNumber"); // NOI18N
                                                          jPanel1.add(phoneNumber,      new
org.netbeans.lib.awtextra.AbsoluteConstraints(145, 299, 291, -1));

           jTabbedPane1.addTab(resourceMap.getString("jPanel1.TabConstraints.tabT
itle"), jPanel1); // NOI18N

          jPanel8.setName("jPanel8"); // NOI18N

           nrSurnameLabel.setText(resourceMap.getString("nrSurnameLabel.text"));
// NOI18N
          nrSurnameLabel.setName("nrSurnameLabel"); // NOI18N

              nrNameLabel.setText(resourceMap.getString("nrNameLabel.text")); //
NOI18N
          nrNameLabel.setName("nrNameLabel"); // NOI18N

           nrPhoneNumberLabel.setText(resourceMap.getString("nrPhoneNumberLabel.t
ext")); // NOI18N
          nrPhoneNumberLabel.setName("nrPhoneNumberLabel"); // NOI18N

          nearestRelativeSurname.setName("nearestRelativeSurname"); // NOI18N

          nearestRelativeName.setName("nearestRelativeName"); // NOI18N

            nearestRelativePhoneNumber.setName("nearestRelativePhoneNumber"); //
NOI18N

           nrRelationshipLabel.setText(resourceMap.getString("nrRelationshipLabel
.text")); // NOI18N
          nrRelationshipLabel.setName("nrRelationshipLabel"); // NOI18N

           nearestRelativeRelationship.setName("nearestRelativeRelationship"); //
NOI18N

                                javax.swing.GroupLayout    jPanel8Layout    =    new
javax.swing.GroupLayout(jPanel8);
          jPanel8.setLayout(jPanel8Layout);
          jPanel8Layout.setHorizontalGroup(
               jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
              .addGroup(jPanel8Layout.createSequentialGroup()
                  .addContainerGap()
                   .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel8Layout.createSequentialGroup()
                                                    .addComponent(nrPhoneNumberLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,       javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
```

```
                                        .addComponent(nearestRelativePhoneNumber,
javax.swing.GroupLayout.PREFERRED_SIZE,                                         291,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel8Layout.createSequentialGroup()
                                                    .addComponent(nrNameLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                                                .addComponent(nearestRelativeName,
javax.swing.GroupLayout.PREFERRED_SIZE,                                         291,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel8Layout.createSequentialGroup()
                                                        .addComponent(nrSurnameLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                                                .addComponent(nearestRelativeSurname,
javax.swing.GroupLayout.PREFERRED_SIZE,                                         291,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel8Layout.createSequentialGroup()
                                                .addComponent(nrRelationshipLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, 67, Short.MAX_VALUE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlac
ement.UNRELATED)
                                        .addComponent(nearestRelativeRelationship,
javax.swing.GroupLayout.PREFERRED_SIZE,                                         291,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(407, 407, 407))
        );
        jPanel8Layout.setVerticalGroup(
            jPanel8Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel8Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(nrSurnameLabel)
                                        .addComponent(nearestRelativeSurname,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(nrNameLabel)
                                            .addComponent(nearestRelativeName,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(nrPhoneNumberLabel)
                                        .addComponent(nearestRelativePhoneNumber,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(jPanel8Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                    .addComponent(nrRelationshipLabel)
                                    .addComponent(nearestRelativeRelationship,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addContainerGap(313, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel8.TabConstraints.tabT
itle"), jPanel8); // NOI18N

        jPanel2.setName("jPanel2"); // NOI18N

        jScrollPane1.setName("jScrollPane1"); // NOI18N

        historyExamEvaluation.setColumns(20);
        historyExamEvaluation.setLineWrap(true);
        historyExamEvaluation.setRows(5);
        historyExamEvaluation.setName("historyExamEvaluation"); // NOI18N
            historyExamEvaluation.setPreferredSize(new  java.awt.Dimension(220,
60));
        jScrollPane1.setViewportView(historyExamEvaluation);

                            javax.swing.GroupLayout   jPanel2Layout   =   new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                                                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                                                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(197, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel2.TabConstraints.tabT
itle"), jPanel2); // NOI18N

        jPanel3.setName("jPanel3"); // NOI18N

        jScrollPane2.setName("jScrollPane2"); // NOI18N

        diceaseProgress.setColumns(20);
        diceaseProgress.setLineWrap(true);
        diceaseProgress.setRows(5);
        diceaseProgress.setName("diceaseProgress"); // NOI18N
        diceaseProgress.setPreferredSize(new java.awt.Dimension(220, 60));
        jScrollPane2.setViewportView(diceaseProgress);

                            javax.swing.GroupLayout   jPanel3Layout   =   new
javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
```

```
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(197, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel3.TabConstraints.tabT
itle"), jPanel3); // NOI18N

        jPanel4.setName("jPanel4"); // NOI18N

        jScrollPane3.setName("jScrollPane3"); // NOI18N

        jTextArea3.setColumns(20);
        jTextArea3.setLineWrap(true);
        jTextArea3.setRows(5);
             jTextArea3.setText(resourceMap.getString("jTextArea3.text")); //
NOI18N
        jTextArea3.setName("jTextArea3"); // NOI18N
        jTextArea3.setPreferredSize(new java.awt.Dimension(220, 60));
        jScrollPane3.setViewportView(jTextArea3);

                            javax.swing.GroupLayout   jPanel4Layout   =   new
javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(jScrollPane3,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE,                                 246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(197, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel4.TabConstraints.tabT
itle"), jPanel4); // NOI18N

        jPanel5.setName("jPanel5"); // NOI18N
```

```
        jScrollPane4.setName("jScrollPane4"); // NOI18N

        exitDiagnosis.setColumns(20);
        exitDiagnosis.setLineWrap(true);
        exitDiagnosis.setRows(5);
        exitDiagnosis.setName("exitDiagnosis"); // NOI18N
        exitDiagnosis.setPreferredSize(new java.awt.Dimension(220, 60));
        jScrollPane4.setViewportView(exitDiagnosis);

                            javax.swing.GroupLayout   jPanel5Layout   =   new
javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(jScrollPane4,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel5Layout.setVerticalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                                                  .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE,                                   246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(197, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel5.TabConstraints.tabT
itle"), jPanel5); // NOI18N

        jPanel6.setName("jPanel6"); // NOI18N

        JScrollPanel5.setName("JScrollPanel5"); // NOI18N

        treatmentIntervention.setColumns(20);
        treatmentIntervention.setLineWrap(true);
        treatmentIntervention.setRows(5);
        treatmentIntervention.setName("treatmentIntervention"); // NOI18N
            treatmentIntervention.setPreferredSize(new  java.awt.Dimension(220,
60));
        JScrollPanel5.setViewportView(treatmentIntervention);

                            javax.swing.GroupLayout   jPanel6Layout   =   new
javax.swing.GroupLayout(jPanel6);
        jPanel6.setLayout(jPanel6Layout);
        jPanel6Layout.setHorizontalGroup(
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()
                                                .addComponent(JScrollPanel5,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel6Layout.setVerticalGroup(
```

```
            jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel6Layout.createSequentialGroup()
                .addContainerGap()
                                                    .addComponent(JScrollPanel5,
javax.swing.GroupLayout.PREFERRED_SIZE,                                     246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(197, Short.MAX_VALUE))
        );

         jTabbedPane1.addTab(resourceMap.getString("jPanel6.TabConstraints.tabT
itle"), jPanel6); // NOI18N

        jPanel7.setName("jPanel7"); // NOI18N

        jLabel9.setText(resourceMap.getString("jLabel9.text")); // NOI18N
        jLabel9.setName("jLabel9"); // NOI18N

        jScrollPane6.setName("jScrollPane6"); // NOI18N

        ecg.setColumns(20);
        ecg.setRows(5);
        ecg.setName("ecg"); // NOI18N
        jScrollPane6.setViewportView(ecg);

        jLabel10.setText(resourceMap.getString("jLabel10.text")); // NOI18N
        jLabel10.setName("jLabel10"); // NOI18N

        jScrollPane8.setName("jScrollPane8"); // NOI18N

        radiography.setColumns(20);
        radiography.setRows(5);
        radiography.setName("radiography"); // NOI18N
        jScrollPane8.setViewportView(radiography);

        jLabel11.setText(resourceMap.getString("jLabel11.text")); // NOI18N
        jLabel11.setName("jLabel11"); // NOI18N

        jScrollPane9.setName("jScrollPane9"); // NOI18N

        otherExams.setColumns(20);
        otherExams.setRows(5);
        otherExams.setName("otherExams"); // NOI18N
        jScrollPane9.setViewportView(otherExams);

                            javax.swing.GroupLayout   jPanel7Layout   =   new
javax.swing.GroupLayout(jPanel7);
        jPanel7.setLayout(jPanel7Layout);
        jPanel7Layout.setHorizontalGroup(
            jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
            .addGroup(jPanel7Layout.createSequentialGroup()
                .addContainerGap()
                 .addGroup(jPanel7Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
                    .addComponent(jLabel10)
                    .addComponent(jLabel11)
                    .addComponent(jLabel9)
                                                    .addComponent(jScrollPane9,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
```

```
                                                    .addComponent(jScrollPane8,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                                                    .addComponent(jScrollPane6,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE))
                    .addContainerGap())
          );
          jPanel7Layout.setVerticalGroup(
               jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
              .addGroup(jPanel7Layout.createSequentialGroup()
                  .addContainerGap()
                  .addComponent(jLabel9)
                  .addGap(18, 18, 18)
                                                    .addComponent(jScrollPane6,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addGap(18, 18, 18)
                  .addComponent(jLabel10)
                  .addGap(18, 18, 18)
                                                    .addComponent(jScrollPane8,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addGap(18, 18, 18)
                  .addComponent(jLabel11)
                  .addGap(18, 18, 18)
                                                    .addComponent(jScrollPane9,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addContainerGap(74, Short.MAX_VALUE))
          );

           jTabbedPane1.addTab(resourceMap.getString("jPanel7.TabConstraints.tabT
itle"), jPanel7); // NOI18N

          jPanel10.setName("jPanel10"); // NOI18N
          jPanel10.setLayout(new java.awt.BorderLayout());

          jSplitPane1.setDividerLocation(290);
          jSplitPane1.setName("jSplitPane1"); // NOI18N

          jScrollPane5.setName("jScrollPane5"); // NOI18N

          jTable1.setModel(new javax.swing.table.DefaultTableModel(
              new Object [][] {
                  {"Αιματοκρίτης", null},
                  {"Λευκά", null},
                  {"Αιμοπετάλια", null},
                  {"ΤΚΕ", null},
                  {"Αιμοσφαιρίνη", null},
                  {"Πολ.", null},
                  {"Λεμφ.", null},
                  {"Μον.", null},
                  {"Ηωσ.", null},
                  {"ΔΕΚ", null}
              },
              new String [] {
                  "Πεδίο", "Τιμή"
              }
          ) {
              Class[] types = new Class [] {
```

```
                java.lang.String.class, java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, true
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jTable1.setName("jTable1"); // NOI18N
        jScrollPane5.setViewportView(jTable1);
         jTable1.getColumnModel().getColumn(0).setHeaderValue(resourceMap.getSt
ring("jTable1.columnModel.title0")); // NOI18N
         jTable1.getColumnModel().getColumn(1).setHeaderValue(resourceMap.getSt
ring("jTable1.columnModel.title1")); // NOI18N

        jSplitPane1.setLeftComponent(jScrollPane5);

        jScrollPane7.setName("jScrollPane7"); // NOI18N

        jTable2.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {"Σάκχαρο", null},
                {"Ουρία", null},
                {"Κρεατινίνη", null},
                {"K", null},
                {"Na", null},
                {"Ca", null},
                {"Χολερυθρίνη ολ.", null},
                {"Χολερυθρίνη αμ.", null},
                {"SGOT", null},
                {"SGPT", null},
                {"γ - GT", null},
                {"Αλκαλ. Φωσφατ.", null},
                {"CPK", null},
                {"LDH", null}
            },
            new String [] {
                "Πεδίο", "Τιμή"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, true
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
```

```
          jTable2.setName("jTable2"); // NOI18N
          jScrollPane7.setViewportView(jTable2);
           jTable2.getColumnModel().getColumn(0).setHeaderValue(resourceMap.getSt
ring("jTable1.columnModel.title0")); // NOI18N
           jTable2.getColumnModel().getColumn(1).setHeaderValue(resourceMap.getSt
ring("jTable1.columnModel.title1")); // NOI18N

          jSplitPane1.setRightComponent(jScrollPane7);

          jPanel10.add(jSplitPane1, java.awt.BorderLayout.CENTER);

           jTabbedPane1.addTab(resourceMap.getString("jPanel10.TabConstraints.tab
Title"), jPanel10); // NOI18N

          jPanel11.setName("jPanel11"); // NOI18N

          JScrollPane6.setName("JScrollPane6"); // NOI18N

          treatmentIntervention1.setColumns(20);
          treatmentIntervention1.setLineWrap(true);
          treatmentIntervention1.setRows(5);
          treatmentIntervention1.setName("treatmentIntervention1"); // NOI18N
             treatmentIntervention1.setPreferredSize(new java.awt.Dimension(220,
60));
          JScrollPane6.setViewportView(treatmentIntervention1);

                              javax.swing.GroupLayout   jPanel11Layout   =   new
javax.swing.GroupLayout(jPanel11);
          jPanel11.setLayout(jPanel11Layout);
          jPanel11Layout.setHorizontalGroup(
              jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
              .addGroup(jPanel11Layout.createSequentialGroup()
                  .addContainerGap()
                                                      .addComponent(JScrollPane6,
javax.swing.GroupLayout.DEFAULT_SIZE, 768, Short.MAX_VALUE)
                  .addContainerGap())
          );
          jPanel11Layout.setVerticalGroup(
              jPanel11Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
              .addGroup(jPanel11Layout.createSequentialGroup()
                  .addContainerGap()
                                                      .addComponent(JScrollPane6,
javax.swing.GroupLayout.PREFERRED_SIZE,                                       246,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addContainerGap(197, Short.MAX_VALUE))
          );

           jTabbedPane1.addTab(resourceMap.getString("jPanel11.TabConstraints.tab
Title"), jPanel11); // NOI18N

          mainPanel.add(jTabbedPane1);

          jPanel9.setMaximumSize(new java.awt.Dimension(250, 40));
          jPanel9.setMinimumSize(new java.awt.Dimension(200, 30));
          jPanel9.setName("jPanel9"); // NOI18N
          jPanel9.setPreferredSize(new java.awt.Dimension(248, 40));

                                        javax.swing.ActionMap     actionMap     =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
```

```
icalRecord.class).getContext().getActionMap(DesktopMedicalRecordView.class,
this);
          submitButton.setAction(actionMap.get("saveRecordChanges")); // NOI18N
            submitButton.setText(resourceMap.getString("submitButton.text")); //
NOI18N
           submitButton.setToolTipText(resourceMap.getString("submitButton.toolTi
pText")); // NOI18N
          submitButton.setMargin(new java.awt.Insets(2, 10, 2, 10));
          submitButton.setName("submitButton"); // NOI18N
          submitButton.setVerticalAlignment(javax.swing.SwingConstants.BOTTOM);
          jPanel9.add(submitButton);

          cancelButton.setAction(actionMap.get("undoRecordChanges")); // NOI18N
            cancelButton.setText(resourceMap.getString("cancelButton.text")); //
NOI18N
          cancelButton.setMargin(new java.awt.Insets(2, 10, 2, 10));
          cancelButton.setName("cancelButton"); // NOI18N
          jPanel9.add(cancelButton);

          mainPanel.add(jPanel9);

          menuBar.setMaximumSize(new java.awt.Dimension(800, 20));
          menuBar.setMinimumSize(new java.awt.Dimension(100, 20));
          menuBar.setName("menuBar"); // NOI18N
          menuBar.setPreferredSize(new java.awt.Dimension(100, 20));

          fileMenu.setText(resourceMap.getString("fileMenu.text")); // NOI18N
          fileMenu.setName("fileMenu"); // NOI18N

          exitMenuItem.setAction(actionMap.get("quit")); // NOI18N
            exitMenuItem.setText(resourceMap.getString("exitMenuItem.text")); //
NOI18N
           exitMenuItem.setToolTipText(resourceMap.getString("exitMenuItem.toolTi
pText")); // NOI18N
          exitMenuItem.setName("exitMenuItem"); // NOI18N
          fileMenu.add(exitMenuItem);

          menuBar.add(fileMenu);

           medicalRecordMenu.setText(resourceMap.getString("medicalRecordMenu.tex
t")); // NOI18N
          medicalRecordMenu.setName("medicalRecordMenu"); // NOI18N

           getRecordMenuItem.setAction(actionMap.get("getRecordViaBluetooth")); /
/ NOI18N
           getRecordMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(ja
va.awt.event.KeyEvent.VK_G, java.awt.event.InputEvent.CTRL_MASK));
           getRecordMenuItem.setText(resourceMap.getString("getRecordMenuItem.tex
t")); // NOI18N
          getRecordMenuItem.setName("getRecordMenuItem"); // NOI18N
          medicalRecordMenu.add(getRecordMenuItem);

           putRecordMenuItem.setAction(actionMap.get("puRecordViaBluetooth")); //
NOI18N
           putRecordMenuItem.setText(resourceMap.getString("putRecordMenuItem.tex
t")); // NOI18N
          putRecordMenuItem.setName("putRecordMenuItem"); // NOI18N
          medicalRecordMenu.add(putRecordMenuItem);

          menuBar.add(medicalRecordMenu);
```

```
        helpMenu.setText(resourceMap.getString("helpMenu.text")); // NOI18N
        helpMenu.setName("helpMenu"); // NOI18N

        aboutMenuItem.setAction(actionMap.get("showAboutBox")); // NOI18N
         aboutMenuItem.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.a
wt.event.KeyEvent.VK_B, java.awt.event.InputEvent.CTRL_MASK));
         aboutMenuItem.setText(resourceMap.getString("aboutMenuItem.text")); //
NOI18N
        aboutMenuItem.setName("aboutMenuItem"); // NOI18N
        helpMenu.add(aboutMenuItem);

        menuBar.add(helpMenu);

        statusPanel.setMaximumSize(new java.awt.Dimension(800, 30));
        statusPanel.setMinimumSize(new java.awt.Dimension(100, 30));
        statusPanel.setName("statusPanel"); // NOI18N
        statusPanel.setPreferredSize(new java.awt.Dimension(100, 30));

        statusPanelSeparator.setName("statusPanelSeparator"); // NOI18N

        statusMessageLabel.setName("statusMessageLabel"); // NOI18N

         statusAnimationLabel.setHorizontalAlignment(javax.swing.SwingConstants
.LEFT);
        statusAnimationLabel.setName("statusAnimationLabel"); // NOI18N

        progressBar.setName("progressBar"); // NOI18N

                        javax.swing.GroupLayout   statusPanelLayout   =   new
javax.swing.GroupLayout(statusPanel);
        statusPanel.setLayout(statusPanelLayout);
        statusPanelLayout.setHorizontalGroup(
            statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                                            .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.DEFAULT_SIZE, 797, Short.MAX_VALUE)
            .addGroup(statusPanelLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(statusMessageLabel)
                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, 613, Short.MAX_VALUE)
                                                        .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                .addComponent(statusAnimationLabel)
                .addContainerGap())
        );
        statusPanelLayout.setVerticalGroup(
            statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
            .addGroup(statusPanelLayout.createSequentialGroup()
                                            .addComponent(statusPanelSeparator,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    2,
javax.swing.GroupLayout.PREFERRED_SIZE)
                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED, 11, Short.MAX_VALUE)
                 .addGroup(statusPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.BASELINE)
                     .addComponent(statusMessageLabel)
```

```
                    .addComponent(statusAnimationLabel)
                                                .addComponent(progressBar,
javax.swing.GroupLayout.PREFERRED_SIZE,    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(3, 3, 3))
        );

        setComponent(mainPanel);
        setMenuBar(menuBar);
        setStatusBar(statusPanel);
        addPropertyChangeListener(new java.beans.PropertyChangeListener() {
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                formPropertyChange(evt);
            }
        });
    }// </editor-fold>

    private void formPropertyChange(java.beans.PropertyChangeEvent evt) {
        // TODO add your handling code here:
    }
    // Variables declaration - do not modify
    private javax.swing.JScrollPane JScrollPane15;
    private javax.swing.JScrollPane JScrollPane16;
    private javax.swing.JMenuItem aboutMenuItem;
    private javax.swing.JTextField address;
    private javax.swing.JLabel addressLabel;
    private javax.swing.JTextField amka;
    private javax.swing.JLabel amkaLabel;
    private javax.swing.JTextField bloodTypeRh;
    private javax.swing.JLabel bloodTypeRhLabel;
    private javax.swing.JButton cancelButton;
    private javax.swing.JTextField city;
    private javax.swing.JLabel cityLabel;
    private javax.swing.JTextArea diceaseProgress;
    private javax.swing.JTextArea ecg;
    private javax.swing.JTextArea exitDiagnosis;
    private javax.swing.JMenuItem getRecordMenuItem;
    private javax.swing.JTextArea historyExamEvaluation;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel11;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JPanel jPanel6;
    private javax.swing.JPanel jPanel7;
    private javax.swing.JPanel jPanel8;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JScrollPane jScrollPane7;
    private javax.swing.JScrollPane jScrollPane8;
    private javax.swing.JScrollPane jScrollPane9;
```

```
private javax.swing.JSplitPane jSplitPane1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTable jTable2;
private javax.swing.JTextArea jTextArea3;
private javax.swing.JPanel mainPanel;
private javax.swing.JMenu medicalRecordMenu;
private javax.swing.JMenuBar menuBar;
private javax.swing.JTextField name;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nearestRelativeName;
private javax.swing.JTextField nearestRelativePhoneNumber;
private javax.swing.JTextField nearestRelativeRelationship;
private javax.swing.JTextField nearestRelativeSurname;
private javax.swing.JLabel nrNameLabel;
private javax.swing.JLabel nrPhoneNumberLabel;
private javax.swing.JLabel nrRelationshipLabel;
private javax.swing.JLabel nrSurnameLabel;
private javax.swing.JTextArea otherExams;
private javax.swing.JTextField phoneNumber;
private javax.swing.JLabel phoneNumberLabel;
private javax.swing.JTextField postalCode;
private javax.swing.JLabel postalCodeLabel;
private javax.swing.JProgressBar progressBar;
private javax.swing.JMenuItem putRecordMenuItem;
private javax.swing.JTextArea radiography;
private javax.swing.JLabel statusAnimationLabel;
private javax.swing.JLabel statusMessageLabel;
private javax.swing.JPanel statusPanel;
private javax.swing.JButton submitButton;
private javax.swing.JTextField surname;
private javax.swing.JLabel surnameLabel;
private javax.swing.JTextArea treatmentIntervention;
private javax.swing.JTextArea treatmentIntervention1;
// End of variables declaration
private final Timer messageTimer;
private final Timer busyIconTimer;
private final Icon idleIcon;
private final Icon[] busyIcons = new Icon[15];
private int busyIconIndex = 0;
private JDialog aboutBox;
private MedicalRecord medicalRecord;
private EmergencyRecord emergencyRecord;
private String connUsername = "username";
private String connPassword;

public MedicalRecord getMedicalRecord() {
    return medicalRecord;
}

public void setMedicalRecord(MedicalRecord medicalRecord) {
    this.medicalRecord = medicalRecord;
}

public EmergencyRecord getEmergencyRecord() {
    return emergencyRecord;
}

public void setEmergencyRecord(EmergencyRecord emergencyRecord) {
    this.emergencyRecord = emergencyRecord;
}
```

```
    public String getConnUsername() {
        return connUsername;
    }

    public void setConnUsername(String connUsername) {
        this.connUsername = connUsername;
    }

    public String getConnPassword() {
        return connPassword;
    }

    public void setConnPassword(String connPassword) {
        this.connPassword = connPassword;
    }

    @Action
    public void saveRecordChanges() {
        Class thisClass = this.getClass();
        for (Object key : getMedicalRecord().getDataTable().keySet()) {
            try {
                Field f = thisClass.getDeclaredField((String) key);
                if (f.get(this) instanceof JTextComponent) {
                                    getMedicalRecord().getDataTable().put(key,
((JTextComponent) f.get(this)).getText());
                } else if (f.get(this) instanceof JTable) {
                    JTable jt = (JTable) f.get(this);
                    TableColumnModel columnModel = jt.getColumnModel();
                    TableColumn column = jt.getColumn(1);
 //                   jt.getModel().
                }
            } catch (NoSuchFieldException ex) {
                    System.err.println("===========NoSuchFieldException: " +
(String) key);
            } catch (Exception ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            }
        }
    }

    @Action
    public void undoRecordChanges() {
        Class thisClass = this.getClass();
        for (Object key : getMedicalRecord().getDataTable().keySet()) {
            try {
                Field f = thisClass.getDeclaredField((String) key);
                JTextComponent jtc = (JTextComponent) f.get(this);
                                                        jtc.setText((String)
getMedicalRecord().getDataTable().get(key));
                f.set(this, jtc);
            } catch (NoSuchFieldException ex) {
                    System.err.println("===========NoSuchFieldException: " +
(String) key);
            } catch (Exception ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            }
        }
    }
```

```
      /**
       *
       * @deprecated
       */
      @Deprecated
      public void saveRecord() {
  //        System.out.println("BEFORE");
  //        getMedicalRecord().print();
  //        System.out.println("BEFORE");
          Class thisClass = this.getClass();
          for (Object key : getMedicalRecord().getDataTable().keySet()) {
  //            if (getResourceMap().containsKey((String) key + ".text")) {
  //                                getMedicalRecord().getH().put((String)  key,
getResourceMap().getString((String) key + ".text"));
  //                System.out.println("        key: " + (String) key);
  //                        System.out.println("ResourceKey: " + (String)  key +
".text");
  //                                System.out.println("            value:  "  +
getResourceMap().getString((String) key + ".text"));
  //            }
              try {
                  Field f = thisClass.getDeclaredField((String) key);
  //                Object fieldO = f.getType();
  //                fieldO = f.getMedicalRecord(this);
  //                        getMedicalRecord().getH().put(key,  ((JTextComponent)
fieldO).getText());
                  JTextComponent jtc = (JTextComponent) f.get(this);
                  getMedicalRecord().getDataTable().put(key, jtc.getText());
  //                System.out.println("Field name = " + f.getName());
  //                if (f.getName().equals((String) key)) {
  //                    System.out.println("Field : " + f.toGenericString());
  //                                    System.out.println("Field  class:  "  +
f.getClass().getName());
  //                    Class<?> c = f.getType();
  //                    System.out.println("Type class :" + c.getName());
  //                    Method m = c.getMethod("getText"); //, (Class[]) null);
  //                                System.out.println("Type  class  Method  :"  +
m.toGenericString());
  //                    String result = (String) m.invoke(this, null);
  //                     System.out.println("!!!!!!!!! field=" + f.getName() + ",
value=" + result);
  //                     System.out.println("!!!!!!!!! field=" + f.getName() + ",
value=" + component.getText());
  //                    if (f.getName().equals("amka"))
  //                                    System.out.println("!!!!!!!!!  AMKA="  +
amka.getText());
  //                }
              } catch (NoSuchFieldException ex) {
                      System.err.println("===========NoSuchFieldException:  "  +
(String) key);
  //                ex.printStackTrace();
  //
Logger.getLogger(DesktopMedicalRecordView.class.getName()).log(Level.WARNING,
"Field = " + (String) key, ex); //"NoSuchFieldException");
  //            } catch (InvocationTargetException ex) {
  //                                System.err.println("|||||||||||||||||||||||||
InvocationTargetException: " + (String) key);
  //                ex.printStackTrace();
```

```
  //
Logger.getLogger(DesktopMedicalRecordView.class.getName()).log(Level.SEVERE,
null, ex);
              } catch (Exception ex) {
  //                  System.err.println("------------------------Exception: " +
(String) key);
  //                ex.printStackTrace();
                   Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
              }
          }
  //        System.out.println("AFTER");
  //        getMedicalRecord().print();
  //        System.out.println("AFTER");
      }

      @Action
      public void showDialogBox() {
          if (dialogBox == null) {
                                                        JFrame    mainFrame    =
DesktopMedicalRecord.getApplication().getMainFrame();
                dialogBox = new DesktopMedicalRecordConnecitonDialog(mainFrame,
this, true);
              dialogBox.setLocationRelativeTo(mainFrame);
          }
          DesktopMedicalRecord.getApplication().show(dialogBox);
      }

      @Action
      public Task getRecordViaBluetooth() {
          System.out.println("getRecordViaBluetooth");
  //        if (connPassword == null) {
          showDialogBox();
  //        }
          obexclient = new OBEXClient();
          return new GetRecordViaBluetoothTask(getApplication());
      }

                      private     class     GetRecordViaBluetoothTask     extends
org.jdesktop.application.Task<Object, Void> {

          GetRecordViaBluetoothTask(org.jdesktop.application.Application app) {
              // Runs on the EDT.  Copy GUI state that
              // doInBackground() depends on from parameters
              // to GetRecordViaBluetoothTask fields, here.
              super(app);
          }

          @Override
          protected Hashtable<String, byte[]> doInBackground() {
              Hashtable<String, byte[]> result = null;
              byte[] medicalRecordData = null;
              byte[] emergencyRecordData = null;
              try {
                  // Your Task's code here.  This method runs
                  // the Swing GUI from here.
                  System.out.println("connUsername=  " + connUsername);
                  System.out.println("connPassword=  " + connPassword);
                  obexclient.connectToObexServer();
                     medicalRecordData = obexclient.get(connUsername.getBytes(),
connPassword.getBytes(), MedicalRecord.RECORD_DATA_TYPE);
```

```
                    emergencyRecordData = obexclient.get(connUsername.getBytes(),
connPassword.getBytes(), EmergencyRecord.RECORD_DATA_TYPE);
                obexclient.closeConnection();
            } catch (BluetoothStateException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            } catch (InterruptedException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            }
            result = new Hashtable<String, byte[]>();
            result.put(MedicalRecord.RECORD_DATA_TYPE, medicalRecordData);
            result.put(EmergencyRecord.RECORD_DATA_TYPE, emergencyRecordData);
            MedicalRecord mr = new MedicalRecord(null);
            EmergencyRecord er = new EmergencyRecord(null);
            mr.fromByteArray(medicalRecordData);
            setMedicalRecord(mr);
            er.fromByteArray(emergencyRecordData);
            setEmergencyRecord(er);

            return result;
        }

        @Override
        protected void succeeded(Object result) {
            // Runs on the EDT.  Update the GUI based on
            // the result computed by doInBackground().
            if (result != null) {
                System.out.println("get success");
                if (result instanceof Hashtable) {
                    System.out.println("hashtable");
                }
            } else {
                System.out.println("get fail");
            }
        }
    }

    @Action
    public Task puRecordViaBluetooth() {
        System.out.println("puRecordViaBluetooth");
        obexclient = new OBEXClient();
        return new PutRecordViaBluetoothTask(getApplication());
    }

                    private     class     PutRecordViaBluetoothTask     extends
org.jdesktop.application.Task<Object, Void> {

        PutRecordViaBluetoothTask(org.jdesktop.application.Application app) {
            // Runs on the EDT.  Copy GUI state that
            // doInBackground() depends on from parameters
            // to GetRecordViaBluetoothTask fields, here.
            super(app);
        }

        @Override
        protected Object doInBackground() {
            // Your Task's code here.  This method runs
```

```
            // the Swing GUI from here.
            String result = null;
            try {
                obexclient.connectToObexServer();
                                        obexclient.put(connUsername.getBytes(),
connPassword.getBytes(),                      getMedicalRecord().toByteArray(),
MedicalRecord.RECORD_DATA_TYPE);
                                        obexclient.put(connUsername.getBytes(),
connPassword.getBytes(),                    getEmergencyRecord().toByteArray(),
EmergencyRecord.RECORD_DATA_TYPE);
                obexclient.closeConnection();
                result = "SUCCESS";

            } catch (BluetoothStateException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            } catch (InterruptedException ex) {
                Logger.getLogger(DesktopMedicalRecordView.class.getName()).log
(Level.SEVERE, null, ex);
            }
            return result; // return your result
        }

        @Override
        protected void succeeded(Object result) {
            // Runs on the EDT.  Update the GUI based on
            // the result computed by doInBackground().
            if (result != null) {
                System.out.println("put success");
            } else {
                System.out.println("put fail");
            }
        }
    }
}
```

```
  /*
   * DesktopMedicalRecordConnecitonDialog.java
   */
  package desktopmedicalrecord;

  import org.jdesktop.application.Action;

  /**
   *
   * @author panayiotis
   */
  public class DesktopMedicalRecordConnecitonDialog extends javax.swing.JDialog
{

      private DesktopMedicalRecordView parent;

      /** Creates new form DesktopMedicalRecordConnecitonDialog */
       public DesktopMedicalRecordConnecitonDialog(javax.swing.JFrame mainFrame,
DesktopMedicalRecordView parent, boolean modal) {
          super(mainFrame, modal);
          this.parent = parent;
          initComponents();
      }

      /** This method is called from within the constructor to
       * initialize the form.
       * WARNING: Do NOT modify this code. The content of this method is
       * always regenerated by the Form Editor.
       */
      @SuppressWarnings("unchecked")
      // <editor-fold defaultstate="collapsed" desc="Generated Code">
      private void initComponents() {

          jPanel1 = new javax.swing.JPanel();
          connPasswordTextField = new javax.swing.JTextField();
          jScrollPane1 = new javax.swing.JScrollPane();
          jTextPane1 = new javax.swing.JTextPane();
          connectButton = new javax.swing.JButton();
          cancelButton = new javax.swing.JButton();

          setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE)
;
          setAlwaysOnTop(true);
                           org.jdesktop.application.ResourceMap   resourceMap   =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
icalRecord.class).getContext().getResourceMap(DesktopMedicalRecordConnecitonDial
og.class);
                setBackground(resourceMap.getColor("connDialog.background"));  //
NOI18N
          setModal(true);
          setName("connDialog"); // NOI18N
          setResizable(false);

          jPanel1.setName("jPanel1"); // NOI18N

          connPasswordTextField.setName("connPasswordTextField"); // NOI18N

          jScrollPane1.setName("jScrollPane1"); // NOI18N

          jTextPane1.setBackground(resourceMap.getColor("jTextPane1.background")
); // NOI18N
```

```
          jTextPane1.setBorder(null);
          jTextPane1.setEditable(false);
                jTextPane1.setText(resourceMap.getString("jTextPane1.text"));  //
NOI18N
                                                              jTextPane1.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
          jTextPane1.setFocusCycleRoot(false);
          jTextPane1.setFocusable(false);
          jTextPane1.setName("jTextPane1"); // NOI18N
          jScrollPane1.setViewportView(jTextPane1);

                                           javax.swing.ActionMap     actionMap     =
org.jdesktop.application.Application.getInstance(desktopmedicalrecord.DesktopMed
icalRecord.class).getContext().getActionMap(DesktopMedicalRecordConnecitonDialog
.class, this);
          connectButton.setAction(actionMap.get("connect")); // NOI18N
           connectButton.setText(resourceMap.getString("connectButton.text")); //
NOI18N
          connectButton.setName("connectButton"); // NOI18N
          connectButton.setPreferredSize(new java.awt.Dimension(70, 27));

          cancelButton.setAction(actionMap.get("cancel")); // NOI18N
            cancelButton.setText(resourceMap.getString("cancelButton.text")); //
NOI18N
          cancelButton.setName("cancelButton"); // NOI18N
          cancelButton.setPreferredSize(new java.awt.Dimension(70, 27));

                                javax.swing.GroupLayout    jPanel1Layout    =    new
javax.swing.GroupLayout(jPanel1);
          jPanel1.setLayout(jPanel1Layout);
          jPanel1Layout.setHorizontalGroup(
               jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
              .addGroup(jPanel1Layout.createSequentialGroup()
                  .addContainerGap()
                   .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                                .addComponent(connPasswordTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,                                          89,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                 .addGap(122, 122, 122))
                           .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
                                                          .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 330, Short.MAX_VALUE)
                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                                        .addComponent(connectButton,
javax.swing.GroupLayout.PREFERRED_SIZE,                                        106,
javax.swing.GroupLayout.PREFERRED_SIZE)
                             .addGap(40, 40, 40)
                                                          .addComponent(cancelButton,
javax.swing.GroupLayout.PREFERRED_SIZE,                                          97,
javax.swing.GroupLayout.PREFERRED_SIZE)
                             .addGap(52, 52, 52)))))
          );
          jPanel1Layout.setVerticalGroup(
               jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
```

```
            .addGroup(jPanel1Layout.createSequentialGroup()
                                                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,                                         45,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                                                .addComponent(connPasswordTextField,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UN
RELATED)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.BASELINE)
                                                    .addComponent(cancelButton,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(connectButton,
javax.swing.GroupLayout.PREFERRED_SIZE,     javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addContainerGap(31, Short.MAX_VALUE))
          );

                                    javax.swing.GroupLayout    layout    =    new
javax.swing.GroupLayout(getContentPane());
          getContentPane().setLayout(layout);
          layout.setHorizontalGroup(
               layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
               .addGroup(layout.createSequentialGroup()
                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())
          );
          layout.setVerticalGroup(
               layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                     .addComponent(jPanel1,  javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
          );

          pack();
      }// </editor-fold>
      // Variables declaration - do not modify
      private javax.swing.JButton cancelButton;
      private javax.swing.JTextField connPasswordTextField;
      private javax.swing.JButton connectButton;
      private javax.swing.JPanel jPanel1;
      private javax.swing.JScrollPane jScrollPane1;
      private javax.swing.JTextPane jTextPane1;
      // End of variables declaration

      @Action
      public void cancel() {
          dispose();
      }

      @Action
      public void connect() {
              parent.setConnPassword((connPasswordTextField.getText()  !=  null  ?
connPasswordTextField.getText() : ""));
          dispose();
```

```
        }
    }
```

```
/*
 * OBEXClient.java
 */
package obex;

import java.io.ByteArrayOutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.bluetooth.BluetoothStateException;
import javax.obex.Authenticator;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.microedition.io.Connector;
import javax.obex.ClientSession;
import javax.obex.HeaderSet;
import javax.obex.Operation;
import javax.obex.PasswordAuthentication;
import javax.obex.ResponseCodes;

public class OBEXClient {

    private byte[] username;
    private byte[] password;
    private ClientSession clientSession;
    private HeaderSet hsOperation;

    public byte[] get(byte[] username, byte[] password, String objName) throws
IOException, BluetoothStateException, InterruptedException {
        System.out.println("username = " + new String(username));
        System.out.println("password = " + new String(password));
        System.out.println("objName = " + objName);

//        connectToObexServer();

        System.out.println("getObject ...");

        // Send the initial GET request to the server
        hsOperation.setHeader(HeaderSet.NAME, objName);

        Operation op = clientSession.get(hsOperation);

        // Get the object from the input stream
        InputStream in = op.openInputStream();

        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int readByte;
        while ((readByte = in.read()) != -1) {
            out.write((byte) readByte);
        }

        // End the transaction
        in.close();
        op.close();

        byte[] obj = out.toByteArray();
        out.flush();
        out.close();

        if (obj != null) {
            System.out.println("object length = " + obj.length);
```

```
        }

        System.out.println("object trimmed : " + new String(obj).trim());
        System.out.println("object original: " + new String(obj));
        System.out.println("getObject done");

        return obj;
    }

      public void put(byte[] username, byte[] password, byte[] obj, String
objName) throws IOException, BluetoothStateException, InterruptedException {
        System.out.println("username = " + new String(username));
        System.out.println("password = " + new String(password));
        System.out.println("objName = " + objName);

//        connectToObexServer();

        System.out.println("sendtObject ...");

        // Include the length header
        hsOperation.setHeader(HeaderSet.NAME, objName);
        hsOperation.setHeader(HeaderSet.LENGTH, new Long(obj.length));

        // Initiate the PUT request
        Operation op = clientSession.put(hsOperation);

        // Open the output stream to put the object to it
        OutputStream out = op.openOutputStream();

        // Send the object to the server
        out.write(obj);
        out.flush();

        // End the transaction
        out.close();
        op.close();
    }

          public void connectToObexServer() throws InterruptedException,
BluetoothStateException, IOException {
        String serverURL;
        String[] searchArgs = null;

        ServicesSearch.main(searchArgs);
        if (ServicesSearch.serviceFound.size() == 0) {
            System.out.println("OBEX service not found");
            throw new IOException("No service found");
        }

        serverURL = (String) ServicesSearch.serviceFound.elementAt(0);
        System.out.println("\n========== CONNECT ==========");
        System.out.println("Connecting to " + serverURL);

        clientSession = (ClientSession) Connector.open(serverURL);
        HeaderSet hsRequest = clientSession.createHeaderSet();
//              hsRequest.createAuthenticationChallenge("MedicalRecord", true,
true);
        ConnectionAuthenticator authenticator = new ConnectionAuthenticator();
        clientSession.setAuthenticator(authenticator);
        HeaderSet hsConnReply = clientSession.connect(hsRequest);
```

```
            switch (hsConnReply.getResponseCode()) {
                case ResponseCodes.OBEX_HTTP_OK:
                    System.out.println("OBEX_HTTP_OK");
                    break;
                case ResponseCodes.OBEX_HTTP_UNAUTHORIZED:
                    System.out.println("OBEX_HTTP_UNAUTHORIZED");
                    throw new IOException("OBEX_HTTP_UNAUTHORIZED");
                default:
                    System.out.println("Failed to connect");
                    throw new IOException("Failed to connect");
            }

            System.out.println("connected");

            hsOperation = clientSession.createHeaderSet();
            hsOperation.setHeader(HeaderSet.COUNT, new Long(1));
                    hsOperation.setHeader(HeaderSet.DESCRIPTION,  "Medical  Record
Exchange");
  //        hsOperation.setHeader(HeaderSet.NAME, "MedicalRecord");
            hsOperation.setHeader(HeaderSet.TYPE, "binary");

            System.out.println("client 1");
        }

        public void closeConnection() {
            try {
                clientSession.close();
            } catch (IOException ex) {
                    Logger.getLogger(OBEXClient.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }

        private class ConnectionAuthenticator implements Authenticator {

            public ConnectionAuthenticator() {
            }

            @Override
                  public  PasswordAuthentication  onAuthenticationChallenge(String
description, boolean isUserIdRequired, boolean isFullAccess) {
                System.out.println("onAuthenticationChallenge");
                System.out.println("description: " + description);
                return new PasswordAuthentication(username, password);
            }

            @Override
            public byte[] onAuthenticationResponse(byte[] userName) {
                System.out.println("onAuthenticationResponse");
                System.out.println("userName = " + new String(userName));
                return password;
            }
        }
    }
```

```
/*
 * ServicesSearch
 */
package obex;

import java.io.IOException;
import java.util.Enumeration;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DataElement;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;

/**
 *
 * Bluetooth Device and Services Discovery class.
 */
public class ServicesSearch implements DiscoveryListener {

    private static final UUID MYOBEX_OBJECT_PUSH = new UUID(0x1105);
    private static final int MOBILE_MAJOR_DEVICE = 0x0200;
    private static final int SERVICE_NAME_ATTRIBUTE = 0x0100;
    private static final String SERVICE_NAME = "MedicalRecord";
    private static final Object inquiryCompletedEvent = new Object();
    private static final Object serviceSearchCompletedEvent = new Object();
        public static final Vector<RemoteDevice> devicesDiscovered = new
Vector<RemoteDevice>();
    public static final Vector<String> serviceFound = new Vector<String>();

       public static void main(String[] args) throws InterruptedException,
BluetoothStateException {
        ServicesSearch servicesSearch = new ServicesSearch();

        devicesDiscovered.clear();
        serviceFound.clear();

        // Search for Devices
        System.out.println("========== DEVICE SEARCH ==========");
        synchronized (inquiryCompletedEvent) {
                                                    boolean started =
LocalDevice.getLocalDevice().getDiscoveryAgent().startInquiry(DiscoveryAgent.GIA
C, servicesSearch);
                    System.out.println("Local Device Bluetooth Address: " +
LocalDevice.getLocalDevice().getBluetoothAddress());
                                System.out.println("Local Devive Name: " +
LocalDevice.getLocalDevice().getFriendlyName());
            if (started) {
                System.out.println("wait for device inquiry to complete...");
                inquiryCompletedEvent.wait();
                     System.out.println(devicesDiscovered.size() + " device(s)
found");
            }
        }
        UUID serviceUUID = MYOBEX_OBJECT_PUSH;
```

```
        if ((args != null) && (args.length > 0)) {
            serviceUUID = new UUID(args[0], false);
        }

        UUID[] searchUuidSet = new UUID[]{serviceUUID};
        int[] attrIDs = new int[]{ServicesSearch.SERVICE_NAME_ATTRIBUTE};

        // Search for services
        System.out.println("\n========== SERVICE SEARCH ==========");
            for (Enumeration<RemoteDevice> en = devicesDiscovered.elements();
en.hasMoreElements();) {
  //            serviceFound.clear();
            RemoteDevice btDevice = en.nextElement();
            synchronized (serviceSearchCompletedEvent) {
                try {
                                    System.out.println("search services on " +
btDevice.getFriendlyName(false) + " (" + btDevice.getBluetoothAddress() + ")");
                } catch (IOException ex) {
                    Logger.getLogger(ServicesSearch.class.getName()).log(Level
.SEVERE, null, ex);
                }
                 LocalDevice.getLocalDevice().getDiscoveryAgent().searchService
s(attrIDs, searchUuidSet, btDevice, servicesSearch);
  //
LocalDevice.getLocalDevsice().getDiscoveryAgent().searchServices(null,
searchUuidSet, btDevice, servicesSearch);
                serviceSearchCompletedEvent.wait();
                        System.out.println(serviceFound.size() + " service(s)
found.");
            }
        }

    }

    @Override
    public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
            System.out.println("Device " + btDevice.getBluetoothAddress() + "
found");
        try {
            System.out.println("\tname = " + btDevice.getFriendlyName(false));
        } catch (IOException iOException) {
        }
                        System.out.println("\tMajor Device Class = " +
Integer.toHexString(cod.getMajorDeviceClass()));
                        System.out.println("\tMinor Device Class = " +
Integer.toHexString(cod.getMinorDeviceClass()));
                           System.out.println("\tService Classes = " +
Integer.toHexString(cod.getServiceClasses()));
        if (cod.getMajorDeviceClass() == ServicesSearch.MOBILE_MAJOR_DEVICE) {
            if (!devicesDiscovered.contains(btDevice)) {
                devicesDiscovered.addElement(btDevice);
            }
        }
    }

    @Override
    public void inquiryCompleted(int discType) {
        synchronized (inquiryCompletedEvent) {
            inquiryCompletedEvent.notifyAll();
        }
        switch (discType) {
```

```
            case DiscoveryListener.INQUIRY_COMPLETED:
                System.out.println("INQUIRY_COMPLETED");
                break;
            case DiscoveryListener.INQUIRY_TERMINATED:
                System.out.println("INQUIRY_TERMINATED");
                break;
            case DiscoveryListener.INQUIRY_ERROR:
                System.out.println("INQUIRY_ERROR");
                break;
            default:
                System.out.println("Unknown Response Code");
        }
        System.out.println("Device Inquiry completed!");
    }

    @Override
    public void servicesDiscovered(int transID, ServiceRecord[] servRecord) {
        System.out.println("service found!");
        for (int i = 0; i < servRecord.length; i++) {
                                                        String     url     =
servRecord[i].getConnectionURL(ServiceRecord.AUTHENTICATE_ENCRYPT, false);
            if (url == null) {
                continue;
            }

  //            for (int j = 111; j < servRecord[i].getAttributeIDs().length; j+
+) {
  //                int k = servRecord[i].getAttributeIDs()[j];
  //                switch (k) {
  //                    case 0x0000:
  //
System.out.println("==ServiceRecordHandle:\n\tattrID=0x"                          +
Integer.toHexString(k) + ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    case 0x0001:
  //
System.out.println("==ServiceClassIDList:\n\tattrID=0x" + Integer.toHexString(k)
+ ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    case 0x0002:
  //
System.out.println("==ServiceRecordState:\n\tattrID=0x" + Integer.toHexString(k)
+ ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    case 0x0003:
  //                            System.out.println("==ServiceID:\n\tattrID=0x" +
Integer.toHexString(k) + ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    case 0x0004:
  //
System.out.println("==ProtocolDescriptorList:\n\tattrID=0x"                       +
Integer.toHexString(k) + ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    case 0x100:
  //                         System.out.println("==Service Name:\n\tattrID=0x" +
Integer.toHexString(k) + ", value=" + servRecord[i].getAttributeValue(k));
  //                        break;
  //                    default:
  //                         System.out.println("Unknown Attribute:  attrID=0x" +
Integer.toHexString(k) + ", value=" + servRecord[i].getAttributeValue(k));
  //                }
```

```
  //            }

                                                    DataElement    serviceName    =
servRecord[i].getAttributeValue(ServicesSearch.SERVICE_NAME_ATTRIBUTE);
                                    if   (serviceName   !=   null   &&   ((String)
serviceName.getValue()).trim().equals(ServicesSearch.SERVICE_NAME)) {
                    serviceFound.addElement(url);
                }
                if (serviceName != null) {
                      System.out.println("service '" + serviceName.getValue() + "'
found on" + url);
                } else {
                    System.out.println("service found " + url);
                }
            }
        }

        @Override
        public void serviceSearchCompleted(int transID, int respCode) {
            synchronized (serviceSearchCompletedEvent) {
                serviceSearchCompletedEvent.notifyAll();
            }
            System.out.println("transID = " + Integer.toHexString(transID));
            switch (respCode) {
                case DiscoveryListener.SERVICE_SEARCH_COMPLETED:
                    System.out.println("SERVICE_SEARCH_COMPLETED");
                    break;
                case DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
                    System.out.println("SERVICE_SEARCH_DEVICE_NOT_REACHABLE");
                    break;
                case DiscoveryListener.SERVICE_SEARCH_ERROR:
                    System.out.println("SERVICE_SEARCH_ERROR");
                    break;
                case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:
                    System.out.println("SERVICE_SEARCH_NO_RECORDS");
                    break;
                case DiscoveryListener.SERVICE_SEARCH_TERMINATED:
                    System.out.println("SERVICE_SEARCH_TERMINATED");
                    break;
                default:
                    System.out.println("Unknown response code.");
            }
            System.out.println("service search completed!");
        }
    }
```

```
#
# DesktopMedicalRecord.properties
#
# Application global resources

Application.name = DesktopMedicalRecord
Application.title = Ψηφιακός Ιατρικός Φακέλος
Application.version = 1.0
Application.vendor = Παναγίωτης Τεμπριώτης
Application.homepage =
Application.description = Η εφαρμογή χρησιμοποιείται σε συνδυασμό με
αντίστοιχη εφαρμογή στο κινητό τηλέφωνο για την ανάγνωση και τροποποίηση
ιατρικού φακέλου.
Application.vendorId = NTUA
Application.id = DesktopAppliDesktopMedicalRecord
lookAndFeel = system
quit.Action.text=Exit
quit.Action.accelerator=ctrl pressed Q
quit.Action.shortDescription=Exit the application
#NOI18N
Application.lookAndFeel=
```

```
#
# DesktopMedicalRecordAboutBox.properties
#
title = About: ${Application.title} ${Application.version}

closeAboutBox.Action.text = &Close

appDescLabel.text=<html>${Application.description}

versionLabel.text=Έκδοση\:

vendorLabel.text=Πωλητής\:

homepageLabel.text=Ιστοσελίδα\:
#NOI18N
imageLabel.icon=about.jpg
closeButton.text=Κλείσιμο
```

```
#
# DesktopMedicalRecordConnecitonDialog.properties
#
#NOI18N
connDialog.background=192, 192, 192
jTextPane1.text=Παρακαλώ εισάγεται τον κωδικό σύνδεσης που εμφανίζεται στο
κινητό τηλέφωνο.
connectButton.text=Σύνδεση
cancelButton.text=Ακύρωση
connect.Action.text=
connect.Action.shortDescription=
cancel.Action.shortDescription=
cancel.Action.text=
#NOI18N
jTextPane1.background=238, 238, 238
```

```
#
# DesktopMedicalRecordView.properties
#
fileMenu.text = Αρχείο
helpMenu.text = Βοήθεια
showAboutBox.Action.text = About...
showAboutBox.Action.shortDescription = Show the application's information
dialog

# status bar resources

StatusBar.messageTimeout = 5000
StatusBar.busyAnimationRate = 30
StatusBar.idleIcon = busyicons/idle-icon.png
StatusBar.busyIcons[0]  = busyicons/busy-icon0.png
StatusBar.busyIcons[1]  = busyicons/busy-icon1.png
StatusBar.busyIcons[2]  = busyicons/busy-icon2.png
StatusBar.busyIcons[3]  = busyicons/busy-icon3.png
StatusBar.busyIcons[4]  = busyicons/busy-icon4.png
StatusBar.busyIcons[5]  = busyicons/busy-icon5.png
StatusBar.busyIcons[6]  = busyicons/busy-icon6.png
StatusBar.busyIcons[7]  = busyicons/busy-icon7.png
StatusBar.busyIcons[8]  = busyicons/busy-icon8.png
StatusBar.busyIcons[9]  = busyicons/busy-icon9.png
StatusBar.busyIcons[10] = busyicons/busy-icon10.png
StatusBar.busyIcons[11] = busyicons/busy-icon11.png
StatusBar.busyIcons[12] = busyicons/busy-icon12.png
StatusBar.busyIcons[13] = busyicons/busy-icon13.png
StatusBar.busyIcons[14] = busyicons/busy-icon14.png
clearFields.Action.shortDescription=
clearFields.Action.text=
saveFieldValues.Action.shortDescription=
saveFieldValues.Action.text=
jPanel8.TabConstraints.tabTitle=Στοιχεία Πλησιέστερου Ασθενή
jPanel6.TabConstraints.tabTitle=Οδηγίες κατά την έξοδο
jPanel5.TabConstraints.tabTitle=Θεραπευτική Αγωγή
jPanel4.TabConstraints.tabTitle=Διάγνωση Εξόδου
jPanel3.TabConstraints.tabTitle=Πορεία Νόσου
jPanel2.TabConstraints.tabTitle=Ιστορικό - Αντικειμενική Εξέταση
jPanel1.TabConstraints.tabTitle=Στοιχεία Ασθενή
jPanel7.TabConstraints.tabTitle=Εξετάσεις
jLabel9.text=ΗΚΓ
jLabel10.text=Ακτινολογικός Έλεγχος
jLabel11.text=Λοιπές Εξετάσεις
exitMenuItem.text=Έξοδος
amkaLabel.text=ΑΜΚΑ
surnameLabel.text=Επώνυμο
nameLabel.text=Όνομα
bloodTypeRhLabel.text=Ομάδα Αίματος Rh
addressLabel.text=Διεύθυνση
postalCodeLabel.text=Τ.Κ.
cityLabel.text=Πόλη
phoneNumberLabel.text=Τηλέφωνο
amka.text=
surname.text=
name.text=
address.text=
bloodTypeRh.text=
postalCode.text=
tmp.Action.text=
tmp.Action.shortDescription=
```

```
saveRecord.Action.accelerator=ctrl pressed S
saveRecord.Action.shortDescription=
saveRecord.Action.text=Save Record
nrSurnameLabel.text=Επώνυμο
nrNameLabel.text=Όνομα
nrPhoneNumberLabel.text=Τηλέφωνο
nrRelationshipLabel.text=Συγγένεια
undoRecordChanges.Action.shortDescription=
undoRecordChanges.Action.text=
jPanel10.TabConstraints.tabTitle=Εργαστηριακές Εξετάσεις
jTable1.columnModel.title0=Πεδίο
jTable1.columnModel.title1=Τιμή
jTable2.columnModel.title3=Title 4
jTable2.columnModel.title2=Title 3
jTextArea3.text=
bluetoothUpdate.Action.shortDescription=Σύνδεση με κινητό για λήψη φακέλου
bluetoothUpdate.Action.text=Σύνδεση
bluetoothUpdate.Action.accelerator=ctrl pressed B
exitMenuItem.toolTipText=Έξοδος από την εφαρμογή
getRecordViaBluetooth.Action.text=Λήψη Φακέλου
getRecordViaBluetooth.Action.accelerator=ctrl pressed B
getRecordViaBluetooth.Action.shortDescription=
jButton3.text=jButton3
jMenuItem1.AccessibleContext.accessibleDescription=Σύνδεση με Bluetooth
jPanel11.TabConstraints.tabTitle=Άμεση Ανάγκη
medicalRecordMenu.text=Ιατρικός Φάκελος
getRecordMenuItem.text=Λήψη
putRecordMenuItem.text=Αποστολή
cancelButton.text=Ακύρωση
submitButton.text=Καταχώρηση
submitButton.toolTipText=
puRecordViaBluetooth.Action.text=
puRecordViaBluetooth.Action.accelerator=ctrl pressed P
puRecordViaBluetooth.Action.shortDescription=
jMenuItem1.text=jMenuItem1
showDialogBox.Action.text=
showDialogBox.Action.shortDescription=
aboutMenuItem.text=Περί
```

MobileMedicalRecord

```
/*
 * CodeGenerator.java
 */
package org.ntua.mobile.medicalrecord.connections;

/**
 *
 * @author Panayiotis Tembriotis
 */
public class CodeGenerator extends java.util.Random {

    static String code;

    public CodeGenerator() {
        getCode();
    }

    private void getCode() {
        CodeGenerator.code = "";
        for (int i = 0; i < 5; i++) {
            code += String.valueOf(getInt(9));
        }
    }

    private int getInt(int n) {
        if (n <= 0) {
            throw new IllegalArgumentException("n must be positive");
        }

        if ((n & -n) == n) // i.e., n is a power of 2
        {
            return (int) ((n * (long) next(31)) >> 31);
        }

        int bits, val;
        do {
            bits = next(31);
            val = bits % n;
        } while (bits - val + (n - 1) < 0);
        return val;
    }
}
```

```
 /*
  * OBEXServer.java
  */
 package org.ntua.mobile.medicalrecord.connections;

 import java.io.ByteArrayOutputStream;
 import java.io.DataInputStream;
 import java.io.DataOutputStream;
 import java.io.IOException;
 import javax.bluetooth.BluetoothStateException;
 import javax.bluetooth.DiscoveryAgent;
 import javax.bluetooth.LocalDevice;
 import javax.bluetooth.UUID;
 import javax.microedition.io.Connector;
 import javax.obex.Authenticator;
 import javax.obex.HeaderSet;
 import javax.obex.Operation;
 import javax.obex.PasswordAuthentication;
 import javax.obex.ResponseCodes;
 import javax.obex.ServerRequestHandler;
 import javax.obex.SessionNotifier;
 import org.ntua.mobile.medicalrecord.midlet.MedicalRecordMIDlet;
 import org.ntua.mobile.medicalrecord.storage.EmergencyRecord;
 import org.ntua.mobile.medicalrecord.storage.MedicalRecord;

 public class OBEXServer {

     private static final String PROTOCOL = "btgoep";
     private static final String HOST = "localhost";
     private static final UUID SERVER_UUID = new UUID(0x1105);
     private static final String SERVICE_NAME = "MedicalRecord";
     private static final String SECURITY =
"authenticate=true;encrypt=true;master=false";
 //    private boolean done;
     private boolean authenticated;
     private byte[] username;
     private byte[] password;
     private MedicalRecordMIDlet parent;

     public OBEXServer(MedicalRecordMIDlet parent) {
         this.parent = parent;
         password = CodeGenerator.code.getBytes();
     }

     public void startServer() throws BluetoothStateException, IOException {
         System.out.println("startServer");
 //        done = false;
         authenticated = false;
         username = parent.getMedicalRecord().getAmka().getBytes();
 //        password = CodeGenerator.code.getBytes();
         LocalDevice.getLocalDevice().setDiscoverable(DiscoveryAgent.GIAC);
         parent.getNetworkUpdateWaitScreen().setText("Κωδικός Σύνδεσης: " +
parent.getMedicalRecord().getAmka());
 //        parent.getNetworkUpdateWaitScreen().setText("Κωδικός Σύνδεσης: " +
new String(password));
         System.out.println("Local Device Bluetooth Address: " +
LocalDevice.getLocalDevice().getBluetoothAddress());
         SessionNotifier serverConnection = (SessionNotifier)
Connector.open(connectionURL());
         MyServerRequestHandler handler = new MyServerRequestHandler();
         MyAuthenticator authenticator = new MyAuthenticator();
```

```
            serverConnection.acceptAndOpen(handler, authenticator);
            System.out.println("Received OBEX connection ");
  //          while (!done) {
  //          }
        }

        private String connectionURL() {
            String url = OBEXServer.PROTOCOL + "://" + OBEXServer.HOST + ":" +
OBEXServer.SERVER_UUID
                    + ";" + "name=" + OBEXServer.SERVICE_NAME + ";" +
OBEXServer.SECURITY;
            System.out.println("URL = " + url);
            return url;
        }

        public class MyServerRequestHandler extends ServerRequestHandler {

            public void onAuthenticationFailure(byte[] userName) {
                parent.getNetworkUpdateWaitScreen().setText("Αποτυχία
επαλήθευσης.");
                super.onAuthenticationFailure(userName);
            }

            public int onConnect(HeaderSet request, HeaderSet reply) {
                System.out.println("onConnect");
                System.out.println("auth = " + authenticated);
                parent.getNetworkUpdateWaitScreen().setText("Σύνδεση");
  //              if (!authenticated) {
  //                  reply.createAuthenticationChallenge("MedicalRecord", true,
true);
  //                  authenticated = true;
  //                  return ResponseCodes.OBEX_HTTP_UNAUTHORIZED;
  //              }
                authenticated = false;
                return super.onConnect(request, reply);
            }

            public void onDisconnect(HeaderSet request, HeaderSet reply) {
                parent.getNetworkUpdateWaitScreen().setText("Ολοκλήρωση
σύνδεσης.");
                System.out.println("onDisconnect");
  //            done = true;
                super.onDisconnect(request, reply);
            }

            public int onGet(Operation op) {
                System.out.println("onGet");
                parent.getNetworkUpdateWaitScreen().setText("Αποστολή φακέλου.");
                byte[] obj = null;
                try {
                    HeaderSet hs = op.getReceivedHeaders();
                    String name = (String) hs.getHeader(HeaderSet.NAME);
                    if (name != null) {
                        System.out.println("object name:" + name);
                        if (name.equals(MedicalRecord.RECORD_DATA_TYPE)) {
                            obj = parent.getMedicalRecord().toByteArray();
                        } else if (name.equals(EmergencyRecord.RECORD_DATA_TYPE))
{
                            obj = parent.getEmergencyRecord().toByteArray();
                        }
                    }
```

```
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            try {
                DataOutputStream out = op.openDataOutputStream();
                out.write(obj);
                out.close();
                op.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            return super.onGet(op);
        }

        public int onPut(Operation op) {
            System.out.println("onPut");
            parent.getNetworkUpdateWaitScreen().setText("Λήψη φακέλου.");
            String name = "";
            try {
                HeaderSet hs = op.getReceivedHeaders();
                name = (String) hs.getHeader(HeaderSet.NAME);
                if (name != null) {
                    System.out.println("object name:" + name);
                } else {
                    throw new IOException();
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            try {
                DataInputStream in = op.openDataInputStream();
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                int readByte;
                while ((readByte = in.read()) != -1) {
                    baos.write((byte) readByte);
                }
                if (name.equals(MedicalRecord.RECORD_DATA_TYPE)) {
                    parent.getMedicalRecord().fromByteArray(baos.toByteArray()
);
                } else if (name.equals(EmergencyRecord.RECORD_DATA_TYPE)) {
                    parent.getEmergencyRecord().fromByteArray(baos.toByteArray
());
                }
                baos.close();
                in.close();
                op.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            return super.onGet(op);
        }
    }

    public class MyAuthenticator implements Authenticator {

        public PasswordAuthentication onAuthenticationChallenge(String
description, boolean isUserIdRequired, boolean isFullAccess) {
            System.out.println("onAuthenticationChallenge");
            parent.getNetworkUpdateWaitScreen().setText("Επαλήθευση.");
            System.out.println("description: " + description);
            authenticated = true;
```

```
            return new PasswordAuthentication(username, password);
          }

          public byte[] onAuthenticationResponse(byte[] username) {
            System.out.println("onAuthenticationResponse");
            parent.getNetworkUpdateWaitScreen().setText("Επαλήθευση.");
  //        System.out.println("userName = " + new String(username));
  //        if (compareArrays(this.username, username) == 0) {
              return password;
  //        }
  //        return null;
          }
  //    private int compareArrays(byte[] array1, byte[] array2) {
  //        int result = -1;
  //        if (array1 != null && array2 != null && (array1.length ==
array2.length)) {
  //            for (int i = 0; i < array1.length; i++) {
  //                if (array1[i] != array2[i]) {
  //                    break;
  //                } else {
  //                    result = 0;
  //                }
  //            }
  //        }
  //        return result;
  //    }
      }
  }
```

```
/*
 * MyCipher.java
 */
package org.ntua.mobile.medicalrecord.encryption;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.ShortBufferException;
import javax.crypto.spec.SecretKeySpec;

/**
 *
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class MyCipher {

    private static final String ALGORITHM = "AES/ECB/PKCS5Padding";
    private static final String SECRET_KEY_ALGORITHM = "AES";
    private static final int KEY_LENGTH = 16;
    private byte[] padding = {};
    private byte[] secretKey;
    private Cipher cipher;
    private Key key;

    public MyCipher(String password) throws IOException,
NoSuchAlgorithmException, NoSuchPaddingException {
        secretKey = createSecretKey(password.getBytes());
        key = new SecretKeySpec(secretKey, 0, secretKey.length,
MyCipher.SECRET_KEY_ALGORITHM);
        cipher = Cipher.getInstance(MyCipher.ALGORITHM);
    }

    public byte[] encrypt(byte[] plainText) throws InvalidKeyException,
IllegalStateException, ShortBufferException, IllegalBlockSizeException,
BadPaddingException {
        cipher.init(Cipher.ENCRYPT_MODE, key);
        return runCipher(plainText);
    }

    public byte[] decrypt(byte[] encryptedText) throws InvalidKeyException,
IllegalStateException, ShortBufferException, IllegalBlockSizeException,
BadPaddingException {
        cipher.init(Cipher.DECRYPT_MODE, key);
        return runCipher(encryptedText);
    }

    private byte[] runCipher(byte[] inputBs) throws IllegalStateException,
ShortBufferException, IllegalBlockSizeException, BadPaddingException {
        int blocksize = 16;
        int outputLength = 0;
        int remainder = inputBs.length % blocksize;
        if (remainder == 0) {
            outputLength = inputBs.length;
        } else {
```

```
                outputLength = inputBs.length - remainder + blocksize;
            }
            byte[] outputBs = new byte[outputLength];
            outputLength = cipher.doFinal(inputBs, 0, inputBs.length, outputBs,
0);
            return fixOutputBytes(outputBs, outputLength);
        }

        private byte[] createSecretKey(byte[] password) throws IOException {
            ByteArrayOutputStream baos = new
ByteArrayOutputStream(MyCipher.KEY_LENGTH);
            int remainingBytes = MyCipher.KEY_LENGTH - password.length;
            baos.write(password, 0, password.length);
            baos.write(padding, 0, remainingBytes);
            byte[] createdKey = baos.toByteArray();
            baos.flush();
            baos.close();
            return createdKey;
        }

        private byte[] fixOutputBytes(byte[] inputBs, int length) {
            byte[] outputBs = new byte[length];
            System.arraycopy(inputBs, 0, outputBs, 0, length);
            return outputBs;

        }
    }
```

```
/*
 * MedicalRecordMIDlet.java
 */
package org.ntua.mobile.medicalrecord.midlet;

import java.util.Hashtable;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.ImageItem;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.List;
import javax.microedition.lcdui.Spacer;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import org.ntua.mobile.medicalrecord.storage.MedicalRecord;
import org.ntua.mobile.medicalrecord.storage.RMS;
import org.netbeans.microedition.lcdui.SimpleTableModel;
import org.netbeans.microedition.lcdui.TableItem;
import org.netbeans.microedition.lcdui.WaitScreen;
import org.netbeans.microedition.util.Executable;
import org.netbeans.microedition.util.SimpleCancellableTask;
import org.ntua.mobile.medicalrecord.connections.CodeGenerator;
import org.ntua.mobile.medicalrecord.connections.OBEXServer;
import org.ntua.mobile.medicalrecord.storage.EmergencyRecord;

/**
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class MedicalRecordMIDlet extends MIDlet implements CommandListener {

    public static final String SMS_SENDER_ADDRESS = "SMS-Address-URL";
    public static final String SMS_SENDER_PORT = "SMS-Sender-Port";
    public static final String SMS_SERVER_PORT = "SMS-Server-Port";
    public static final String MMS_SENDER_ADDRESS = "MMS-Address-URL";
    public static final String MMS_SENDER_PORT = "MMS-Sender-Port";
    public static final String MMS_SERVER_PORT = "MMS-Server-Port";
    public static final String MMS_APPLICATION_ID = "MMS-Application-ID";
    public static final String DATAGRAM_SENDER_ADDRESS = "Datagram-Address-
URL";
    public static final String DATAGRAM_SENDER_PORT = "Datagram-Sender-Port";
    public static final String DATAGRAM_SERVER_PORT = "Datagram-Server-Port";
    private boolean midletPaused = false;
    private int recordId = -1;
    private String password;
    private SimpleTableModel labExams1;
    private SimpleTableModel labExams2;
    private MedicalRecord medicalRecord;
    private EmergencyRecord emergencyRecord;
    private RMS rms;
    private CodeGenerator passwordGenerator;
    private OBEXServer obexServer;
    //<editor-fold defaultstate="collapsed" desc=" Generated Fields ">
```

```
    private java.util.Hashtable __previousDisplayables = new
java.util.Hashtable();
    private List mainMenuList;
    private Form medicalRecordForm01;
    private TextField bloodTypeRh;
    private TextField name;
    private TextField surname;
    private TextField address;
    private TextField postalCode;
    private TextField phoneNumber;
    private TextField city;
    private StringItem amka;
    private WaitScreen networkUpdateWaitScreen;
    private List networkUpdateMenuList;
    private Alert pinInvalidAlert;
    private Alert updateMedicalRecordSuccessAlert;
    private Alert updateMedicalRecordFailAlert;
    private Alert recordInvalidAlert;
    private Form medicalRecordForm03;
    private StringItem historyExamEvaluation;
    private Form medicalRecordForm08;
    private StringItem otherExams;
    private StringItem radiography;
    private StringItem ecg;
    private Form medicalRecordForm07;
    private StringItem exitDirectivesComments;
    private Form medicalRecordForm05;
    private StringItem exitDiagnosis;
    private Form medicalRecordForm06;
    private StringItem treatmentIntervention;
    private Form medicalRecordForm02;
    private TextField nearestRelativeName;
    private TextField nearestRelativeSurname;
    private TextField nearestRelativeRelationship;
    private TextField nearestRelativePhoneNumber;
    private Form medicalRecordForm04;
    private StringItem diceaseProgress;
    private Form labExamsForm01;
    private TableItem tableItem;
    private Form loginScreen;
    private TextField pinField;
    private Spacer spacer;
    private StringItem publicItem;
    private Alert deleteRecordAlert;
    private Form labExamsForm02;
    private TableItem tableItem1;
    private Alert amkaInvalidAlert;
    private Form InitialScreen;
    private Spacer spacer1;
    private TextField initialAmkaTextField;
    private TextField initialPinTextField;
    private Alert deleteInvalidRecordAlert;
    private Command exitCommand;
    private Command backCommand;
    private Command okCommand;
    private Command nextCommand;
    private Command loginCommand;
    private Command continueCommand;
    private Command cancelCommand;
    private Command deleteCommand;
    private Image sandglassImage;
```

```
      private Image alertImage;
      private SimpleCancellableTask simpleCancellableTask;
      private Image medicalrecordIcon;
      //</editor-fold>

      // <editor-fold defaultstate="collapsed" desc="Constructor">
      /**
       * The MedicalRecordMIDlet constructor.
       */
      public MedicalRecordMIDlet() {
  //        System.out.println("SmsSenderAddress: " +
getNetworkConnections().getSmsSenderAddress() + "\n" + "SmsServerAddress: " +
getNetworkConnections().getSmsServerAddress() + "\n" + "SMS_SENDER_ADDRESS: " +
getAppProperty(SMS_SENDER_ADDRESS) + "\n" + "SMS_SENDER_PORT: " +
getAppProperty(SMS_SENDER_PORT) + "\n" + "SMS_SERVER_PORT: " +
getAppProperty(SMS_SERVER_PORT) + "\n" + "MmsSenderAddress: " +
getNetworkConnections().getMmsSenderAddress() + "\n" + "MmsServerAddress: " +
getNetworkConnections().getMmsServerAddress() + "\n" + "MMS_SENDER_ADDRESS: " +
getAppProperty(MMS_SENDER_ADDRESS) + "\n" + "MMS_SENDER_PORT: " +
getAppProperty(MMS_SENDER_PORT) + "\n" + "MMS_SERVER_PORT: " +
getAppProperty(MMS_SERVER_PORT) + "\n" + "MMS_APPLICATION_ID: " +
getAppProperty(MMS_APPLICATION_ID) + "\n" + "DATAGRAM_SENDER_ADDRESS: " +
getAppProperty(DATAGRAM_SENDER_ADDRESS) + "\n" + "DATAGRAM_SENDER_PORT: " +
getAppProperty(DATAGRAM_SENDER_PORT) + "\n" + "DATAGRAM_SERVER_PORT: " +
getAppProperty(DATAGRAM_SERVER_PORT));
      }// </editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Methods ">
      /**
       * Switches a display to previous displayable of the current displayable.
       * The <code>display</code> instance is obtain from the
<code>getDisplay</code> method.
       */
      private void switchToPreviousDisplayable() {
          Displayable __currentDisplayable = getDisplay().getCurrent();
          if (__currentDisplayable != null) {
              Displayable __nextDisplayable = (Displayable)
__previousDisplayables.get(__currentDisplayable);
              if (__nextDisplayable != null) {
                  switchDisplayable(null, __nextDisplayable);
              }
          }
      }
      //</editor-fold>
      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
initialize ">
      /**
       * Initilizes the application.
       * It is called only once when the MIDlet is started. The method is called
before the <code>startMIDlet</code> method.
       */
      private void initialize() {
          // write pre-initialize user code here
          generatePassword();
  //        int recordId;
  //        getRMS().deleteRecord();
  //        MedicalRecord mr = new MedicalRecord(MedicalRecord.dummyRecord());
  //        try {
  //            recordId = getRMS().addMedicalRecord(mr);
  //            System.out.println("Record ID: " + recordId);
  ////            getRMS().getMedicalRecord(recordId).print();
```

```
  //        } catch (Exception ex) {
  //            ex.printStackTrace();
  //        }

          // write post-initialize user code here
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
startMIDlet ">
      /**
       * Performs an action assigned to the Mobile Device - MIDlet Started
point.
       */
      public void startMIDlet() {
          // write pre-action user code here
          isFirstTime();
          // write post-action user code here
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
resumeMIDlet ">
      /**
       * Performs an action assigned to the Mobile Device - MIDlet Resumed
point.
       */
      public void resumeMIDlet() {
          // write pre-action user code here
          System.out.println("Resume MIDlet.");

          // write post-action user code here
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
switchDisplayable ">
      /**
       * Switches a current displayable in a display. The <code>display</code>
instance is taken from <code>getDisplay</code> method. This method is used by
all actions in the design for switching displayable.
       * @param alert the Alert which is temporarily set to the display; if
<code>null</code>, then <code>nextDisplayable</code> is set immediately
       * @param nextDisplayable the Displayable to be set
       */
      public void switchDisplayable(Alert alert, Displayable nextDisplayable) {
          // write pre-switch user code here
          Display display = getDisplay();
          Displayable __currentDisplayable = display.getCurrent();
          if (__currentDisplayable != null  &&  nextDisplayable != null) {
              __previousDisplayables.put(nextDisplayable, __currentDisplayable);
          }
          if (alert == null) {
              display.setCurrent(nextDisplayable);
          } else {
              display.setCurrent(alert, nextDisplayable);
          }
  // write post-switch user code here
      }
      //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
commandAction for Displayables ">
    /**
     * Called by a system to indicated that a command has been invoked on a
particular displayable.
     * @param command the Command that was invoked
     * @param displayable the Displayable where the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
        // write pre-action user code here
        if (displayable == InitialScreen) {
            if (command == exitCommand) {
                // write pre-action user code here
                exitMIDlet();
                // write post-action user code here
            } else if (command == loginCommand) {
                // write pre-action user code here
                isAmkaValid();
                // write post-action user code here
            }
        } else if (displayable == deleteInvalidRecordAlert) {
            if (command == cancelCommand) {
                // write pre-action user code here
                switchToPreviousDisplayable();
                // write post-action user code here
            } else if (command == continueCommand) {
                // write pre-action user code here
                getRMS().deleteRecordStore();
                exitMIDlet();
                // write post-action user code here
            }
        } else if (displayable == deleteRecordAlert) {
            if (command == cancelCommand) {
                // write pre-action user code here
                switchToPreviousDisplayable();
                // write post-action user code here
            } else if (command == continueCommand) {
                // write pre-action user code here
                getRMS().deleteRecordStore();
                exitMIDlet();
                // write post-action user code here
            }
        } else if (displayable == labExamsForm01) {
            if (command == backCommand) {
                // write pre-action user code here
                switchDisplayable(null, getMedicalRecordForm08());
                // write post-action user code here
            } else if (command == cancelCommand) {
                // write pre-action user code here
                switchDisplayable(null, getMainMenuList());
                // write post-action user code here
            } else if (command == nextCommand) {
                // write pre-action user code here
                switchDisplayable(null, getLabExamsForm02());
                // write post-action user code here
            } else if (command == okCommand) {
                // write pre-action user code here
                isSaveSuccess();
                // write post-action user code here
            }
        } else if (displayable == labExamsForm02) {
```

```
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getLabExamsForm01());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == okCommand) {
        // write pre-action user code here
        isSaveSuccess();
        // write post-action user code here
    }
} else if (displayable == loginScreen) {
    if (command == exitCommand) {
        // write pre-action user code here
        exitMIDlet();
        // write post-action user code here
    } else if (command == loginCommand) {
        // write pre-action user code here
        isPinValid();
        // write post-action user code here
    }
} else if (displayable == mainMenuList) {
    if (command == List.SELECT_COMMAND) {
        // write pre-action user code here
        mainMenuListAction();
        // write post-action user code here
    } else if (command == exitCommand) {
        // write pre-action user code here
        exitMIDlet();
        // write post-action user code here
    }
} else if (displayable == medicalRecordForm01) {
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == nextCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm02());
        // write post-action user code here
    } else if (command == okCommand) {
        // write pre-action user code here
        isSaveSuccess();
        // write post-action user code here
    }
} else if (displayable == medicalRecordForm02) {
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm01());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == nextCommand) {
```

```
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm04());
            // write post-action user code here
        } else if (command == okCommand) {
            // write pre-action user code here
            isSaveSuccess();
            // write post-action user code here
        }
    } else if (displayable == medicalRecordForm03) {
        if (command == backCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm02());
            // write post-action user code here
        } else if (command == cancelCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMainMenuList());
            // write post-action user code here
        } else if (command == nextCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm04());
            // write post-action user code here
        } else if (command == okCommand) {
            // write pre-action user code here
            isSaveSuccess();
            // write post-action user code here
        }
    } else if (displayable == medicalRecordForm04) {
        if (command == backCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm03());
            // write post-action user code here
        } else if (command == cancelCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMainMenuList());
            // write post-action user code here
        } else if (command == nextCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm05());
            // write post-action user code here
        } else if (command == okCommand) {
            // write pre-action user code here
            isSaveSuccess();
            // write post-action user code here
        }
    } else if (displayable == medicalRecordForm05) {
        if (command == backCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm04());
            // write post-action user code here
        } else if (command == cancelCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMainMenuList());
            // write post-action user code here
        } else if (command == nextCommand) {
            // write pre-action user code here
            switchDisplayable(null, getMedicalRecordForm06());
            // write post-action user code here
        } else if (command == okCommand) {
            // write pre-action user code here
            isSaveSuccess();
            // write post-action user code here
```

```
    }
} else if (displayable == medicalRecordForm06) {
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm05());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == nextCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm07());
        // write post-action user code here
    } else if (command == okCommand) {
        // write pre-action user code here
        isSaveSuccess();
        // write post-action user code here
    }
} else if (displayable == medicalRecordForm07) {
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm06());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == nextCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm08());
        // write post-action user code here
    } else if (command == okCommand) {
        // write pre-action user code here
        isSaveSuccess();
        // write post-action user code here
    }
} else if (displayable == medicalRecordForm08) {
    if (command == backCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMedicalRecordForm07());
        // write post-action user code here
    } else if (command == cancelCommand) {
        // write pre-action user code here
        switchDisplayable(null, getMainMenuList());
        // write post-action user code here
    } else if (command == nextCommand) {
        // write pre-action user code here
        switchDisplayable(null, getLabExamsForm01());
        // write post-action user code here
    } else if (command == okCommand) {
        // write pre-action user code here
        isSaveSuccess();
        // write post-action user code here
    }
} else if (displayable == networkUpdateMenuList) {
    if (command == List.SELECT_COMMAND) {
        // write pre-action user code here
        networkUpdateMenuListAction();
        // write post-action user code here
    } else if (command == backCommand) {
```

```
                // write pre-action user code here
                switchDisplayable(null, getMainMenuList());
                // write post-action user code here
            }
        } else if (displayable == networkUpdateWaitScreen) {
            if (command == WaitScreen.FAILURE_COMMAND) {
                // write pre-action user code here
                System.out.println("fail: " +
getNetworkUpdateWaitScreen().getTask().getFailureMessage());
                switchDisplayable(getUpdateMedicalRecordFailAlert(),
getNetworkUpdateMenuList());
                // write post-action user code here
            } else if (command == WaitScreen.SUCCESS_COMMAND) {
                // write pre-action user code here
                System.out.println("success " +
getNetworkUpdateWaitScreen().getTask().getFailureMessage());
                switchDisplayable(getUpdateMedicalRecordSuccessAlert(),
getMainMenuList());
                // write post-action user code here
            }
        } else if (displayable == recordInvalidAlert) {
            if (command == backCommand) {
                // write pre-action user code here
                switchToPreviousDisplayable();
                // write post-action user code here
            } else if (command == deleteCommand) {
                // write pre-action user code here
                getRMS().deleteRecordStore();
                exitMIDlet();
                // write post-action user code here
            }
        }
        // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
mainMenuList ">
    /**
     * Returns an initiliazed instance of mainMenuList component.
     * @return the initialized component instance
     */
    public List getMainMenuList() {
        if (mainMenuList == null) {
            // write pre-init user code here
            mainMenuList = new
List("\u0399\u03B1\u03C4\u03C1\u03B9\u03BA\u03CC\u03C2
\u03A6\u03AC\u03BA\u03B5\u03BB\u03BF\u03C2", Choice.IMPLICIT);
            mainMenuList.append("\u0391\u03BD\u03AC\u03B3\u03BD\u03C9\u03C3\u0
3B7", null);
            mainMenuList.append("\u0395\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u0
3C3\u03B7", null);
            mainMenuList.append("\u0394\u03B9\u03B1\u03B3\u03C1\u03B1\u03C6\u0
3AE", null);
            mainMenuList.addCommand(getExitCommand());
            mainMenuList.setCommandListener(this);
            mainMenuList.setSelectedFlags(new boolean[] { false, false,
false });
            // write post-init user code here
        }
        return mainMenuList;
```

```
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
mainMenuListAction ">
    /**
     * Performs an action assigned to the selected list element in the
mainMenuList component.
     */
    public void mainMenuListAction() {
        // enter pre-action user code here
        String __selectedString =
getMainMenuList().getString(getMainMenuList().getSelectedIndex());
        if (__selectedString != null) {
            if
(__selectedString.equals("\u0391\u03BD\u03AC\u03B3\u03BD\u03C9\u03C3\u03B7")) {
                // write pre-action user code here
                System.out.println("recordId=" + recordId);
                getMedicalRecord().fromByteArray(getRMS().getRecord(getMedical
Record().getRecordId()));
                switchDisplayable(null, getMedicalRecordForm01());
                // write post-action user code here
            } else if
(__selectedString.equals("\u0395\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u03C3\u03B7
")) {
                // write pre-action user code here
                networkUpdateWaitScreen = null;
                getMedicalRecord().fromByteArray(getRMS().getRecord(getMedical
Record().getRecordId()));
                switchDisplayable(null, getNetworkUpdateMenuList());
                // write post-action user code here
            } else if
(__selectedString.equals("\u0394\u03B9\u03B1\u03B3\u03C1\u03B1\u03C6\u03AE")) {
                // write pre-action user code here
                switchDisplayable(null, getDeleteRecordAlert());
                // write post-action user code here
            }
        }
        // enter post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm01 ">
    /**
     * Returns an initiliazed instance of medicalRecordForm01 component.
     * @return the initialized component instance
     */
    public Form getMedicalRecordForm01() {
        if (medicalRecordForm01 == null) {
            // write pre-init user code here
            medicalRecordForm01 = new
Form("\u03A3\u03C4\u03BF\u03B9\u03C7\u03B5\u03AF\u03B1
\u0391\u03C3\u03B8\u03B5\u03BD\u03AE", new Item[] { getAmka(), getSurname(),
getName(), getBloodTypeRh(), getAddress(), getPostalCode(), getCity(),
getPhoneNumber() });
            medicalRecordForm01.addCommand(getBackCommand());
            medicalRecordForm01.addCommand(getNextCommand());
            medicalRecordForm01.addCommand(getOkCommand());
            medicalRecordForm01.addCommand(getCancelCommand());
            medicalRecordForm01.setCommandListener(this);
```

```
            // write post-init user code here
  //            getAmka().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getSurname().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getName().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getBloodTypeRh().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getAddress().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getPostalCode().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getCity().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
  //            getPhoneNumber().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
          }
          return medicalRecordForm01;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
backCommand ">
      /**
       * Returns an initiliazed instance of backCommand component.
       * @return the initialized component instance
       */
      public Command getBackCommand() {
          if (backCommand == null) {
              // write pre-init user code here
              backCommand = new Command("\u03A0\u03AF\u03C3\u03C9",
Command.BACK, 0);
              // write post-init user code here
          }
          return backCommand;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
exitCommand ">
      /**
       * Returns an initiliazed instance of exitCommand component.
       * @return the initialized component instance
       */
      public Command getExitCommand() {
          if (exitCommand == null) {
              // write pre-init user code here
              exitCommand = new Command("\u0388\u03BE\u03BF\u03B4\u03BF\u03C2",
Command.EXIT, 0);
              // write post-init user code here
          }
          return exitCommand;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter: okCommand
">
      /**
       * Returns an initiliazed instance of okCommand component.
       * @return the initialized component instance
```

```
     */
    public Command getOkCommand() {
        if (okCommand == null) {
            // write pre-init user code here
            okCommand = new
Command("\u0391\u03C0\u03BF\u03B8\u03AE\u03BA\u03B5\u03C5\u03C3\u03B7",
Command.OK, 0);
            // write post-init user code here
        }
        return okCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: surname
">
    /**
     * Returns an initiliazed instance of surname component.
     * @return the initialized component instance
     */
    public TextField getSurname() {
        if (surname == null) {
            // write pre-init user code here
            surname = new
TextField("\u0395\u03C0\u03CE\u03BD\u03C5\u03BC\u03BF", null, 16,
TextField.ANY);
            surname.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
            surname.setInitialInputMode("UCB_GREEK");
            // write post-init user code here
            surname.setString(getMedicalRecord().getSurname());
        }
        return surname;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: name ">
    /**
     * Returns an initiliazed instance of name component.
     * @return the initialized component instance
     */
    public TextField getName() {
        if (name == null) {
            // write pre-init user code here
            name = new TextField("\u038C\u03BD\u03BF\u03BC\u03B1", null, 16,
TextField.ANY);
            name.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
            name.setInitialInputMode("UCB_GREEK");
            // write post-init user code here
            name.setString(getMedicalRecord().getName());
        }
        return name;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
bloodTypeRh ">
    /**
     * Returns an initiliazed instance of bloodTypeRh component.
```

```
       * @return the initialized component instance
       */
      public TextField getBloodTypeRh() {
          if (bloodTypeRh == null) {
              // write pre-init user code here
              bloodTypeRh = new TextField("\u039F\u03BC\u03AC\u03B4\u03B1
\u0391\u03AF\u03BC\u03B1\u03C4\u03BF\u03C2 Rh", null, 4, TextField.ANY);
              bloodTypeRh.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
              bloodTypeRh.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              bloodTypeRh.setString(getMedicalRecord().getBloodTypeRh());
          }
          return bloodTypeRh;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter: address
">
      /**
       * Returns an initiliazed instance of address component.
       * @return the initialized component instance
       */
      public TextField getAddress() {
          if (address == null) {
              // write pre-init user code here
              address = new
TextField("\u0394\u03B9\u03B5\u03CD\u03B8\u03C5\u03BD\u03C3\u03B7", null, 64,
TextField.ANY);
              address.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
              address.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              address.setString(getMedicalRecord().getAddress());
          }
          return address;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
postalCode ">
      /**
       * Returns an initiliazed instance of postalCode component.
       * @return the initialized component instance
       */
      public TextField getPostalCode() {
          if (postalCode == null) {
              // write pre-init user code here
              postalCode = new TextField("\u03A4.\u039A.", null, 5,
TextField.NUMERIC);
              postalCode.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
              postalCode.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              postalCode.setString(getMedicalRecord().getPostalCode());
          }
          return postalCode;
      }
```

```
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
phoneNumber ">
      /**
       * Returns an initiliazed instance of phoneNumber component.
       * @return the initialized component instance
       */
      public TextField getPhoneNumber() {
          if (phoneNumber == null) {
              // write pre-init user code here
              phoneNumber = new
TextField("\u03A4\u03B7\u03BB\u03AD\u03C6\u03C9\u03BD\u03BF", null, 16,
TextField.PHONENUMBER);
              phoneNumber.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
              // write post-init user code here
              phoneNumber.setString(getMedicalRecord().getPhoneNumber());
          }
          return phoneNumber;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter: city ">
      /**
       * Returns an initiliazed instance of city component.
       * @return the initialized component instance
       */
      public TextField getCity() {
          if (city == null) {
              // write pre-init user code here
              city = new TextField("\u03A0\u03CC\u03BB\u03B7", null, 16,
TextField.ANY);
              city.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
              city.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              city.setString(getMedicalRecord().getCity());
          }
          return city;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isLoginSuccess ">
      /**
       * Performs an action assigned to the isLoginSuccess if-point.
       */
      public void isLoginSuccess() {
          // enter pre-if user code here
          boolean isSuccess = false;
          int[] recordIds;
          try {
              if ((recordIds =
getRMS().getRecordIds(MedicalRecord.RECORD_DATA_TYPE)) != null
                      && recordIds.length == 1 &&
getRMS().getRecord(recordIds[0]) != null) {
                  recordId = recordIds[0];
                  System.out.println("this.recordId=" + recordId);
```

```
                isSuccess = true;
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        if (isSuccess) {
                // write pre-action user code here
            switchDisplayable(null, getMainMenuList());
                // write post-action user code here
        } else {
                // write pre-action user code here
  //            getRMS().deleteRecord(MedicalRecord.RECORD_DATA_TYPE);
            switchDisplayable(null, getRecordInvalidAlert());
                // write post-action user code here
        }
        // enter post-if user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isSaveSuccess ">
    /**
     * Performs an action assigned to the isSaveSuccess if-point.
     */
    public void isSaveSuccess() {
        // enter pre-if user code here
        boolean isSuccess = false;
        getMedicalRecord().setPatientAttrs(
                getSurname().getString()/*.replace('\n', ' ')*/,
                getName().getString()/*.replace('\n', ' ')*/,
                getBloodTypeRh().getString()/*.replace('\n', ' ')*/,
                getAddress().getString()/*.replace('\n', ' ')*/,
                getPostalCode().getString()/*.replace('\n', ' ')*/,
                getCity().getString()/*.replace('\n', ' ')*/,
                getPhoneNumber().getString()/*.replace('\n', ' ')*/,
                getNearestRelativeSurname().getString()/*.replace('\n', '
')*/,
                getNearestRelativeName().getString()/*.replace('\n', ' ')*/,
                getNearestRelativePhoneNumber().getString()/*.replace('\n', '
')*/,
                getNearestRelativeRelationship().getString()/*.replace('\n', '
')*/);
        try {
            getRMS().updateRecord(getMedicalRecord().getRecordId(),
getMedicalRecord().toByteArray());
            isSuccess = true;
        } catch (Exception ex) {
            isSuccess = false;
            ex.printStackTrace();
        }
        if (isSuccess) {
                // write pre-action user code here
            switchDisplayable(getUpdateMedicalRecordSuccessAlert(),
getMainMenuList());
                // write post-action user code here
  //            getMedicalRecord().print();
        } else {
                // write pre-action user code here
            switchDisplayable(getUpdateMedicalRecordFailAlert(),
getMainMenuList());
                // write post-action user code here
```

```
            }
            // enter post-if user code here
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isPinValid ">
        /**
         * Performs an action assigned to the isPinValid if-point.
         */
        public void isPinValid() {
            // enter pre-if user code here
            password = getPinField().getString();
            boolean isSuccess = checkPassword();
            if (isSuccess) {
                    // write pre-action user code here
                isLoginSuccess();
                    // write post-action user code here
            } else {
                    // write pre-action user code here
                switchDisplayable(getPinInvalidAlert(), getLoginScreen());
                    // write post-action user code here
            }
            // enter post-if user code here
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
networkUpdateWaitScreen ">
        /**
         * Returns an initiliazed instance of networkUpdateWaitScreen component.
         * @return the initialized component instance
         */
        public WaitScreen getNetworkUpdateWaitScreen() {
            if (networkUpdateWaitScreen == null) {
                // write pre-init user code here
                networkUpdateWaitScreen = new WaitScreen(getDisplay());
                networkUpdateWaitScreen.setTitle("\u0395\u03BD\u03B7\u03BC\u03AD\u
03C1\u03C9\u03C3\u03B7 \u0399\u03B1\u03C4\u03C1\u03B9\u03BA\u03BF\u03CD
\u03A6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C5");
                networkUpdateWaitScreen.setCommandListener(this);
                networkUpdateWaitScreen.setFullScreenMode(true);
                networkUpdateWaitScreen.setImage(getSandglassImage());
                networkUpdateWaitScreen.setText("\u03A0\u03B1\u03C1\u03B1\u03BA\u0
3B1\u03BB\u03CE
\u03A0\u03B5\u03C1\u03B9\u03BC\u03AD\u03BD\u03B5\u03C4\u03B5 ...");
                networkUpdateWaitScreen.setTask(getSimpleCancellableTask());
                // write post-init user code here
            }
            return networkUpdateWaitScreen;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
networkUpdateMenuList ">
        /**
         * Returns an initiliazed instance of networkUpdateMenuList component.
         * @return the initialized component instance
         */
        public List getNetworkUpdateMenuList() {
            if (networkUpdateMenuList == null) {
```

```
            // write pre-init user code here
            networkUpdateMenuList = new
List("\u0395\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u03C3\u03B7
\u0399\u03B1\u03C4\u03C1\u03B9\u03BA\u03BF\u03CD
\u03A6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C5", Choice.IMPLICIT);
            networkUpdateMenuList.append("\u03A3\u03CD\u03BD\u03B4\u03B5\u03C3
\u03B7 \u03BC\u03B5 \u0397/\u03A5", null);
            networkUpdateMenuList.addCommand(getBackCommand());
            networkUpdateMenuList.setCommandListener(this);
            networkUpdateMenuList.setSelectedFlags(new boolean[] { false });
            // write post-init user code here
        }
        return networkUpdateMenuList;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
networkUpdateMenuListAction ">
    /**
     * Performs an action assigned to the selected list element in the
networkUpdateMenuList component.
     */
    public void networkUpdateMenuListAction() {
        // enter pre-action user code here
        getNetworkUpdateWaitScreen().setText("\u03A0\u03B1\u03C1\u03B1\u03BA\u
03B1\u03BB\u03CE
\u03A0\u03B5\u03C1\u03B9\u03BC\u03AD\u03BD\u03B5\u03C4\u03B5 ...");
        getUpdateMedicalRecordFailAlert().setString("\u0391\u03C0\u03BF\u03C4\
u03C5\u03C7\u03AF\u03B1
\u03B5\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u03C3\u03B7\u03C2");
        String __selectedString =
getNetworkUpdateMenuList().getString(getNetworkUpdateMenuList().getSelectedIndex
());
        if (__selectedString != null) {
            if
(__selectedString.equals("\u03A3\u03CD\u03BD\u03B4\u03B5\u03C3\u03B7
\u03BC\u03B5 \u0397/\u03A5")) {
                // write pre-action user code here
                getSimpleCancellableTask().setExecutable(bluetoothUpdateEx
ecutable());
                switchDisplayable(null, getNetworkUpdateWaitScreen());
                // write post-action user code here
            }
        }
        // enter post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
updateMedicalRecordFailAlert ">
    /**
     * Returns an initiliazed instance of updateMedicalRecordFailAlert
component.
     * @return the initialized component instance
     */
    public Alert getUpdateMedicalRecordFailAlert() {
        if (updateMedicalRecordFailAlert == null) {
            // write pre-init user code here
            updateMedicalRecordFailAlert = new
Alert("\u03A3\u03C6\u03AC\u03BB\u03BC\u03B1",
"\u0391\u03C0\u03BF\u03C4\u03C5\u03C7\u03AF\u03B1
```

```
\u03B5\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u03C3\u03B7\u03C2", getAlertImage(),
AlertType.ERROR);
            updateMedicalRecordFailAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return updateMedicalRecordFailAlert;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
alertImage ">
    /**
     * Returns an initiliazed instance of alertImage component.
     * @return the initialized component instance
     */
    public Image getAlertImage() {
        if (alertImage == null) {
            // write pre-init user code here
            try {
                alertImage =
Image.createImage("/org/ntua/mobile/medicalrecord/images/alert.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return alertImage;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
sandglassImage ">
    /**
     * Returns an initiliazed instance of sandglassImage component.
     * @return the initialized component instance
     */
    public Image getSandglassImage() {
        if (sandglassImage == null) {
            // write pre-init user code here
            try {
                sandglassImage =
Image.createImage("/org/ntua/mobile/medicalrecord/images/sandglass.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return sandglassImage;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
updateMedicalRecordSuccessAlert ">
    /**
     * Returns an initiliazed instance of updateMedicalRecordSuccessAlert
component.
     * @return the initialized component instance
     */
    public Alert getUpdateMedicalRecordSuccessAlert() {
        if (updateMedicalRecordSuccessAlert == null) {
            // write pre-init user code here
```

```
            updateMedicalRecordSuccessAlert = new
Alert("\u0395\u03BD\u03B7\u03BC\u03AD\u03C1\u03C9\u03C3\u03B7", "\u039F
\u0399\u03B1\u03C4\u03C1\u03B9\u03BA\u03CC\u03C2
\u03A6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C2
\u03B5\u03BD\u03B7\u03BC\u03B5\u03C1\u03CE\u03B8\u03B7\u03BA\u03B5", null,
AlertType.INFO);
            updateMedicalRecordSuccessAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return updateMedicalRecordSuccessAlert;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
pinInvalidAlert ">
    /**
     * Returns an initiliazed instance of pinInvalidAlert component.
     * @return the initialized component instance
     */
    public Alert getPinInvalidAlert() {
        if (pinInvalidAlert == null) {
            // write pre-init user code here
            pinInvalidAlert = new
Alert("\u03A3\u03C6\u03AC\u03BB\u03BC\u03B1", "\u039C\u03B7
\u03B1\u03C0\u03BF\u03B4\u03B5\u03BA\u03C4\u03CC PIN", getAlertImage(),
AlertType.ERROR);
            pinInvalidAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return pinInvalidAlert;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
recordInvalidAlert ">
    /**
     * Returns an initiliazed instance of recordInvalidAlert component.
     * @return the initialized component instance
     */
    public Alert getRecordInvalidAlert() {
        if (recordInvalidAlert == null) {
            // write pre-init user code here
            recordInvalidAlert = new
Alert("\u03A3\u03C6\u03AC\u03BB\u03BC\u03B1",
"\u03A0\u03A1\u039F\u03A3\u039F\u03A7\u0397: \u039B\u03AC\u03B8\u03BF\u03C2
PIN \u03AE \u03BC\u03B7 \u03B1\u03C0\u03BF\u03B4\u03B5\u03BA\u03C4\u03AE
\u03BA\u03B1\u03C4\u03AC\u03C3\u03C4\u03B1\u03C3\u03B7
\u03C6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C5!", getAlertImage(), AlertType.ERROR);
            recordInvalidAlert.addCommand(getDeleteCommand());
            recordInvalidAlert.addCommand(getBackCommand());
            recordInvalidAlert.setCommandListener(this);
            recordInvalidAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return recordInvalidAlert;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
simpleCancellableTask ">
    /**
```

```
       * Returns an initiliazed instance of simpleCancellableTask component.
       * @return the initialized component instance
       */
      public SimpleCancellableTask getSimpleCancellableTask() {
          if (simpleCancellableTask == null) {
              // write pre-init user code here
              simpleCancellableTask = new SimpleCancellableTask();
              simpleCancellableTask.setExecutable(new
org.netbeans.microedition.util.Executable() {
                  public void execute() throws Exception {
                      // write task-execution user code here
                  }
              });
              // write post-init user code here
          }
          return simpleCancellableTask;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm03 ">
      /**
       * Returns an initiliazed instance of medicalRecordForm03 component.
       * @return the initialized component instance
       */
      public Form getMedicalRecordForm03() {
          if (medicalRecordForm03 == null) {
              // write pre-init user code here
              medicalRecordForm03 = new
Form("\u0399\u03C3\u03C4\u03BF\u03C1\u03B9\u03BA\u03CC -
\u0391\u03BD\u03C4\u03B9\u03BA\u03B5\u03B9\u03BC\u03B5\u03BD\u03B9\u03BA\u03AE \
u0395\u03BE\u03AD\u03C4\u03B1\u03C3\u03B7", new Item[]
{ getHistoryExamEvaluation() });
              medicalRecordForm03.addCommand(getBackCommand());
              medicalRecordForm03.addCommand(getNextCommand());
              medicalRecordForm03.addCommand(getOkCommand());
              medicalRecordForm03.addCommand(getCancelCommand());
              medicalRecordForm03.setCommandListener(this);
              // write post-init user code here
  //
getNearestRelativeSurname().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size() / 2);
  //
getNearestRelativeName().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size() / 2);
  //
getNearestRelativePhoneNumber().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size() / 2);
  //
getNearestRelativeRelationship().setPreferredSize(medicalRecordForm1.getWidth(),
medicalRecordForm1.getHeight() / medicalRecordForm1.size());
          }
          return medicalRecordForm03;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
nextCommand ">
      /**
       * Returns an initiliazed instance of nextCommand component.
       * @return the initialized component instance
```

```
     */
    public Command getNextCommand() {
        if (nextCommand == null) {
            // write pre-init user code here
            nextCommand = new
Command("\u0395\u03C0\u03CC\u03BC\u03B5\u03BD\u03BF", Command.SCREEN, 0);
            // write post-init user code here
        }
        return nextCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm04 ">
    /**
     * Returns an initiliazed instance of medicalRecordForm04 component.
     * @return the initialized component instance
     */
    public Form getMedicalRecordForm04() {
        if (medicalRecordForm04 == null) {
            // write pre-init user code here
            medicalRecordForm04 = new
Form("\u03A0\u03BF\u03C1\u03B5\u03AF\u03B1 \u039D\u03CC\u03C3\u03BF\u03C5", new
Item[] { getDiceaseProgress() });
            medicalRecordForm04.addCommand(getBackCommand());
            medicalRecordForm04.addCommand(getOkCommand());
            medicalRecordForm04.addCommand(getNextCommand());
            medicalRecordForm04.addCommand(getCancelCommand());
            medicalRecordForm04.setCommandListener(this);
            // write post-init user code here
        }
        return medicalRecordForm04;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm06 ">
    /**
     * Returns an initiliazed instance of medicalRecordForm06 component.
     * @return the initialized component instance
     */
    public Form getMedicalRecordForm06() {
        if (medicalRecordForm06 == null) {
            // write pre-init user code here
            medicalRecordForm06 = new
Form("\u0398\u03B5\u03C1\u03B1\u03C0\u03B5\u03C5\u03C4\u03B9\u03BA\u03AE
\u0391\u03B3\u03C9\u03B3\u03AE -
\u0395\u03C0\u03B5\u03BC\u03B2\u03AC\u03C3\u03B5\u03B9\u03C2", new Item[]
{ getTreatmentIntervention() });
            medicalRecordForm06.addCommand(getBackCommand());
            medicalRecordForm06.addCommand(getNextCommand());
            medicalRecordForm06.addCommand(getOkCommand());
            medicalRecordForm06.addCommand(getCancelCommand());
            medicalRecordForm06.setCommandListener(this);
            // write post-init user code here
        }
        return medicalRecordForm06;
    }
    //</editor-fold>
```

```
      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm05 ">
      /**
       * Returns an initiliazed instance of medicalRecordForm05 component.
       * @return the initialized component instance
       */
      public Form getMedicalRecordForm05() {
          if (medicalRecordForm05 == null) {
              // write pre-init user code here
              medicalRecordForm05 = new
Form("\u0394\u03B9\u03AC\u03B3\u03BD\u03C9\u03C3\u03B7
\u0395\u03BE\u03CC\u03B4\u03BF\u03C5", new Item[] { getExitDiagnosis() });
              medicalRecordForm05.addCommand(getBackCommand());
              medicalRecordForm05.addCommand(getNextCommand());
              medicalRecordForm05.addCommand(getOkCommand());
              medicalRecordForm05.addCommand(getCancelCommand());
              medicalRecordForm05.setCommandListener(this);
              // write post-init user code here
          }
          return medicalRecordForm05;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm07 ">
      /**
       * Returns an initiliazed instance of medicalRecordForm07 component.
       * @return the initialized component instance
       */
      public Form getMedicalRecordForm07() {
          if (medicalRecordForm07 == null) {
              // write pre-init user code here
              medicalRecordForm07 = new
Form("\u039F\u03B4\u03B7\u03B3\u03AF\u03B5\u03C2 \u03BA\u03B1\u03C4\u03AC
\u03C4\u03B7\u03BD \u03AD\u03BE\u03BF\u03B4\u03BF -
\u03A0\u03B1\u03C1\u03B1\u03C4\u03B7\u03C1\u03AE\u03C3\u03B5\u03B9\u03C2", new
Item[] { getExitDirectivesComments() });
              medicalRecordForm07.addCommand(getBackCommand());
              medicalRecordForm07.addCommand(getNextCommand());
              medicalRecordForm07.addCommand(getOkCommand());
              medicalRecordForm07.addCommand(getCancelCommand());
              medicalRecordForm07.setCommandListener(this);
              // write post-init user code here
          }
          return medicalRecordForm07;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm08 ">
      /**
       * Returns an initiliazed instance of medicalRecordForm08 component.
       * @return the initialized component instance
       */
      public Form getMedicalRecordForm08() {
          if (medicalRecordForm08 == null) {
              // write pre-init user code here
              medicalRecordForm08 = new
Form("\u0395\u03BE\u03B5\u03C4\u03AC\u03C3\u03B5\u03B9\u03C2", new Item[]
{ getEcg(), getRadiography(), getOtherExams() });
              medicalRecordForm08.addCommand(getBackCommand());
```

```
            medicalRecordForm08.addCommand(getOkCommand());
            medicalRecordForm08.addCommand(getNextCommand());
            medicalRecordForm08.addCommand(getCancelCommand());
            medicalRecordForm08.setCommandListener(this);
            // write post-init user code here
  //            getEcg().setPreferredSize(medicalRecordForm7.getWidth(),
medicalRecordForm7.getHeight() / 3);
  //            getRadiography().setPreferredSize(medicalRecordForm7.getWidth(),
medicalRecordForm7.getHeight() / 3);
  //            getOtherExams().setPreferredSize(medicalRecordForm7.getWidth(),
medicalRecordForm7.getHeight() / 3);
          }
          return medicalRecordForm08;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalRecordForm02 ">
      /**
       * Returns an initiliazed instance of medicalRecordForm02 component.
       * @return the initialized component instance
       */
      public Form getMedicalRecordForm02() {
          if (medicalRecordForm02 == null) {
              // write pre-init user code here
              medicalRecordForm02 = new
Form("\u03A3\u03C4\u03BF\u03B9\u03C7\u03B5\u03AF\u03B1
\u03A0\u03BB\u03B7\u03C3\u03B9\u03AD\u03C3\u03C4\u03B5\u03C1\u03BF\u03C5
\u03A3\u03C5\u03B3\u03B3\u03B5\u03BD\u03AE", new Item[]
{ getNearestRelativeSurname(), getNearestRelativeName(),
getNearestRelativePhoneNumber(), getNearestRelativeRelationship() });
              medicalRecordForm02.addCommand(getBackCommand());
              medicalRecordForm02.addCommand(getOkCommand());
              medicalRecordForm02.addCommand(getNextCommand());
              medicalRecordForm02.addCommand(getCancelCommand());
              medicalRecordForm02.setCommandListener(this);
              // write post-init user code here
          }
          return medicalRecordForm02;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
nearestRelativeSurname ">
      /**
       * Returns an initiliazed instance of nearestRelativeSurname component.
       * @return the initialized component instance
       */
      public TextField getNearestRelativeSurname() {
          if (nearestRelativeSurname == null) {
              // write pre-init user code here
              nearestRelativeSurname = new
TextField("\u0395\u03C0\u03CE\u03BD\u03C5\u03BC\u03BF", null, 16,
TextField.ANY);
              nearestRelativeSurname.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              nearestRelativeSurname.setString(getMedicalRecord().getNearestRela
tiveSurname());
          }
          return nearestRelativeSurname;
      }
```

```
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
nearestRelativeName ">
      /**
       * Returns an initiliazed instance of nearestRelativeName component.
       * @return the initialized component instance
       */
      public TextField getNearestRelativeName() {
          if (nearestRelativeName == null) {
              // write pre-init user code here
              nearestRelativeName = new
TextField("\u038C\u03BD\u03BF\u03BC\u03B1", null, 16, TextField.ANY);
              nearestRelativeName.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              nearestRelativeName.setString(getMedicalRecord().getNearestRelativ
eName());
          }
          return nearestRelativeName;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
nearestRelativePhoneNumber ">
      /**
       * Returns an initiliazed instance of nearestRelativePhoneNumber
component.
       * @return the initialized component instance
       */
      public TextField getNearestRelativePhoneNumber() {
          if (nearestRelativePhoneNumber == null) {
              // write pre-init user code here
              nearestRelativePhoneNumber = new
TextField("\u03A4\u03B7\u03BB\u03AD\u03C6\u03C9\u03BD\u03BF", null, 16,
TextField.PHONENUMBER);
              nearestRelativePhoneNumber.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              nearestRelativePhoneNumber.setString(getMedicalRecord().getNearest
RelativePhoneNumber());
          }
          return nearestRelativePhoneNumber;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
nearestRelativeRelationship ">
      /**
       * Returns an initiliazed instance of nearestRelativeRelationship
component.
       * @return the initialized component instance
       */
      public TextField getNearestRelativeRelationship() {
          if (nearestRelativeRelationship == null) {
              // write pre-init user code here
              nearestRelativeRelationship = new
TextField("\u03A3\u03C5\u03B3\u03B3\u03AD\u03BD\u03B5\u03B9\u03B1", null, 32,
TextField.ANY);
              nearestRelativeRelationship.setInitialInputMode("UCB_GREEK");
              // write post-init user code here
              nearestRelativeRelationship.setString(getMedicalRecord().getNeares
tRelativeRelationship());
```

```
            }
            return nearestRelativeRelationship;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
labExamsForm01 ">
        /**
         * Returns an initiliazed instance of labExamsForm01 component.
         * @return the initialized component instance
         */
        public Form getLabExamsForm01() {
            if (labExamsForm01 == null) {
                // write pre-init user code here
                labExamsForm01 = new
Form("\u0395\u03BE\u03B5\u03C4\u03AC\u03C3\u03B5\u03B9\u03C2 1/2", new Item[]
{ getTableItem() });
                labExamsForm01.addCommand(getBackCommand());
                labExamsForm01.addCommand(getOkCommand());
                labExamsForm01.addCommand(getNextCommand());
                labExamsForm01.addCommand(getCancelCommand());
                labExamsForm01.setCommandListener(this);
                // write post-init user code here
  //              getTableItem().setPreferredSize(medicalRecordForm9.getWidth(),
medicalRecordForm9.getHeight());
            }
            return labExamsForm01;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter: tableItem
">
        /**
         * Returns an initiliazed instance of tableItem component.
         * @return the initialized component instance
         */
        public TableItem getTableItem() {
            if (tableItem == null) {
                // write pre-init user code here
                tableItem = new TableItem(getDisplay(), null);
                // write post-init user code here
                tableItem.setModel(getLabExamsTable01());
            }
            return tableItem;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
historyExamEvaluation ">
        /**
         * Returns an initiliazed instance of historyExamEvaluation component.
         * @return the initialized component instance
         */
        public StringItem getHistoryExamEvaluation() {
            if (historyExamEvaluation == null) {
                // write pre-init user code here
                historyExamEvaluation = new StringItem(null, null);
                // write post-init user code here
                historyExamEvaluation.setText(getMedicalRecord().getHistoryExamEva
luation());
            }
```

```
            return historyExamEvaluation;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter: amka ">
        /**
         * Returns an initiliazed instance of amka component.
         * @return the initialized component instance
         */
        public StringItem getAmka() {
            if (amka == null) {
                // write pre-init user code here
                amka = new StringItem("\u0391\u039C\u039A\u0391", null);
                amka.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER |
Item.LAYOUT_EXPAND | Item.LAYOUT_2);
                // write post-init user code here
                amka.setText(getMedicalRecord().getAmka());
            }
            return amka;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
diceaseProgress ">
        /**
         * Returns an initiliazed instance of diceaseProgress component.
         * @return the initialized component instance
         */
        public StringItem getDiceaseProgress() {
            if (diceaseProgress == null) {
                // write pre-init user code here
                diceaseProgress = new StringItem(null, null);
                // write post-init user code here
                diceaseProgress.setText(getMedicalRecord().getDiceaseProgress());
            }
            return diceaseProgress;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
exitDiagnosis ">
        /**
         * Returns an initiliazed instance of exitDiagnosis component.
         * @return the initialized component instance
         */
        public StringItem getExitDiagnosis() {
            if (exitDiagnosis == null) {
                // write pre-init user code here
                exitDiagnosis = new StringItem(null, null);
                // write post-init user code here
                exitDiagnosis.setText(getMedicalRecord().getExitDiagnosis());
            }
            return exitDiagnosis;
        }
        //</editor-fold>

        //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
treatmentIntervention ">
        /**
         * Returns an initiliazed instance of treatmentIntervention component.
```

```
       * @return the initialized component instance
       */
      public StringItem getTreatmentIntervention() {
          if (treatmentIntervention == null) {
              // write pre-init user code here
              treatmentIntervention = new StringItem(null, null);
              // write post-init user code here
              treatmentIntervention.setText(getMedicalRecord().getTreatmentInter
vention());
          }
          return treatmentIntervention;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
exitDirectivesComments ">
      /**
       * Returns an initiliazed instance of exitDirectivesComments component.
       * @return the initialized component instance
       */
      public StringItem getExitDirectivesComments() {
          if (exitDirectivesComments == null) {
              // write pre-init user code here
              exitDirectivesComments = new StringItem(null, null);
              // write post-init user code here
              exitDirectivesComments.setText(getMedicalRecord().getExitDirective
sComments());
          }
          return exitDirectivesComments;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter: ecg ">
      /**
       * Returns an initiliazed instance of ecg component.
       * @return the initialized component instance
       */
      public StringItem getEcg() {
          if (ecg == null) {
              // write pre-init user code here
              ecg = new StringItem("\u0397\u039A\u0393", null);
              // write post-init user code here
              ecg.setText(getMedicalRecord().getEcg());
          }
          return ecg;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
radiography ">
      /**
       * Returns an initiliazed instance of radiography component.
       * @return the initialized component instance
       */
      public StringItem getRadiography() {
          if (radiography == null) {
              // write pre-init user code here
              radiography = new
StringItem("\u0391\u03BA\u03C4\u03B9\u03BD\u03BF\u03BB\u03BF\u03B3\u03B9\u03BA\u
03CC\u03C2 \u0388\u03BB\u03B5\u03B3\u03C7\u03BF\u03C2", null);
              // write post-init user code here
```

```
            radiography.setText(getMedicalRecord().getRadiography());
        }
        return radiography;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
otherExams ">
    /**
     * Returns an initiliazed instance of otherExams component.
     * @return the initialized component instance
     */
    public StringItem getOtherExams() {
        if (otherExams == null) {
            // write pre-init user code here
            otherExams = new
StringItem("\u039B\u03BF\u03B9\u03C0\u03AD\u03C2
\u0395\u03BE\u03B5\u03C4\u03AC\u03C3\u03B5\u03B9\u03C2", null);
            // write post-init user code here
            otherExams.setText(getMedicalRecord().getOtherExams());
        }
        return otherExams;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
loginScreen ">
    /**
     * Returns an initiliazed instance of loginScreen component.
     * @return the initialized component instance
     */
    public Form getLoginScreen() {
        if (loginScreen == null) {
            // write pre-init user code here
            loginScreen = new
Form("\u0399\u03B1\u03C4\u03C1\u03B9\u03BA\u03CC\u03C2
\u03A6\u03AC\u03BA\u03B5\u03BB\u03BF\u03C2", new Item[] { getPinField(),
getSpacer(), getPublicItem() });
            loginScreen.addCommand(getExitCommand());
            loginScreen.addCommand(getLoginCommand());
            loginScreen.setCommandListener(this);
            // write post-init user code here
  //            getSpacer().setPreferredSize(loginScreen.getWidth(),
getPinField().getPreferredHeight());
  //            getPublicItem().setPreferredSize(loginScreen.getWidth(),
loginScreen.getHeight() - getPinField().getPreferredHeight() -
getSpacer().getPreferredHeight());
        }
        return loginScreen;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
loginCommand ">
    /**
     * Returns an initiliazed instance of loginCommand component.
     * @return the initialized component instance
     */
    public Command getLoginCommand() {
        if (loginCommand == null) {
            // write pre-init user code here
```

```
            loginCommand = new
Command("\u0395\u03AF\u03C3\u03BF\u03B4\u03BF\u03C2", Command.OK, 0);
            // write post-init user code here
        }
        return loginCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: spacer ">
    /**
     * Returns an initiliazed instance of spacer component.
     * @return the initialized component instance
     */
    public Spacer getSpacer() {
        if (spacer == null) {
            // write pre-init user code here
            spacer = new Spacer(1, 1);
            // write post-init user code here
        }
        return spacer;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: pinField
">
    /**
     * Returns an initiliazed instance of pinField component.
     * @return the initialized component instance
     */
    public TextField getPinField() {
        if (pinField == null) {
            // write pre-init user code here
            pinField = new TextField("PIN", "1234", 4, TextField.NUMERIC |
TextField.PASSWORD);
            pinField.setLayout(ImageItem.LAYOUT_CENTER | Item.LAYOUT_TOP |
Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_EXPAND |
Item.LAYOUT_2);
            // write post-init user code here
        }
        return pinField;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
labExamsForm02 ">
    /**
     * Returns an initiliazed instance of labExamsForm02 component.
     * @return the initialized component instance
     */
    public Form getLabExamsForm02() {
        if (labExamsForm02 == null) {
            // write pre-init user code here
            labExamsForm02 = new
Form("\u0395\u03BE\u03B5\u03C4\u03AC\u03C3\u03B5\u03B9\u03C2 2/2", new Item[]
{ getTableItem1() });
            labExamsForm02.addCommand(getBackCommand());
            labExamsForm02.addCommand(getOkCommand());
            labExamsForm02.addCommand(getCancelCommand());
            labExamsForm02.setCommandListener(this);
            // write post-init user code here
```

```
  //            getTableItem1().setPreferredSize(medicalRecordForm10.getWidth(),
medicalRecordForm10.getHeight());
          }
          return labExamsForm02;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
tableItem1 ">
      /**
       * Returns an initiliazed instance of tableItem1 component.
       * @return the initialized component instance
       */
      public TableItem getTableItem1() {
          if (tableItem1 == null) {
              // write pre-init user code here
              tableItem1 = new TableItem(getDisplay(), null);
              // write post-init user code here
              tableItem1.setModel(getLabExamsTable02());
          }
          return tableItem1;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
publicItem ">
      /**
       * Returns an initiliazed instance of publicItem component.
       * @return the initialized component instance
       */
      public StringItem getPublicItem() {
          if (publicItem == null) {
              // write pre-init user code here
              String text = "";
              int[] recordIds;
              publicItem = new StringItem("", "");
              publicItem.setLayout(ImageItem.LAYOUT_LEFT | Item.LAYOUT_TOP |
Item.LAYOUT_VCENTER | ImageItem.LAYOUT_NEWLINE_BEFORE | Item.LAYOUT_EXPAND |
Item.LAYOUT_VEXPAND | Item.LAYOUT_2);
              // write post-init user code here
              if (getRMS().getNumRecords(EmergencyRecord.RECORD_DATA_TYPE) > 0
                      && (recordIds =
getRMS().getRecordIds(EmergencyRecord.RECORD_DATA_TYPE)) != null
                      && recordIds.length == 1 &&
getRMS().getRecord(recordIds[0]) != null) {
                  recordId = recordIds[0];
                  getEmergencyRecord().fromByteArray(getRMS().getRecord(getEmerg
encyRecord().getRecordId()));
                  text = getEmergencyRecord().getEmergency();
              }
              publicItem.setText(text);
          }
          return publicItem;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
cancelCommand ">
      /**
       * Returns an initiliazed instance of cancelCommand component.
       * @return the initialized component instance
```

```
     */
    public Command getCancelCommand() {
        if (cancelCommand == null) {
            // write pre-init user code here
            cancelCommand = new
Command("\u0391\u03BA\u03CD\u03C1\u03C9\u03C3\u03B7", Command.CANCEL, 0);
            // write post-init user code here
        }
        return cancelCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
deleteRecordAlert ">
    /**
     * Returns an initiliazed instance of deleteRecordAlert component.
     * @return the initialized component instance
     */
    public Alert getDeleteRecordAlert() {
        if (deleteRecordAlert == null) {
            // write pre-init user code here
            deleteRecordAlert = new
Alert("\u0395\u03B9\u03B4\u03BF\u03C0\u03BF\u03AF\u03B7\u03C3\u03B7",
"\u03A0\u03C1\u03BF\u03C3\u03BF\u03C7\u03AE: \u039C\u03B5 \u03C4\u03B7
\u03B4\u03B9\u03B1\u03B3\u03C1\u03B1\u03C6\u03AE \u03C4\u03BF\u03C5
\u03B9\u03B1\u03C4\u03C1\u03B9\u03BA\u03BF\u03CD
\u03C6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C5 \u03B8\u03B1
\u03C7\u03B1\u03B8\u03BF\u03CD\u03BD \u03CC\u03BB\u03B1 \u03C4\u03B1
\u03B4\u03B5\u03B4\u03BF\u03BC\u03AD\u03BD\u03B1!", null,
AlertType.CONFIRMATION);
            deleteRecordAlert.addCommand(getCancelCommand());
            deleteRecordAlert.addCommand(getContinueCommand());
            deleteRecordAlert.setCommandListener(this);
            deleteRecordAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return deleteRecordAlert;
    }
    //</editor-fold>
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
continueCommand ">
    /**
     * Returns an initiliazed instance of continueCommand component.
     * @return the initialized component instance
     */
    public Command getContinueCommand() {
        if (continueCommand == null) {
            // write pre-init user code here
            continueCommand = new
Command("\u03A3\u03C5\u03BD\u03AD\u03C7\u03B5\u03B9\u03B1", Command.SCREEN, 0);
            // write post-init user code here
        }
        return continueCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isFirstTime ">
    /**
```

```
       * Performs an action assigned to the isFirstTime if-point.
       */
      public void isFirstTime() {
          // enter pre-if user code here
          boolean isSuccess = true;
          int numRecords =
getRMS().getNumRecords(MedicalRecord.RECORD_DATA_TYPE);
          System.out.println("numRecords=" + numRecords);
          if (numRecords > 1) {
              getRMS().deleteRecord(MedicalRecord.RECORD_DATA_TYPE);
          } else if (numRecords == 1) {
              System.out.println("RecordStore valid.");
              isSuccess = false;
          }
          if (isSuccess) {
              // write pre-action user code here
              switchDisplayable(null, getInitialScreen());
              // write post-action user code here
          } else {
              // write pre-action user code here
              switchDisplayable(null, getLoginScreen());
              // write post-action user code here
          }
          // enter post-if user code here
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
InitialScreen ">
      /**
       * Returns an initiliazed instance of InitialScreen component.
       * @return the initialized component instance
       */
      public Form getInitialScreen() {
          if (InitialScreen == null) {
              // write pre-init user code here
              InitialScreen = new Form("\u039D\u03AD\u03BF\u03C2
\u03A6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C2", new Item[] { getSpacer1(),
getInitialAmkaTextField(), getInitialPinTextField() });
              InitialScreen.addCommand(getLoginCommand());
              InitialScreen.addCommand(getExitCommand());
              InitialScreen.setCommandListener(this);
              // write post-init user code here
  //            getSpacer1().setPreferredSize(InitialForm.getWidth(),
InitialForm.getHeight() / 3);
          }
          return InitialScreen;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
initialPinTextField ">
      /**
       * Returns an initiliazed instance of initialPinTextField component.
       * @return the initialized component instance
       */
      public TextField getInitialPinTextField() {
          if (initialPinTextField == null) {
              // write pre-init user code here
              initialPinTextField = new TextField("PIN", "1234", 4,
TextField.NUMERIC | TextField.PASSWORD);
```

```
            initialPinTextField.setLayout(ImageItem.LAYOUT_CENTER |
Item.LAYOUT_TOP | Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | Item.LAYOUT_EXPAND
| Item.LAYOUT_2);
            // write post-init user code here
        }
        return initialPinTextField;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
initialAmkaTextField ">
    /**
     * Returns an initiliazed instance of initialAmkaTextField component.
     * @return the initialized component instance
     */
    public TextField getInitialAmkaTextField() {
        if (initialAmkaTextField == null) {
            // write pre-init user code here
            initialAmkaTextField = new TextField("AMKA", "12392834091", 32,
TextField.NUMERIC);
            initialAmkaTextField.setLayout(ImageItem.LAYOUT_CENTER |
Item.LAYOUT_TOP | Item.LAYOUT_BOTTOM | Item.LAYOUT_VCENTER | Item.LAYOUT_EXPAND
| Item.LAYOUT_2);
            // write post-init user code here
        }
        return initialAmkaTextField;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isAmkaValid ">
    /**
     * Performs an action assigned to the isAmkaValid if-point.
     */
    public void isAmkaValid() {
        boolean isSuccess = true;
        // enter pre-if user code here
        String text = getInitialAmkaTextField().getString();
        if (text != null && text.length() == 11) {
            for (int i = 0; i < text.length(); i++) {
                if (!Character.isDigit(text.charAt(i))) {
                    isSuccess = false;
                    break;
                }
            }
        } else {
            isSuccess = false;
        }
        if (isSuccess) {
            // write pre-action user code here
            isInitialPinValid();
            // write post-action user code here
        } else {
            // write pre-action user code here
            switchDisplayable(getAmkaInvalidAlert(), getInitialScreen());
            // write post-action user code here
        }
        // enter post-if user code here
    }
    //</editor-fold>
```

```
      //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
amkaInvalidAlert ">
      /**
       * Returns an initiliazed instance of amkaInvalidAlert component.
       * @return the initialized component instance
       */
      public Alert getAmkaInvalidAlert() {
          if (amkaInvalidAlert == null) {
              // write pre-init user code here
              amkaInvalidAlert = new
Alert("\u03A3\u03C6\u03AC\u03BB\u03BC\u03B1", "\u039C\u03B7
\u03B1\u03C0\u03BF\u03B4\u03B5\u03BA\u03C4\u03CC\u03C2
\u0391\u039C\u039A\u0391", getAlertImage(), AlertType.ERROR);
              amkaInvalidAlert.setTimeout(Alert.FOREVER);
              // write post-init user code here
          }
          return amkaInvalidAlert;
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Method:
isInitialPinValid ">
      /**
       * Performs an action assigned to the isInitialPinValid if-point.
       */
      public void isInitialPinValid() {
          // enter pre-if user code here
          boolean isSuccess = false;
          password = getInitialPinTextField().getString();
          if (checkPassword()) {
              MedicalRecord mr = new MedicalRecord(new Hashtable());
              mr.setAmka(getInitialAmkaTextField().getString());
              mr.setDateL(System.currentTimeMillis());
              try {
                  recordId = getRMS().addRecord(MedicalRecord.RECORD_DATA_TYPE,
mr.toByteArray());
              } catch (Exception ex) {
                  ex.printStackTrace();
              }
          }
          if (recordId > 0) {
              isSuccess = true;
              System.out.println("addMedicalRecord, recordId=" + recordId +
".");
          }

          if (isSuccess) {
              // write pre-action user code here
              switchDisplayable(null, getMainMenuList());
              // write post-action user code here
          } else {
              // write pre-action user code here
              switchDisplayable(getPinInvalidAlert(), getInitialScreen());
              // write post-action user code here
          }
          // enter post-if user code here
      }
      //</editor-fold>

      //<editor-fold defaultstate="collapsed" desc=" Generated Getter: spacer1
">
```

```
    /**
     * Returns an initiliazed instance of spacer1 component.
     * @return the initialized component instance
     */
    public Spacer getSpacer1() {
        if (spacer1 == null) {
            // write pre-init user code here
            spacer1 = new Spacer(16, 1);
            // write post-init user code here
        }
        return spacer1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
medicalrecordIcon ">
    /**
     * Returns an initiliazed instance of medicalrecordIcon component.
     * @return the initialized component instance
     */
    public Image getMedicalrecordIcon() {
        if (medicalrecordIcon == null) {
            // write pre-init user code here
            try {
                medicalrecordIcon =
Image.createImage("/org/ntua/mobile/medicalrecord/images/medicalrecord.png");
            } catch (java.io.IOException e) {
                e.printStackTrace();
            }
            // write post-init user code here
        }
        return medicalrecordIcon;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
deleteCommand ">
    /**
     * Returns an initiliazed instance of deleteCommand component.
     * @return the initialized component instance
     */
    public Command getDeleteCommand() {
        if (deleteCommand == null) {
            // write pre-init user code here
            deleteCommand = new
Command("\u0394\u03B9\u03B1\u03B3\u03C1\u03B1\u03C6\u03AE", Command.SCREEN, 0);
            // write post-init user code here
        }
        return deleteCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
deleteInvalidRecordAlert ">
    /**
     * Returns an initiliazed instance of deleteInvalidRecordAlert component.
     * @return the initialized component instance
     */
    public Alert getDeleteInvalidRecordAlert() {
        if (deleteInvalidRecordAlert == null) {
            // write pre-init user code here
```

```
            deleteInvalidRecordAlert = new Alert("alert",
"\u03A0\u03C1\u03BF\u03C3\u03BF\u03C7\u03AE: \u039C\u03B5 \u03C4\u03B7
\u03B4\u03B9\u03B1\u03B3\u03C1\u03B1\u03C6\u03AE \u03C4\u03BF\u03C5
\u03B9\u03B1\u03C4\u03C1\u03B9\u03BA\u03BF\u03CD
\u03C6\u03B1\u03BA\u03AD\u03BB\u03BF\u03C5 \u03B8\u03B1
\u03C7\u03B1\u03B8\u03BF\u03CD\u03BD \u03CC\u03BB\u03B1 \u03C4\u03B1
\u03B4\u03B5\u03B4\u03BF\u03BC\u03AD\u03BD\u03B1!", getAlertImage(), null);
            deleteInvalidRecordAlert.addCommand(getCancelCommand());
            deleteInvalidRecordAlert.addCommand(getContinueCommand());
            deleteInvalidRecordAlert.setCommandListener(this);
            deleteInvalidRecordAlert.setTimeout(Alert.FOREVER);
            // write post-init user code here
        }
        return deleteInvalidRecordAlert;
    }
    //</editor-fold>

    // <editor-fold defaultstate="collapsed" desc="LabEmaxs">
    /**
     * Table for Lab Exams Results 1/2
     * @return initialized table
     */
    public SimpleTableModel getLabExamsTable01() {
        if (labExams1 == null) {
            labExams1 = new SimpleTableModel(new String[][]{
                        new java.lang.String[]{"       ΠΕΔΙΟ       ", "
TIMH       "},
                        new java.lang.String[]{"Αιματοκρίτης",
getMedicalRecord().getLabHematocrit()},
                        new java.lang.String[]{"Λευκά",
getMedicalRecord().getLabWhite()},
                        new java.lang.String[]{"Αιμοπετάλια",
getMedicalRecord().getLabPlatelets()},
                        new java.lang.String[]{"ΤΚΕ",
getMedicalRecord().getLabTke()},
                        new java.lang.String[]{"Αιμοσφαιρίνη",
getMedicalRecord().getLabHaemoglobin()},
                        new java.lang.String[]{"Πολ.",
getMedicalRecord().getLabPol()},
                        new java.lang.String[]{"Λεμφ.",
getMedicalRecord().getLabLemf()},
                        new java.lang.String[]{"Μον.",
getMedicalRecord().getLabMon()},
                        new java.lang.String[]{"Ηωσ.",
getMedicalRecord().getLabHws()},
                        new java.lang.String[]{"ΔΕΚ",
getMedicalRecord().getLabDek()},}, null);
        }
        return labExams1;
    }

    /**
     * Table for Lab Exams Results 2/2
     * @return initialized table
     */
    public SimpleTableModel getLabExamsTable02() {
        if (labExams2 == null) {
            labExams2 = new SimpleTableModel(new String[][]{
                        new java.lang.String[]{"       ΠΕΔΙΟ       ", "
TIMH       "},
```

```
                        new java.lang.String[]{"Σάκχαρο",
getMedicalRecord().getLabGlucose()},
                        new java.lang.String[]{"Κρεατινίνη",
getMedicalRecord().getLabCreatinine()},
                        new java.lang.String[]{"K",
getMedicalRecord().getLabK()},
                        new java.lang.String[]{"Na",
getMedicalRecord().getLabNa()},
                        new java.lang.String[]{"Ca",
getMedicalRecord().getLabCa()},
                        new java.lang.String[]{"Χολερυθρίνη ολ.",
getMedicalRecord().getLabBilirubinTotal()},
                        new java.lang.String[]{"Χολερυθρίνη αμ.",
getMedicalRecord().getLabBilirubinDirect()},
                        new java.lang.String[]{"SGOT",
getMedicalRecord().getLabSgot()},
                        new java.lang.String[]{"SGPT",
getMedicalRecord().getLabSgpt()},
                        new java.lang.String[]{"γ – GT",
getMedicalRecord().getLabGgt()},
                        new java.lang.String[]{"Αλκαλ. Φωσφατ.",
getMedicalRecord().getLabAlp()},
                        new java.lang.String[]{"CPK",
getMedicalRecord().getLabCpk()},
                        new java.lang.String[]{"LDH",
getMedicalRecord().getLabLdh()}
                    }, null);
        }
        return labExams2;
    }
    // </editor-fold>

    /**
     * Returns a display instance.
     * @return the display instance.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
     * Exits MIDlet.
     */
    public void exitMIDlet() {
        switchDisplayable(null, null);
        destroyApp(true);
        notifyDestroyed();

    }

    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and
initialize/starts or resumes the MIDlet.
     */
    public void startApp() {
        System.out.println("startApp.");
        if (isMidletPaused()) {
            resumeMIDlet();
        } else {
            initialize();
```

```
            startMIDlet();
        }
        setMidletPaused(false);
    }

    /**
     * Called when MIDlet is paused.
     */
    public void pauseApp() {
//        setMessagingCancelled(true);
        System.out.println("pauseApp.");
        setMidletPaused(true);
    }

    /**
     * Called to signal the MIDlet to terminate.
     * @param unconditional if true, then the MIDlet has to be unconditionally
terminated and all resources has to be released.
     */
    public void destroyApp(boolean unconditional) {
        System.out.println("dstroyApp.");
        if (unconditional) {
            getRMS().closeRecordStores();
//          closeSmsConnections();
        }
    }

    private boolean checkPassword() {
        boolean isValid = true;
        if (password != null && password.length() >= 4) {
            for (int i = 0; i < password.length(); i++) {
                if (!Character.isDigit(password.charAt(i)) && !
Character.isLowerCase(password.charAt(i)) && !
Character.isUpperCase(password.charAt(i))) {
                    isValid = false;
                    break;
                }
            }
        } else {
            isValid = false;
        }
        return isValid;
    }

    public RMS getRMS() {
        if (this.rms == null) {
            this.rms = new RMS(password);
        } else {
            rms.setPassword(password);
        }
        return rms;
    }

    public OBEXServer getObexServer() {
        if (obexServer == null) {
            obexServer = new OBEXServer(this);
        }
        return obexServer;
    }

    public void setObexServer(OBEXServer obexServer) {
```

```
    this.obexServer = obexServer;
}

/**
 *
 * @return
 */
public MedicalRecord getMedicalRecord() {
    if (medicalRecord == null) {
        medicalRecord = new MedicalRecord(new Hashtable());
        medicalRecord.setRecordId(recordId);
    }
    return medicalRecord;
}

/**
 *
 * @return
 */
public EmergencyRecord getEmergencyRecord() {
    if (emergencyRecord == null) {
        emergencyRecord = new EmergencyRecord(new Hashtable());
        emergencyRecord.setRecordId(recordId);
    }
    return emergencyRecord;
}

/**
 *
 * @return
 */
public int getRecordId() {
    return recordId;
}

/**
 *
 * @return
 */
public boolean isMidletPaused() {
    return midletPaused;
}

/**
 *
 * @param midletPaused
 */
public void setMidletPaused(boolean midletPaused) {
    this.midletPaused = midletPaused;
}

Executable bluetoothUpdateExecutable() {
    return new Executable() {

        public void execute() throws Exception {
            System.out.println("bluetoothUpdateExecutable 0.");
            try {
                getObexServer().startServer();
                System.out.println("bluetoothUpdateExecutable 1.");
            } catch (Exception ex) {
                ex.printStackTrace();
```

```
                    if (ex.getMessage() != null) {
                        System.out.println("bluetoothUpdateExecutable 2.");
                        getUpdateMedicalRecordFailAlert().setString(ex.getMess
age());
                    } else {
                        System.out.println("bluetoothUpdateExecutable 3.");
                        getUpdateMedicalRecordFailAlert().setString(ex.toStrin
g());
                    }
                    throw new Exception();
                }
            }
        };
    }

    public void generatePassword() {
        if (passwordGenerator == null) {
            passwordGenerator = new CodeGenerator();
        }
    }
}
```

```
/*
 * EmergencyRecord.java
 */
package org.ntua.mobile.medicalrecord.storage;

import java.util.Hashtable;

/**
 * This class contains all possible attributes of a Medical Record.
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class EmergencyRecord extends GenericRecord {

    public static final String RECORD_DATA_TYPE = RMS.EMERGENCY;

    public EmergencyRecord(Hashtable dataTable) {
        super(dataTable);
    }

    public String getEmergency() {
        return getDataTable().get("emergency") != null ?
getDataTable().get("emergency").toString() : "";
    }

    public void setEmergency(String emergency) {
        getDataTable().put("emergency", emergency);
    }

    public static final Hashtable dummyRecord() {
        Hashtable dummyH = new Hashtable();
        dummyH.put("emergency", "διαβητικός");
        return dummyH;
    }
}
```

```
/*
 * GenericRecord.java
 */
package org.ntua.mobile.medicalrecord.storage;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.Enumeration;
import java.util.Hashtable;

/**
 *
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class GenericRecord {

    private int recordId;
    private long dateL;
    private Hashtable dataTalbe = new Hashtable();

    public GenericRecord() {
    }

    public GenericRecord(Hashtable dataTable) {
        this.dataTalbe = dataTable;
        if (dataTable != null && dataTable.size() > 0) {
            convertNullToEmptyString();
        }
    }

    public long getDateL() {
        return dateL;
    }

    public void setDateL(long dateL) {
        this.dateL = dateL;
    }

    public int getRecordId() {
        return recordId;
    }

    public void setRecordId(int recordId) {
        this.recordId = recordId;
    }

    public Hashtable getDataTable() {
        return dataTalbe;
    }

    public void setDataTable(Hashtable dataTable) {
        this.dataTalbe = dataTable;
    }

    public byte[] toByteArray() {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(baos);
        String value;
```

```
            try {
  //              System.out.println("dateL = " + String.valueOf(dateL));
                dos.writeLong(dateL);
                Enumeration e = getDataTable().keys();
                while (e.hasMoreElements()) {
                    String key = (String) e.nextElement();
                    value = (String) getDataTable().get(key);
  //                  System.out.println("key=" + key + ", value=" + value);
  //                  if (value == null) {
  //                      System.out.println("\n\n NULL \n\n");
  //                  }
                    dos.writeUTF(key);
                    dos.writeUTF((String) getDataTable().get(key));
                }
  //              dos.flush();
            } catch (IOException iOException) {
                iOException.printStackTrace();
                baos.reset();
            }
            return baos.toByteArray();
        }

        public void fromByteArray(byte[] recordDAtaBs) {
            if (recordDAtaBs == null || recordDAtaBs.length == 0) {
                recordDAtaBs = new byte[0];
            }
            if (dataTalbe == null) {
                dataTalbe = new Hashtable();
            }
            ByteArrayInputStream bais = new ByteArrayInputStream(recordDAtaBs);
            DataInputStream dis = new DataInputStream(bais);
            try {
                dateL = dis.readLong();
  //              System.out.println("dateL = " + String.valueOf(dateL));
                while (dis.available() > 0) {
                    System.out.flush();
  //                  if (dataTalbe != null) {
                    dataTalbe.put(dis.readUTF(), dis.readUTF());
  //                  }
                }
            } catch (IOException iOException) {
                iOException.printStackTrace();
                dataTalbe.clear();
            }
        }

        public void print() {
            Enumeration e = getDataTable().keys();
            while (e.hasMoreElements()) {
                String key = (String) e.nextElement();
                System.out.println(key + " = " + getDataTable().get(key));
            }
        }

        private void convertNullToEmptyString() {
            Enumeration e = dataTalbe.keys();
            while (e.hasMoreElements()) {
                dataTalbe.put(e, dataTalbe.get(e) != null ? dataTalbe.get(e) :
"");
            }
        }}
```

```
/*
 * MedicalRecord.java
 */
package org.ntua.mobile.medicalrecord.storage;

import java.util.Hashtable;

/**
 * This class contains all possible attributes of a Medical Record.
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class MedicalRecord extends GenericRecord {

    public static final String RECORD_DATA_TYPE = RMS.I4_NO_LABS;

    public MedicalRecord(Hashtable dataTable) {
        super(dataTable);
    }

    public void setPatientAttrs(String surname, String name, String
bloodTypeRh, String address, String postalCode, String city, String phoneNumber,
             String nearestRelativeSurname, String nearestRelativeName, String
nearestRelativePhoneNumber, String nearestRelativeRelationship) {
        getDataTable().put("surname", surname);
        getDataTable().put("name", name);
        getDataTable().put("bloodTypeRh", bloodTypeRh);
        getDataTable().put("address", address);
        getDataTable().put("postalCode", postalCode);
        getDataTable().put("city", city);
        getDataTable().put("phoneNumber", phoneNumber);
        getDataTable().put("nearestRelativeSurname", nearestRelativeSurname);
        getDataTable().put("nearestRelativeName", nearestRelativeName);
        getDataTable().put("nearestRelativePhoneNumber",
nearestRelativePhoneNumber);
        getDataTable().put("nearestRelativeRelationship",
nearestRelativeRelationship);
    }

    public String getAmka() {
        return getDataTable().get("amka") != null ?
getDataTable().get("amka").toString() : "";
    }

    public void setAmka(String amka) {
        getDataTable().put("amka", amka);
    }

    public String getSurname() {
        return getDataTable().get("surname") != null ?
getDataTable().get("surname").toString() : "";
    }

    public void setSurname(String surname) {
        getDataTable().put("surname", surname);
    }

    public String getName() {
        return getDataTable().get("name") != null ?
getDataTable().get("name").toString() : "";
    }
```

```
      public void setName(String name) {
          getDataTable().put("name", name);
      }

      public String getBloodTypeRh() {
          return getDataTable().get("bloodTypeRh") != null ?
getDataTable().get("bloodTypeRh").toString() : "";
      }

      public void setBloodTypeRh(String bloodTypeRh) {
          getDataTable().put("bloodTypeRh", bloodTypeRh);
      }

      public String getAddress() {
          return getDataTable().get("address") != null ?
getDataTable().get("address").toString() : "";
      }

      public void setAddress(String address) {
          getDataTable().put("address", address);
      }

      public String getCity() {
          return getDataTable().get("city") != null ?
getDataTable().get("city").toString() : "";
      }

      public void setCity(String city) {
          getDataTable().put("city", city);
      }

      public String getPostalCode() {
          return getDataTable().get("postalCode") != null ?
getDataTable().get("postalCode").toString() : "";
      }

      public void setPostalCode(String postalCode) {
          getDataTable().put("postalCode", postalCode);
      }

      public String getPhoneNumber() {
          return getDataTable().get("phoneNumber") != null ?
getDataTable().get("phoneNumber").toString() : "";
      }

      public void setPhoneNumber(String phoneNumber) {
          getDataTable().put("phoneNumber", phoneNumber);
      }

      public String getDiceaseProgress() {
          return getDataTable().get("diceaseProgress") != null ?
getDataTable().get("diceaseProgress").toString() : "";
      }

      public void setDiceaseProgress(String diceaseProgress) {
          getDataTable().put("diceaseProgress", diceaseProgress);
      }

      public String getEcg() {
          return getDataTable().get("ecg") != null ?
getDataTable().get("ecg").toString() : "";
```

```
    }

    public void setEcg(String ecg) {
        getDataTable().put("ecg", ecg);
    }

    public String getExitDiagnosis() {
        return getDataTable().get("exitDiagnosis") != null ?
getDataTable().get("exitDiagnosis").toString() : "";
    }

    public void setExitDiagnosis(String exitDiagnosis) {
        getDataTable().put("exitDiagnosis", exitDiagnosis);
    }

    public String getExitDirectivesComments() {
        return getDataTable().get("exitDirectivesComments") != null ?
getDataTable().get("exitDirectivesComments").toString() : "";
    }

    public void setExitDirectivesComments(String exitDirectivesComments) {
        getDataTable().put("exitDirectivesComments", exitDirectivesComments);
    }

    public String getHistoryExamEvaluation() {
        return getDataTable().get("historyExamEvaluation") != null ?
getDataTable().get("historyExamEvaluation").toString() : "";
    }

    public void setHistoryExamEvaluation(String historyExamEvaluation) {
        getDataTable().put("historyExamEvaluation", historyExamEvaluation);
    }

    public String getNearestRelativeName() {
        return getDataTable().get("nearestRelativeName") != null ?
getDataTable().get("nearestRelativeName").toString() : "";
    }

    public void setNearestRelativeName(String nearestRelativeName) {
        getDataTable().put("nearestRelativeName", nearestRelativeName);
    }

    public String getNearestRelativePhoneNumber() {
        return getDataTable().get("nearestRelativePhoneNumber") != null ?
getDataTable().get("nearestRelativePhoneNumber").toString() : "";
    }

    public void setNearestRelativePhoneNumber(String
nearestRelativePhoneNumber) {
        getDataTable().put("nearestRelativePhoneNumber",
nearestRelativePhoneNumber);
    }

    public String getNearestRelativeRelationship() {
        return getDataTable().get("nearestRelativeRelationship") != null ?
getDataTable().get("nearestRelativeRelationship").toString() : "";
    }

    public void setNearestRelativeRelationship(String
nearestRelativeRelationship) {
```

```
            getDataTable().put("nearestRelativeRelationship",
nearestRelativeRelationship);
        }

        public String getNearestRelativeSurname() {
            return getDataTable().get("nearestRelativeSurname") != null ?
getDataTable().get("nearestRelativeSurname").toString() : "";
        }

        public void setNearestRelativeSurname(String nearestRelativeSurname) {
            getDataTable().put("nearestRelativeSurname", nearestRelativeSurname);
        }

        public String getOtherExams() {
            return getDataTable().get("otherExams") != null ?
getDataTable().get("otherExams").toString() : "";
        }

        public void setOtherExams(String otherExams) {
            getDataTable().put("otherExams", otherExams);
        }

        public String getRadiography() {
            return getDataTable().get("radiography") != null ?
getDataTable().get("radiography").toString() : "";
        }

        public void setRadiography(String radiography) {
            getDataTable().put("radiography", radiography);
        }

        public String getTreatmentIntervention() {
            return getDataTable().get("treatmentIntervention") != null ?
getDataTable().get("treatmentIntervention").toString() : "";
        }

        public void setTreatmentIntervention(String treatmentIntervention) {
            getDataTable().put("treatmentIntervention", treatmentIntervention);
        }

        public String getLabAlp() {
            return getDataTable().get("labAlp") != null ?
getDataTable().get("labAlp").toString() : "";
        }

        public void setLabAlp(String labAlp) {
            getDataTable().put("labAlp", labAlp);
        }

        public String getLabBilirubinDirect() {
            return getDataTable().get("labBilirubinDirect") != null ?
getDataTable().get("labBilirubinDirect").toString() : "";
        }

        public void setLabBilirubinDirect(String labBilirubinDirect) {
            getDataTable().put("labBilirubinDirect", labBilirubinDirect);
        }

        public String getLabBilirubinTotal() {
            return getDataTable().get("labBilirubinTotal") != null ?
getDataTable().get("labBilirubinTotal").toString() : "";
```

```
    }

    public void setLabBilirubinTotal(String labBilirubinTotal) {
        getDataTable().put("labBilirubinTotal", labBilirubinTotal);
    }

    public String getLabCa() {
        return getDataTable().get("labCa") != null ?
getDataTable().get("labCa").toString() : "";
    }

    public void setLabCa(String labCa) {
        getDataTable().put("labCa", labCa);
    }

    public String getLabCpk() {
        return getDataTable().get("labCpk") != null ?
getDataTable().get("labCpk").toString() : "";
    }

    public void setLabCpk(String labCpk) {
        getDataTable().put("labCpk", labCpk);
    }

    public String getLabCreatinine() {
        return getDataTable().get("labCreatinine") != null ?
getDataTable().get("labCreatinine").toString() : "";
    }

    public void setLabCreatinine(String labCreatinine) {
        getDataTable().put("labCreatinine", labCreatinine);
    }

    public String getLabDek() {
        return getDataTable().get("labDek") != null ?
getDataTable().get("labDek").toString() : "";
    }

    public void setLabDek(String labDek) {
        getDataTable().put("labDek", labDek);
    }

    public String getLabGgt() {
        return getDataTable().get("labGgt") != null ?
getDataTable().get("labGgt").toString() : "";
    }

    public void setLabGgt(String labGgt) {
        getDataTable().put("labGgt", labGgt);
    }

    public String getLabGlucose() {
        return getDataTable().get("labGlucose") != null ?
getDataTable().get("labGlucose").toString() : "";
    }

    public void setLabGlucose(String labGlucose) {
        getDataTable().put("labGlucose", labGlucose);
    }

    public String getLabHaemoglobin() {
```

```
        return getDataTable().get("labHaemoglobin") != null ?
getDataTable().get("labHaemoglobin").toString() : "";
    }

    public void setLabHaemoglobin(String labHaemoglobin) {
        getDataTable().put("labHaemoglobin", labHaemoglobin);
    }

    public String getLabHematocrit() {
        return getDataTable().get("labHematocrit") != null ?
getDataTable().get("labHematocrit").toString() : "";
    }

    public void setLabHematocrit(String labHematocrit) {
        getDataTable().put("labHematocrit", labHematocrit);
    }

    public String getLabHws() {
        return getDataTable().get("labHws") != null ?
getDataTable().get("labHws").toString() : "";
    }

    public void setLabHws(String labHws) {
        getDataTable().put("labHws", labHws);
    }

    public String getLabK() {
        return getDataTable().get("labK") != null ?
getDataTable().get("labK").toString() : "";
    }

    public void setLabK(String labK) {
        getDataTable().put("labK", labK);
    }

    public String getLabLdh() {
        return getDataTable().get("labLdh") != null ?
getDataTable().get("labLdh").toString() : "";
    }

    public void setLabLdh(String labLdh) {
        getDataTable().put("labLdh", labLdh);
    }

    public String getLabLemf() {
        return getDataTable().get("labLemf") != null ?
getDataTable().get("labLemf").toString() : "";
    }

    public void setLabLemf(String labLemf) {
        getDataTable().put("labLemf", labLemf);
    }

    public String getLabMon() {
        return getDataTable().get("labMon") != null ?
getDataTable().get("labMon").toString() : "";
    }

    public void setLabMon(String labMon) {
        getDataTable().put("labMon", labMon);
    }
```

```
      public String getLabNa() {
          return getDataTable().get("labNa") != null ?
getDataTable().get("labNa").toString() : "";
      }

      public void setLabNa(String labNa) {
          getDataTable().put("labNa", labNa);
      }

      public String getLabPlatelets() {
          return getDataTable().get("labPlatelets") != null ?
getDataTable().get("labPlatelets").toString() : "";
      }

      public void setLabPlatelets(String labPlatelets) {
          getDataTable().put("labPlatelets", labPlatelets);
      }

      public String getLabPol() {
          return getDataTable().get("labPol") != null ?
getDataTable().get("labPol").toString() : "";
      }

      public void setLabPol(String labPol) {
          getDataTable().put("labPol", labPol);
      }

      public String getLabSgot() {
          return getDataTable().get("labSgot") != null ?
getDataTable().get("labSgot").toString() : "";
      }

      public void setLabSgot(String labSgot) {
          getDataTable().put("labSgot", labSgot);
      }

      public String getLabSgpt() {
          return getDataTable().get("labSgpt") != null ?
getDataTable().get("labSgpt").toString() : "";
      }

      public void setLabSgpt(String labSgpt) {
          getDataTable().put("labSgpt", labSgpt);
      }

      public String getLabTke() {
          return getDataTable().get("labTke") != null ?
getDataTable().get("labTke").toString() : "";
      }

      public void setLabTke(String labTke) {
          getDataTable().put("labTke", labTke);
      }

      public String getLabUrea() {
          return getDataTable().get("labUrea") != null ?
getDataTable().get("labUrea").toString() : "";
      }

      public void setLabUrea(String labUrea) {
```

```
            getDataTable().put("labUrea", labUrea);
        }

        public String getLabWhite() {
            return getDataTable().get("labWhite") != null ?
getDataTable().get("labWhite").toString() : "";
        }

        public void setLabWhite(String labWhite) {
            getDataTable().put("labWhite", labWhite);
        }

    //    public void setAll(String amka, String surname, String name, String
bloodTypeRh, String address, String postalCode, String city, String phoneNumber,
String historyExamEvaluation, String diceaseProgress, String exitDiagnosis,
String treatmentIntervention, String exitDirectivesComments, String ecg, String
radiography, String otherExams, String nearestRelativeSurname, String
nearestRelativeName, String nearestRelativePhoneNumber, String
nearestRelativeRelationship) {
    //        getDataTable().put("amka", amka);
    //        getDataTable().put("surname", surname);
    //        getDataTable().put("name", name);
    //        getDataTable().put("bloodTypeRh", bloodTypeRh);
    //        getDataTable().put("address", address);
    //        getDataTable().put("postalCode", postalCode);
    //        getDataTable().put("city", city);
    //        getDataTable().put("phoneNumber", phoneNumber);
    //        getDataTable().put("nearestRelativeSurname",
nearestRelativeSurname);
    //        getDataTable().put("nearestRelativeName", nearestRelativeName);
    //        getDataTable().put("nearestRelativePhoneNumber",
nearestRelativePhoneNumber);
    //        getDataTable().put("nearestRelativeRelationship",
nearestRelativeRelationship);
    //        getDataTable().put("historyExamEvaluation", historyExamEvaluation);
    //        getDataTable().put("diceaseProgress", diceaseProgress);
    //        getDataTable().put("exitDiagnosis", exitDiagnosis);
    //        getDataTable().put("treatmentIntervention", treatmentIntervention);
    //        getDataTable().put("exitDirectivesComments",
exitDirectivesComments);
    //        getDataTable().put("ecg", ecg);
    //        getDataTable().put("radiography", radiography);
    //        getDataTable().put("otherExams", otherExams);
    //    }
    //
        public static final Hashtable dummyRecord() {
            Hashtable dummyH = new Hashtable();
            dummyH.put("amka", "12058100017");
            dummyH.put("surname", "Τεμπριώτης");
            dummyH.put("name", "Παναγιώτης");
            dummyH.put("bloodTypeRh", "O+");
            dummyH.put("address", "Μπουκουβάλα 44, Γκύζη");
            dummyH.put("postalCode", "11475");
            dummyH.put("city", "Αθήνα");
            dummyH.put("phoneNumber", "+302107771831");
            dummyH.put("nearestRelativeSurname", "Τεμπριώτης");
            dummyH.put("nearestRelativeName", "Νίκος");
            dummyH.put("nearestRelativePhoneNumber", "+35722331925");
            dummyH.put("nearestRelativeRelationship", "πατέρας");
            dummyH.put("historyExamEvaluation", "");
            dummyH.put("diceaseProgress", "");
```

```
          dummyH.put("exitDiagnosis", "");
          dummyH.put("treatmentIntervention", "");
          dummyH.put("exitDirectivesComments", "");
          dummyH.put("ecg", "");
          dummyH.put("radiography", "");
          dummyH.put("otherExams", "");
          return dummyH;
      }
  //    private Hashtable h = new Hashtable();
  //    public MedicalRecord(String amka, String surname, String name, String
bloodTypeRh, String address, String postalCode, String city, String phoneNumber,
String historyExamEvaluation, String diceaseProgress, String exitDiagnosis,
String treatmentIntervention, String exitDirectivesComments, String ecg, String
radiography, String otherExams, String nearestRelativeSurname, String
nearestRelativeName, String nearestRelativePhoneNumber, String
nearestRelativeRelationship) {
  //        h.put("amka", amka);
  //        h.put("surname", surname);
  //        h.put("name", name);
  //        h.put("bloodTypeRh", bloodTypeRh);
  //        h.put("address", address);
  //        h.put("postalCode", postalCode);
  //        h.put("city", city);
  //        h.put("phoneNumber", phoneNumber);
  //        h.put("nearestRelativeSurname", nearestRelativeSurname);
  //        h.put("nearestRelativeName", nearestRelativeName);
  //        h.put("nearestRelativePhoneNumber", nearestRelativePhoneNumber);
  //        h.put("nearestRelativeRelationship", nearestRelativeRelationship);
  //        h.put("historyExamEvaluation", historyExamEvaluation);
  //        h.put("diceaseProgress", diceaseProgress);
  //        h.put("exitDiagnosis", exitDiagnosis);
  //        h.put("treatmentIntervention", treatmentIntervention);
  //        h.put("exitDirectivesComments", exitDirectivesComments);
  //        h.put("ecg", ecg);
  //        h.put("radiography", radiography);
  //        h.put("otherExams", otherExams);
  //    }
  //
  //    public MedicalRecord() {
  //        this("", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
"", "", "", "");
  //    }
  //
  //    /**
  //     * Recursive constructor invocation.
  //     * @param rcS String[] containing the variables as read from the
RecordStore.
  //     */
  //    public MedicalRecord(String[] rcS) {
  //        this(rcS[0], rcS[1], rcS[2], rcS[3], rcS[4], rcS[5], rcS[6], rcS[7],
rcS[8], rcS[9],
  //                rcS[10], rcS[11], rcS[12], rcS[13], rcS[14], rcS[15],
rcS[16], rcS[17], rcS[18], rcS[19]);
  //    }
  }
```

```
/*
 * RMS.java
 */
package org.ntua.mobile.medicalrecord.storage;

import java.security.NoSuchAlgorithmException;
import javax.crypto.NoSuchPaddingException;
import org.ntua.mobile.medicalrecord.encryption.MyCipher;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import javax.microedition.rms.RecordEnumeration;
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import javax.microedition.rms.RecordStoreFullException;
import javax.microedition.rms.RecordStoreNotFoundException;
import javax.microedition.rms.RecordFilter;
import javax.microedition.rms.RecordComparator;

/**
 * @author Panayiotis Tembriotis <panayiotist@gmail.com>
 */
public class RMS implements RecordFilter, RecordComparator {

    private static final String MEDICAL_RECORD_RS = "MedicalRecords";
    public static final String I4_NO_LABS = "I4_NO_LABS";
    public static final String I4_LAB_ONLY = "I4_LABS_ONLY";
    public static final String EMERGENCY = "EMERGENCY";
    private String recordDataType;
    private MyCipher encryptor;
    private String password;

    public RMS(String password) {
        this.password = password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public int[] getRecordIds(String recordDataType) {
        int[] recordIds = null;
        this.recordDataType = recordDataType;
        try {
            RecordStore rs = openRecordStore();
            RecordEnumeration e = rs.enumerateRecords(this, null, true);
            recordIds = new int[e.numRecords()];
            for (int i = 0; i < recordIds.length; i++) {
                recordIds[i] = e.nextRecordId();
            }
        } catch (RecordStoreException recordStoreException) {
            System.out.println("recordids=" + recordIds);
            recordStoreException.printStackTrace();
            recordIds = null;
        }
        System.out.println("recordIds length=" + recordIds.length);
        return recordIds;
    }
```

```
    public byte[] getRecord(int recordId) {
        byte[] data = null;
        try {
            RecordStore rs = openRecordStore();
            data = callDecrypt(rs.getRecord(recordId));
        } catch (Exception exception) {
            exception.printStackTrace();
            data = null;
        }
        return data;
    }

    public int addRecord(String RecordDataType, byte[] recordDataBs) throws
IOException, RecordStoreException, Exception {
        this.recordDataType = RecordDataType;
        int recordId;
        RecordStore rs = openRecordStore();
        recordDataBs = callEncrypt(recordDataBs);
        recordId = rs.addRecord(recordDataBs, 0, recordDataBs.length);
        rs.closeRecordStore();
        return recordId;
    }

    public void updateRecord(int recordId, byte[] recordDataBs) throws
IOException, RecordStoreException, Exception {
        System.out.println("Update recordId=" + recordId);
        RecordStore rs = openRecordStore();
        recordDataBs = callEncrypt(recordDataBs);
        rs.setRecord(recordId, recordDataBs, 0, recordDataBs.length);
        rs.closeRecordStore();
    }

    public void deleteRecord(String RecordDataType) {
        this.recordDataType = RecordDataType;
        try {
            RecordStore rs = openRecordStore();
            RecordEnumeration e = rs.enumerateRecords(this, null, true);
            while (e.hasNextElement()) {
                try {
                    rs.deleteRecord(e.nextRecordId());
                } catch (RecordStoreException ex) {
                }
            }
            e.destroy();
            rs.closeRecordStore();
        } catch (RecordStoreException recordStoreException) {
            recordStoreException.printStackTrace();
        }
    }

    public int getNumRecords(String RecordDataType) {
        this.recordDataType = RecordDataType;
        int records = -1;
        try {
            RecordStore rs = openRecordStore();
            System.out.println("rs.numrecords=" + rs.getNumRecords());
            RecordEnumeration e = rs.enumerateRecords(this, null, true);
            records = e.numRecords();
            System.out.println("records=" + records);
            e.destroy();
            rs.closeRecordStore();
```

```
        } catch (RecordStoreException recordStoreException) {
            recordStoreException.printStackTrace();
            records = -1;
        }
        return records;
    }

    public void closeRecordStores() {
        String[] recordNames = RecordStore.listRecordStores();
        RecordStore rs;
        if (recordNames != null && recordNames.length > 0) {
            for (int i = 0; i < recordNames.length; i++) {
                try {
                    rs = RecordStore.openRecordStore(recordNames[i], false,
RecordStore.AUTHMODE_PRIVATE, true);
                } catch (RecordStoreException ex) {
                    break;
                }
                while (true) {
                    try {
                        rs.closeRecordStore();
                    } catch (RecordStoreException ex) {
                        break;
                    }
                }
            }
        }
    }

    public void deleteRecordStore() {
        closeRecordStores();
        try {
            RecordStore.deleteRecordStore(RMS.MEDICAL_RECORD_RS);
        } catch (RecordStoreException ex) {
            ex.printStackTrace();
        }
    }

    private byte[] callEncrypt(byte[] bs) throws RecordStoreException,
IOException, Exception {
        bs = getEncryptor().encrypt(bs);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        DataOutputStream dos = new DataOutputStream(baos);
        dos.writeUTF(recordDataType);
        dos.write(bs);
        dos.flush();
        return baos.toByteArray();
    }

    private byte[] callDecrypt(byte[] bs) throws RecordStoreException,
IOException, Exception {
        ByteArrayInputStream bais = new ByteArrayInputStream(bs);
        DataInputStream dis = new DataInputStream(bais);
        recordDataType = DataInputStream.readUTF(dis);
        bs = new byte[bais.available()];
        bais.read(bs);
        bais.close();
        return getEncryptor().decrypt(bs);
 //        return bs;
    }
```

```
      private MyCipher getEncryptor() throws NoSuchAlgorithmException,
NoSuchPaddingException, IOException {
          if (encryptor == null) {
              encryptor = new MyCipher(password);
          }
          return encryptor;
      }

      private RecordStore openRecordStore() throws RecordStoreNotFoundException,
RecordStoreFullException, RecordStoreException {
          return RecordStore.openRecordStore(RMS.MEDICAL_RECORD_RS, true,
RecordStore.AUTHMODE_PRIVATE, true);
      }

      public boolean matches(byte[] candidate) {
          System.out.println("recordDataType=" + recordDataType);
          if (recordDataType == null) {
              return false;
          }
          ByteArrayInputStream bais = new ByteArrayInputStream(candidate);
          DataInputStream dis = new DataInputStream(bais);
          String recordDataTypeTest = null;
          try {
              recordDataTypeTest = dis.readUTF();
          } catch (Exception ex) {
              return false;
          }
          return (this.recordDataType.equals(recordDataTypeTest));
      }

      public int compare(byte[] rec1, byte[] rec2) {
          try {
  //            rec1 = decrypt(rec1);
  //            rec2 = decrypt(rec2);
              ByteArrayInputStream bais1 = new ByteArrayInputStream(rec1);
              DataInputStream dis1 = new DataInputStream(bais1);
              ByteArrayInputStream bais2 = new ByteArrayInputStream(rec2);
              DataInputStream dis2 = new DataInputStream(bais2);
              dis1.readUTF();
              dis2.readUTF();
              long date1 = dis1.readLong();
              long date2 = dis2.readLong();
              if (date1 < date2) {
                  return RecordComparator.PRECEDES;
              } else if (date1 > date2) {
                  return RecordComparator.FOLLOWS;
              } else {
                  return RecordComparator.EQUIVALENT;
              }
          } catch (Exception ex) {
              return RecordComparator.EQUIVALENT;
          }
      }
  }
```