ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογές κινητών συσκευών για τη διαχείριση έξυπνων καρτών με χρήση τεχνολογιών JavaCard και J2ME

Καπετανάκης Ηλίας
M3020033

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2004

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)
στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εφαρμογές κινητών συσκευών για τη διαχείριση έξυπνων καρτών με χρήση τεχνολογιών JavaCard και J2ME

Καπετανάκης Ηλίας
M3020033

Επιβλέπων Καθηγητής : Θεόδωρος Αποστολόπουλος

Εξωτερικός Κριτής : Ιωάννης Μήλης

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2004

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους ανθρώπους που με βοήθησαν να εκπονήσω αυτήν την εργασία . Ευχαριστώ θερμά :

- Τον Καθηγητή του Οικονομικού Πανεπιστημίου Αθηνών, κύριο Αποστολόπουλο Θεόδωρο για την καθοδήγησή και τις χρήσιμες συμβουλές του κατά την εκπόνηση της εργασίας.

- Τον υποψήφιο διδάκτορα Οικονόμου Γιώργο , για τις συμβουλές και τον χρόνο που διέθεσε για μένα.

- Τους δικούς μου ανθρώπους για την ηθική συμπαράσταση που μου προσέφεραν.

Περίληψη

Ζούμε σε μια εποχή που η ανάγκη για ικανοποίηση καθημερινών αναγκών και για πραγματοποίηση συναλλαγών οπουδήποτε και οποτεδήποτε ολοένα και αυξάνει. Αυτή η ανάγκη, σε συνδυασμό με την απαίτηση για ασφαλή περάτωση τέτοιων διαδικασιών, αναδεικνύουν την ανάγκη για ανάπτυξη ευέλικτων και ασφαλών εφαρμογών σε συσκευές που μας συνοδεύουν καθημερινά. Η απάντηση σε αυτή την ανάγκη είναι οι έξυπνες κάρτες και οι κινητές συσκευές.

Οι δύο τεχνολογίες που έχουν αναπτυχθεί από την Sun για τις έξυπνες κάρτες και τις κινητές συσκευές είναι η Java Card και η J2ME (Java 2 Platform Micro Edition) αντίστοιχα. Η J2ME έχει τα πλεονεκτήματα της Java τεχνολογίας, προσαρμοσμένα στις embedded συσκευές και περιλαμβάνει ένα ευέλικτο user interface, ένα εύρωστο μοντέλο ασφάλειας, ένα ευρύ φάσμα από ενσωματωμένα δικτυακά πρωτόκολλα και υποστήριξη για δικτυακές ή μη εφαρμογές. Η τεχνολογία Java Card διατηρεί και αυτή πολλά από τα οφέλη της γλώσσας προγραμματισμού Java - παραγωγικότητα, ασφάλεια, ευρωστία, εργαλεία, και φορητότητα - επιτρέποντας την χρήση της τεχνολογία της Java σε έξυπνες κάρτες.

Η J2ME, παρέχει διαφορετικά configurations και profiles, ανάλογα με τα χαρακτηριστικά της συσκευής που πρόκειται να χρησιμοποιηθεί. Τα configurations αποτελούνται από μια εικονική μηχανή (virtual machine) και ένα ελάχιστο σύνολο βιβλιοθηκών κλάσεων και υπάρχουν δύο J2ME configurations: η Connected Limited Device Configuration (CLDC), και η Connected Device Configuration (CDC). Τα configurations συνδυάζονται με τα profiles, τα οποία καθορίζουν περαιτέρω τον κύκλο ζωής των εφαρμογών, τη διεπαφή του χρήστη και την πρόσβαση σε συγκεκριμένες ιδιότητες των συσκευών. Τα profiles είναι : το Mobile Information Device Profile (MIDP), το Foundation Profile (FP), το Personal Profile (PP) και το Personal Basis Profile (PBP). Συνδυασμένο με το CLDC, το MIDP παρέχει ένα πλήρες Java runtime περιβάλλον το οποίο ισχυροποιεί τις ικανότητες των φορητών συσκευών και ελαχιστοποιεί την κατανάλωση μνήμης και ισχύος.

Στην Java Card η εικονική μηχανή (VM), ο ορισμός της γλώσσας, και τα κύρια πακέτα έχουν γίνει συμπαγή και σύντομα για να φέρουν την τεχνολογία της Java στο περιορισμένο σε πόρους περιβάλλον των έξυπνων καρτών. Η Java Card Virtual Machine υλοποιείται ως δύο ανεξάρτητα κομμάτια. Το πρώτο κομμάτι της JCVM εκτελείτε off-card σε κάποιο PC και κάνει όλη τη δουλειά που απαιτείται για να φορτωθούν οι κλάσεις. Το δεύτερο on-card κομμάτι της JCVM , περιλαμβάνει τον bytecode interpreter. Αυτό σημαίνει ότι επιπλέον προεπεξεργασία χρειάζεται πριν φορτωθεί το applet στην κάρτα. Μέσα στην κάρτα , το Java Card Runtime Environment (JCRE) περιλαμβάνει την on-card JCVM και τις κλάσεις στο JavaCard framework.

Μια ολοκληρωμένη εφαρμογή Java Card αποτελείται από μία back-end εφαρμογή και συστήματα, μία host εφαρμογή (εκτός κάρτας), μία συσκευή διεπαφής (interface device – card reader) και το applet που βρίσκεται πάνω στην κάρτα, τα πιστοποιητικά του χρήστη και το συνοδευτικό λογισμικό. Το μοντέλο διαβίβασης μηνυμάτων είναι η βάση για όλες τις επικοινωνίες με Java Cards και στο κέντρο του μοντέλου αυτού είναι το Application Protocol Data Unit (APDU), ένα λογικό πακέτο δεδομένων το οποίο ανταλλάσσεται μεταξύ κάποιας τερματικής εφαρμογής και του Java Card Framework. Το Java Card Framework λαμβάνει και προωθεί στο κατάλληλο applet κάθε εισερχόμενη εντολή APDU που αποστέλλεται από την τερματική εφαρμογή. Το applet επεξεργάζεται την APDU εντολή και επιστρέφει μία APDU απάντηση. Η κάρτα λοιπόν παίζει έναν παθητικό ρόλο περιμένοντας να εξυπηρετήσει τις APDU εντολές που καταφθάνουν σε αυτή.

Οι τρόποι επικοινωνίας μιας κάρτας και μιας τερματικής εφαρμογής ποικίλουν αν και όλοι χρησιμοποιούν το μοντέλο μηνυμάτων βασισμένο στις APDU. Τέτοιοι τρόποι επικοινωνίας είναι το Java Card RMI (JCRMI), το Security And Trust Service API (SATSA) και το OpenCard Framework . Ειδικότερα το OpenCard Framework είναι ένα στάνταρ framework που ανακοινώθηκε από ένα βιομηχανικό consortium, μεγάλων εταιριών που δραστηριοποιούνται στις έξυπνες κάρτες και παρέχει δια-λειτουργικότητα μεταξύ πολλών hardware and software πλατφόρμων. Το OpenCard Framework είναι ένα ανοιχτό στάνταρ που προσφέρει μία αρχιτεκτονική και ένα σύνολο από APIs τα οποία διευκολύνουν αυτούς που αναπτύσσουν εφαρμογές και τους παρόχους υπηρεσιών να χτίζουν και να διανέμουν λύσεις βασισμένες σε

έξυπνες κάρτες και πιο συγκεκριμένα τις τερματικές εφαρμογές που είναι γραμμένες σε Java, σε οποιοδήποτε OpenCard συμβατό περιβάλλον.

Στόχος της εργασίας αυτής είναι η δημιουργία μιας εφαρμογής διαχείρισης μιας έξυπνης κάρτας από μία κινητή συσκευή συνδυάζοντας όλες τις σχετικές τεχνολογίες. Η έξυπνη κάρτα θα υλοποιήσει τις λειτουργίες ενός ηλεκτρονικού πορτοφολιού και βάση αυτού θα υλοποιηθούν και οι υπόλοιπες εφαρμογές. Πρώτο βήμα είναι η κατασκευή της Java Card που αντιστοιχεί σε αυτή την έξυπνη κάρτα, δεύτερο η δημιουργία τερματικής εφαρμογής σε PC που διαχειρίζεται την κάρτα επικοινωνώντας με τη βοήθεια του Open Card Framework και τρίτο και τελευταίο η χρησιμοποίηση των συστατικών των δύο προηγούμενων βημάτων και η υλοποίηση και των υπολοίπων απαραίτητων συστατικών για την κατασκευή εφαρμογής για κινητή συσκευή που θα διαχειρίζεται την κάρτα.

Η Java Card που υλοποιείται είναι ένα ηλεκτρονικό πορτοφόλι. Συνεπώς θα πρέπει να υπάρχει επαλήθευση ενός PIN για την αυθεντικοποίηση του χρήστη, ενώ για να γίνει πιο λειτουργική η κάρτα, τοποθετείται και μία λειτουργία ανανέωσης του PIN, δηλαδή αντικατάστασης του παλιού με ένα νέο PIN επιλογής του χρήστη. Οι υπόλοιπες λειτουργίες της κάρτας είναι η πίστωση, η χρέωση και η ερώτηση υπολοίπου.

Το πρώτο σενάριο διαχείρισης της κάρτας από τερματική εφαρμογή σε PC περιλαμβάνει καταρχάς τα τμήματα του συστήματος που είναι αντιληπτά από το χρήστη , την Java Card και την τερματική εφαρμογή. Για να επικοινωνήσουν όμως αυτά τα δύο συστατικά με ένα τρόπο ώστε να είναι ανεξάρτητος από τον αναγνώστη καρτών ή τον τύπο της κάρτας , ενώ θα μπορεί να χρησιμοποιηθεί και από άλλου τύπου τερματικές εφαρμογές , πρέπει να μεσολαβήσει το Open Card Framework.

Στο δεύτερο σενάριο που αποτελεί και τον στόχο της εργασίας, χρησιμοποιούνται τα συστατικά της Java Card και του Open Card Framework που έχουν ήδη υλοποιηθεί. Τα δύο επιπλέον συστατικά που πρέπει να υλοποιηθούν είναι μία εφαρμογή για την κινητή συσκευή γραμμένη σε J2ME και μία εφαρμογή java η οποία θα αναλάβει να επικοινωνεί με το Open Card Framework και να προωθεί τις αιτήσεις που θα καταφθάνουν από την εφαρμογή της κινητής συσκευής. Η επικοινωνία μεταξύ της

κινητής συσκευής και του PC θα γίνει μέσω sockets όπου η J2ME εφαρμογή θα παίζει τον ρόλο του πελάτη (client) ενώ η εφαρμογή το PC τον ρόλο του εξυπηρετητή (server). Συνεπώς το Open Card Framework και ο server αποτελούν μία Java Card – Mobile Device Gateway. Η εφαρμογή J2ME μέσω GPRS ή με κάποιον άλλο τρόπο που θα τις παρέχει σύνδεση στο internet αποστέλλει μηνύματα στο Gateway το οποίο και αναλαμβάνει την επικοινωνία με την Java Card. Η αντίστροφη διαδικασία ακολουθείται για την μεταφορά των απαντήσεων της κάρτας στην κινητή συσκευή.

Executive Summary

In our time, we have an ongoing need to satisfy every day occasions and make transactions at anytime. This requirement, in combination with the demand for secure completion of such procedures, emerge the need for development of secure and flexible applications for devices that we use every day. The answer to this need is, smart cards and mobile devices.

The two technologies that Sun has developed , one for smart cards and one for mobile devices are Java Card and J2ME (Java 2 Platform Micro Edition). The J2ME platform delivers the power and benefits of Java technology tailored for consumer and embedded devices — including a flexible user interface, robust security model, broad range of built-in network protocols, and support for networked and disconnected applications. Java Card technology, it keeps too many of the benefits of Java language as productivity, security, robustness, tools and portability, allowing the use of Java technology in smart cards.

The J2ME, provides different Configurations and Profiles, according to the characteristics of the device that will be used. Configurations are composed of a virtual machine and a minimal set of class libraries. Currently, there are two J2ME configurations: the Connected Limited Device Configuration(CLDC), and the Connected Device Configuration (CDC). In order to provide a complete runtime environment targeted at specific device categories,configurations must be combined with profiles, that further define the application life cycle model, the user interface, and access to device specific properties. There are four J2ME Profiles : Mobile Information Device Profile (MIDP), Foundation Profile (FP), Personal Profile (PP) και το Personal Basis Profile (PBP). Combined with CLDC, MIDP provides a complete Java runtime environment that leverages the capabilities of handheld devices and minimizes both memory and power consumption.

In Java Card, Virtual Machine, language definition and the main packages are compact and short to bring Java technology in the limited to sources environment of smart cards. The JCVM is implemented as two separated pieces. The first piece of the

JCVM executes off-card on a PC or workstation and does all the work required for loading classes and resolving references. The second, on-card part of the JCVM, includes the bytecode interpreter. This means that additional preprocessing is needed before the applet is loaded onto the card. Inside the card, the Java Card Runtime Environment (JCRE) comprises the on-card JCVM and the classes in the JavaCard framework.

A complete Java Card application consists of a back-end application and systems, a host (off-card) application, an interface device (card reader), and the on-card applet, user credentials, and supporting software. All these elements together compose a secure end-to-end application. The message-passing model is the basis for all Java Card communications. At its center is the Application Protocol Data Unit (APDU), a logical data packet that's exchanged between the card reader and the Java Card Framework. The Java Card Framework receives and forwards to the appropriate applet any incoming command APDU sent by the card reader. The applet processes the command APDU, and returns a response APDU. So the card plays a passive role waiting to serve the APDU commands that arrive.

There are many models for communication between a host application and a Java Card applet but all of them use the APDU based message model. Such models are Java Card RMI (JCRMI), Security And Trust Service API (SATSA) and Open Card Framework. Especially, Open Card Framework is a standard framework announced by an Industry consortium that provides for inter-operable smart cards solutions across many hardware and software platforms. The OpenCard Framework is an open standard providing an architecture and a set of APIs that enable application developers and service providers to build and deploy smart card aware solutions in any OpenCard-compliant environment.

The main goal of this thesis is the implementation of a mobile device application that it will manage a smart card, combining all the relative technologies. The smart card will have all the functions of an electronic wallet, and according to these functions we will implement all the other applications. First step is the implementation of a Java Card with the characteristics for the smart card we described, second is the implementation of a PC terminal application that manages the card communicating

via Open Card Framework and third is the use of the components we previously created and the implementation of the rest necessary components to create the mobile device application that manages the card.

As we mentioned before, the Java Card is an electronic wallet. So it has to verify a PIN for user authentication and for to increase the functionality of the card it will have a PIN update function, to replace the old PIN with the one the user selects. The rest functions of the card are credit, debit and get balance.

The first scenario that includes card management from a PC terminal application, it contains two user–perceived components, the Java Card and the terminal application. The communication of these two components in a card reader or card type independent way will involve Open Card Framework. The most useful feature is that Open Card Framework can be used from other terminal application types and that's why we will use it in the second scenario.

In the second scenario, we use the Java Card and the Open Card Framework we have already implemented. The two additional components that must be implemented are a J2ME application for the mobile device and a Java application which will communicate with Open Card Framework to forward the J2ME application requests. The communication between the mobile device and PC will be via a socket connection where J2ME application is the client and PC application is server. So, Open Card Framework and server are a Java Card-Mobile Device Gateway. The J2ME application via GPRS or with some other way, it sends messages to the Gateway which take care of the communication with Java Card. The reverse process is followed for transferring the responses of the card to mobile device.

Περιεχόμενα

Εισαγωγή

Ζούμε σε μια εποχή που η ανάγκη για ικανοποίηση καθημερινών αναγκών και για πραγματοποίηση συναλλαγών οπουδήποτε και οποτεδήποτε ολοένα και αυξάνει. Αυτή η ανάγκη, σε συνδυασμό με την απαίτηση για ασφαλή περάτωση τέτοιων διαδικασιών, αναδεικνύουν την ανάγκη για ανάπτυξη ευέλικτων και ασφαλών εφαρμογών σε συσκευές που μας συνοδεύουν καθημερινά. Η απάντηση σε αυτή την ανάγκη είναι οι έξυπνες κάρτες και οι κινητές συσκευές.

Στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μιας εφαρμογής διαχείρισης μιας έξυπνης κάρτας από μία κινητή συσκευή. Έχουν επιλεγεί να χρησιμοποιηθούν τεχνολογίες της οικογένειας Java, για τον προγραμματισμό των επιμέρους συστημάτων και πιο συγκεκριμένα η τεχνολογία J2SE για τον προγραμματισμό των desktop εφαρμογών, η J2ME για τον προγραμματισμό των εφαρμογών σε κινητές συσκευές και τέλος η Java Card για την δημιουργία της εφαρμογής της έξυπνης κάρτας.

Στο πρώτο μέρος της εργασίας επιχειρείται μία εισαγωγή στις επιμέρους τεχνολογίες και ειδικότερα στην Java Card . Μελετάται η αρχιτεκτονική της τεχνολογίας , καθώς και το μοντέλο επικοινωνίας μιας έξυπνης κάρτας υλοποιημένης με την τεχνολογία Java Card με κάποιας εξωτερικής συσκευής. Έτσι φτάνουμε στο σημείο να μπορούμε να κατανοήσουμε τον τρόπο λειτουργίας της και να πάρουμε τις κατάλληλες σχεδιαστικές αποφάσεις για την εφαρμογή μας.

Στο δεύτερο μέρος, ασχολούμαστε με την υλοποίηση της εφαρμογής. Αρχικά αναλύουμε τα βήματα για την κατασκευή μιας έξυπνης κάρτας Java Card η οποία να πραγματοποιεί τις λειτουργίες ενός ηλεκτρονικού πορτοφολιού. Έπειτα παρουσιάζουμε δύο σενάρια χρήσης. Το πρώτο αφορά την διαχείριση της κάρτας από μία εφαρμογή που βρίσκεται σε ένα PC ενώ το δεύτερο που είναι και το ζητούμενο αφορά την διαχείριση της κάρτας από μία κινητή συσκευή.

Πιο συγκεκριμένα στο 1° κεφάλαιο κάνουμε μία γενική επισκόπηση των τεχνολογιών J2ME και Java Card, στο 2° κεφάλαιο αναλύουμε με λεπτομέρεια την τεχνολογία

Java Card, στο 3° κεφάλαιο, εξετάζουμε την αρχιτεκτονική των Java Card εφαρμογών, στο 4° κεφάλαιο εξετάζουμε τρόπους σύνδεσης τερματικών εφαρμογών με Java Card Applets, στο 5° κεφάλαιο σχεδιάζουμε και υλοποιούμε ένα ηλεκτρονικό πορτοφόλι με τεχνολογία Java Card, στο 6° κεφάλαιο σχεδιάζουμε και υλοποιούμε την τερματική εφαρμογή για την διαχείριση του ηλεκτρονικού πορτοφολιού, στο 7° κεφάλαιο σχεδιάζουμε και υλοποιούμε εφαρμογή για κινητή συσκευή , για την διαχείριση του ηλεκτρονικού πορτοφολιού και τέλος στο 8° κεφάλαιο παραθέτουμε τα συμπεράσματά μας.

ΚΕΦΑΛΑΙΟ 1

ΤΕΧΝΟΛΟΓΙΕΣ JAVA

Κεφάλαιο 1

1. ΤΕΧΝΟΛΟΓΙΕΣ JAVA

Η γλώσσα Java είναι πλέον μία πολύ γνωστή αντικειμενοστρεφής γλώσσα προγραμματισμού που δημιουργήθηκε από την Sun Microsystems και χρησιμοποιείται ευρέως. Τα δύο κυριότερα χαρακτηριστικά της είναι η αντικειμενοστρέφεια και η μεταφερσιμότητα. Η Java έχει όλα εκείνα τα πλεονεκτήματα που χαρακτηρίζουν της αντικειμενοστραφείς γλώσσες ενώ η μεταφερσιμότητα είναι η ιδιότητα εκείνη που την κάνει να ξεχωρίζει από τις υπόλοιπες αντικειμενοστραφείς γλώσσες αφού είναι ανεξάρτητη πλατφόρμας και μπορεί να τρέξει σε μηχανήματα με διαφορετικά χαρακτηριστικά και λειτουργικά συστήματα.

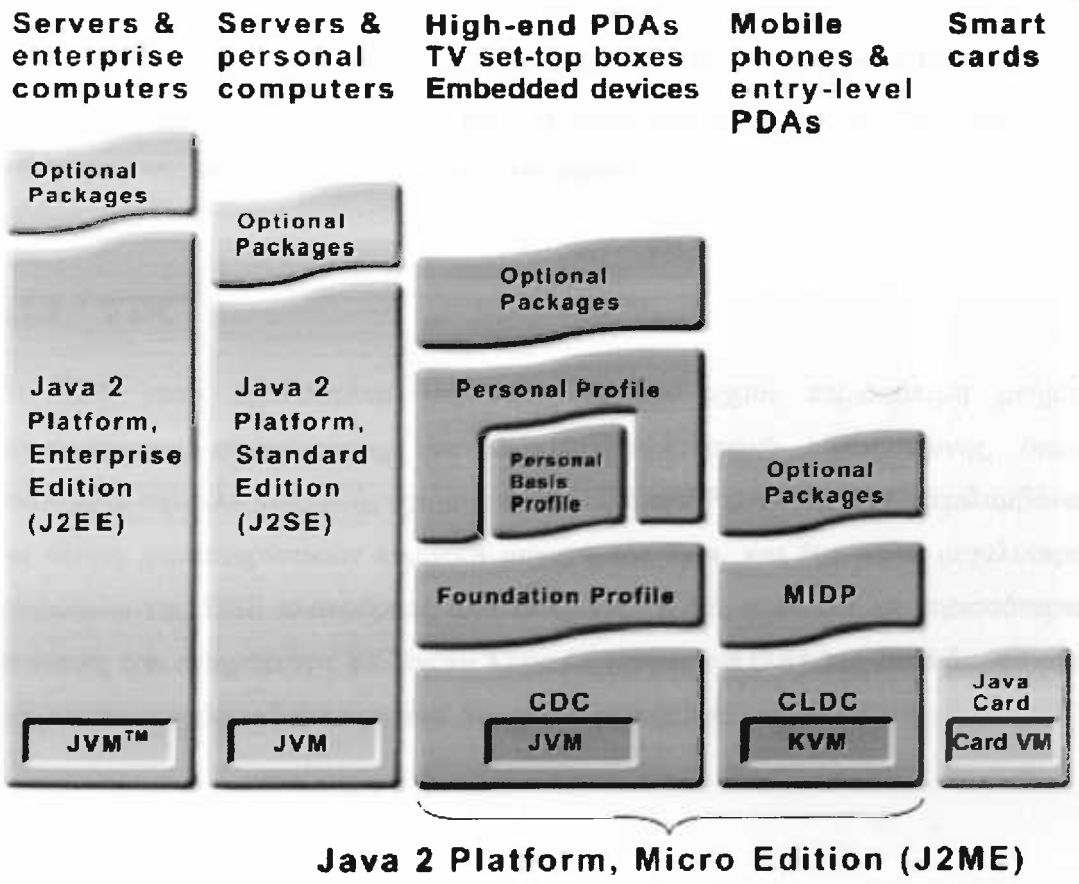
Ένα πλήθος τεχνολογιών που αναπτύσσονται σε διάφορους τομείς και βασίζονται στην Java , αυξάνουν την αξία της και παρέχουν ιδανικές λύσεις εάν συνδυαστούν, αφού έχοντας κάποιος την τεχνογνωσία της Java μπορεί εύκολα να προσαρμοσθεί στις νέες αυτές τεχνολογίες. Στη συνέχεια θα δούμε τις τεχνολογίες J2ME και JavaCard οι οποίες δίνουν απαντήσεις στις προκλήσεις της σημερινής εποχής.

1.1 Java 2 Platform , Micro Edition

Η Java 2 Platform, Micro Edition (J2ME), είναι η java πλατφόρμα με έμφαση σε embedded συσκευές όπως κινητά τηλέφωνα και PDAs. Όπως οι συμπληρωματικές

τεχνολογίες J2EE (Java 2 Platform , Enterprise Edition), J2SE (Java 2 Platform , Standard Edition) και Java Card λεπτομέρειες για την οποία θα δούμε στη συνέχεια, έτσι και η J2ME είναι ένα σύνολο από Java APIs .

Η J2ME έχει τα πλεονεκτήματα της Java τεχνολογίας, προσαρμοσμένα για τις embedded συσκευές και περιλαμβάνοντας ένα ευέλικτο user interface, ένα εύρωστο μοντέλο ασφάλειας, ένα ευρύ φάσμα από ενσωματωμένα δικτυακά πρωτόκολλα και υποστήριξη για δικτυακές ή μη εφαρμογές. Με την J2ME, εφαρμογές γράφονται μία φορά για μία πληθώρα συσκευών, μπορούν να γίνουν download δυναμικά και να αυξήσουν τις φυσικές ικανότητες των συσκευών [1].



Σχήμα 1.1.1 – Η διάρθρωση των Java τεχνολογιών

1.1.1 Configurations

Τα configurations αποτελούνται από μια εικονική μηχανή (virtual machine) και ένα ελάχιστο σύνολο βιβλιοθηκών κλάσεων. Παρέχουν τη βασική λειτουργία για μια ιδιαίτερη σειρά συσκευών που έχουν παρόμοια χαρακτηριστικά, όπως η συνδεσιμότητα δικτύων και χώρος μνήμης. Υπάρχουν δύο J2ME configurations: η Connected Limited Device Configuration (CLDC), και η Connected Device Configuration (CDC).

1.1.1.1 CLDC

Το CLDC είναι το μικρότερο των δύο configurations, σχεδιασμένο όπως καταμαρτυρεί και το όνομά του για συσκευές με ασυνεχείς δικτυακές συνδέσεις, αργούς επεξεργαστές και περιορισμένη μνήμη, συσκευές όπως τα κινητά τηλέφωνα, και τα PDAs. Αυτές οι συσκευές έχουν τυπικά είτε 16bit είτε 32bit CPUs, και ένα ελάχιστο από 128 KB σε 512 KB χώρο μνήμης διαθέσιμο για την υλοποίηση της πλατφόρμα της Java και για τις σχετικές εφαρμογές.

1.1.1.2 CDC

Το CDC είναι σχεδιασμένο για συσκευές που έχουν περισσότερη μνήμη, γρηγορότερους επεξεργαστές, και μεγαλύτερο δικτυακό εύρος ζώνης, όπως συστήματα τηλεπληροφορικής οχημάτων, και high-end PDAs. Το CDC περιλαμβάνει μια πλήρη χαρακτηριστικών εικονική μηχανή της Java, και ένα πολύ μεγαλύτερο υποσύνολο της J2SE πλατφόρμας από το CLDC. Κατά συνέπεια, οι περισσότερες συσκευές που στοχεύει το CDC έχουν 32 bit CPUs και ένα ελάχιστο 2MB διαθέσιμης μνήμης για την πλατφόρμα της Java και των σχετικές εφαρμογών.

1.1.2 Profiles

Προκειμένου να παρασχεθεί ένα πλήρες runtime περιβάλλον που στοχεύει σε συγκεκριμένες κατηγορίες συσκευών, τα configurations πρέπει να συνδυαστούν με ένα σύνολο επιπέδου APIs υψηλότερου επιπέδου, τα profiles, τα οποία καθορίζουν

περαιτέρω τον κύκλο ζωής των εφαρμογών, τη διεπαφή του χρήστη και την πρόσβαση σε συγκεκριμένες ιδιότητες των συσκευών.

1.1.2.1 MIDP - Mobile Information Device Profile

Το Mobile Information Device Profile (MIDP) σχεδιάστηκε για κινητά τηλέφωνα και entry-level PDAs. Προσφέρει τη βασική λειτουργικότητα των εφαρμογών που απαιτείται από τις κινητές εφαρμογές, συμπεριλαμβανομένου της διεπαφής του χρήστη, της συνδεσιμότητας δικτύων, της τοπικής αποθήκευσης δεδομένων, και της διαχείρισης της εφαρμογής. Συνδυασμένο με το CLDC, το MIDP παρέχει ένα πλήρες Java περιβάλλον runtime το οποίο ισχυροποιεί τις ικανότητες των φορητών συσκευών και ελαχιστοποιεί και την κατανάλωση μνήμης και ισχύος.

1.1.2.2 FP - Foundation Profile

Τα CDC profiles είναι διαστρωματομένα ώστε τα profiles να μπορούν να προστεθούν όπως απαιτούνται για να παρέχουν τη λειτουργικότητα της εφαρμογής για διαφορετικούς τύπους συσκευών. Το Foundation Profile (FP) είναι το χαμηλότερου επιπέδου profile για το CDC. Παρέχει μια ανοιχτή σε δίκτυα υλοποίηση του CDC που μπορεί να χρησιμοποιηθεί για τις βαθιά ενσωματωμένες εφαρμογές χωρίς διεπαφή χρήστη. Μπορεί επίσης να συνδυαστεί με το Personal Basis Profile και το Personal Profile , profiles που θα δούμε στη συνέχεια, για συσκευές που απαιτούν γραφική διεπαφή χρήστη (GUI).

1.1.2.3 PP – Personal Profile

Το Personal Profile (PP) είναι το CDC profile που στοχεύει σε συσκευές οι οποίες απαιτούν υποστήριξη πλήρους GUI ή applet, όπως high-end PDAs, communicators, και κονσόλες παιχνιδιών. Περιλαμβάνει το πλήρες Java Abstract Window Toolkit (AWT) και προσφέρει εύκολη προσπέλαση στα web applets που είναι σχεδιασμένα για τη χρήση σε ένα desktop περιβάλλον.

1.1.2.4 PBP – Personal Basis Profile

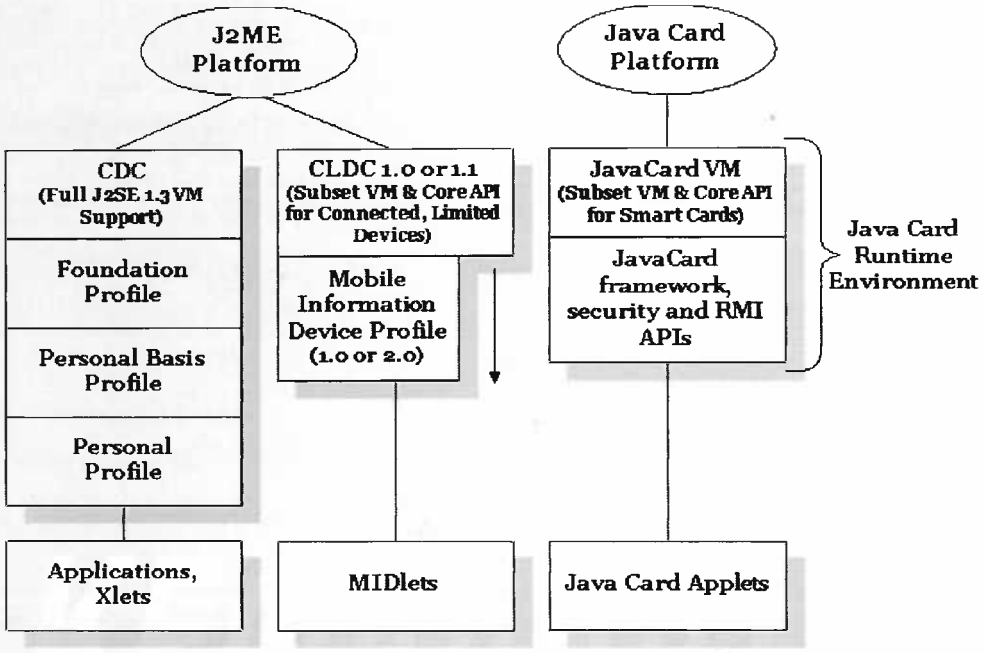
Το Personal Basis Profile (PBP), ένα υποσύνολο του PP, παρέχει ένα περιβάλλον εφαρμογών για συσκευές συνδεδεμένων σε δίκτυο οι οποίες υποστηρίζουν ένα βασικό επίπεδο γραφικής παρουσίασης ή απαιτούν τη χρήση εξειδικευμένων γραφικών toolkit για τις συγκεκριμένες εφαρμογές. Οι συσκευές περιλαμβάνουν τα συστήματα τηλεπληροφορικής οχημάτων, και τα infokiosks. Αμφότερα τα PP και PBP βρίσκονται πάνω από το CDC και το FP.

1.2 Java Card

Η τεχνολογία Java Card διατηρεί πολλά από τα οφέλη της γλώσσας προγραμματισμού Java - παραγωγικότητα, ασφάλεια, ευρωστία, εργαλεία, και φορητότητα - επιτρέποντας την χρήση της τεχνολογία της Java σε έξυπνες κάρτες. Η εικονική μηχανή (VM), ο ορισμός της γλώσσας, και τα κύρια πακέτα έχουν γίνει συμπαγή και σύντομα για να φέρουν την τεχνολογία της Java στο περιορισμένο σε πόρους περιβάλλον των έξυπνων καρτών.

Η Java Card Virtual Machine υλοποιείται ως δύο ανεξάρτητα κομμάτια [2]. Το πρώτο κομμάτι της JCVM εκτελείτε off-card σε κάποιο PC και κάνει όλη τη δουλειά που απαιτείται για να φορτωθούν οι κλάσεις. Το δεύτερο on-card κομμάτι της JCVM , περιλαμβάνει τον bytecode interpreter. Αυτό σημαίνει ότι επιπλέον προεπεξεργασία χρειάζεται πριν φορτωθεί το applet στην κάρτα. Μέσα στην κάρτα , το Java Card Runtime Environment (JCRE) περιλαμβάνει την on-card JCVM και τις κλάσεις στο JavaCard framework.

Αντιπαραθέτοντας τις τεχνολογίες J2ME και Java Card οι οποίες και οι δύο αποτελούν ειδικά συμπυκνωμένες εκδόσεις της JAVA βλέπουμε πως και τα δύο configurations CDC και CLDC καθώς και τα συνδεδεμένα προφίλ τους είναι μέρη από την πλατφόρμα J2ME, ενώ η πλατφόρμα Java Card είναι μία ξεχωριστεί πλατφόρμα που δημιουργήθηκε ειδικά για το περιβάλλον των έξυπνων καρτών. Στο σχήμα 1.3.1 παρουσιάζονται οι δύο τεχνολογίες σε αντιστοιχία [3].



Σχήμα 1.2.1 – Οι τεχνολογίες J2ME και JavaCard

Στο κεφάλαιο 2 γίνεται αναλυτική περιγραφή της αρχιτεκτονικής και του τρόπου λειτουργία της Java Card τεχνολογίας.

ΚΕΦΑΛΑΙΟ 2

JAVA CARD

Κεφάλαιο 2

2. JAVA CARD

2.1 Έξυπνες Κάρτες

Οι έξυπνες κάρτες (smart cards) είναι πολύ χρήσιμες σε θέματα που άπτονται της προσωπικής ασφάλειας. Μπορούν να χρησιμοποιηθούν για να προσθέσουν πιστοποίηση και ασφαλή πρόσβαση σε Πληροφοριακά Συστήματα, τα οποία απαιτούν ένα υψηλό επίπεδο ασφάλειας. Η πληροφορία που αποθηκεύεται σε μία έξυπνη κάρτα είναι φορητή. Με την τεχνολογία της Java Card μπορούν να μεταφερθούν παντού πολύτιμες και ευαίσθητες προσωπικές πληροφορίες όπως το ιατρικό ιστορικό, αριθμούς πιστωτικών καρτών ή ακόμη και ηλεκτρονικά μετρητά υπόλοιπα σε ένα μέσο που καταλαμβάνει λίγο χώρο και είναι πολύ ασφαλές.

Οι έξυπνες κάρτες δεν είναι κάτι καινούριο. Εμφανίστηκαν στην Ευρώπη δύο δεκαετίες πριν, με τη μορφή καρτών μνήμης (memory cards), όχι και τόσο έξυπνες, οι οποίες συνήθιζαν να αποθηκεύουν κρίσιμες τηλεφωνικές πληροφορίες με σκοπό την μείωση των κλεφτών από τηλέφωνα που τότε λειτουργούσαν με κέρματα.

Η τεχνολογία των έξυπνων καρτών είναι μία βιομηχανία προτύπων ορισμένη και ελεγχόμενη από το Joint Technical Committee 1 (JTC1) του International Standards Organization (ISO) και του International Electronic Committee (IEC). Η σειρά από τα διεθνή πρότυπα ISO/IEC 7816, εμφανίστηκε το 1987 με τις τελευταίες ενημερώσεις

το 2003, ορίζοντας ποικιλία απόψεων για την έξυπνη κάρτα, περιλαμβάνοντας φυσικά χαρακτηριστικά, φυσικές επικοινωνίες, ηλεκτρονικά σήματα και πρωτόκολλα μετάδοσης, απαιτήσεις, αρχιτεκτονική ασφάλειας, αίτηση αναγνώρισης και κοινά στοιχεία δεδομένων.

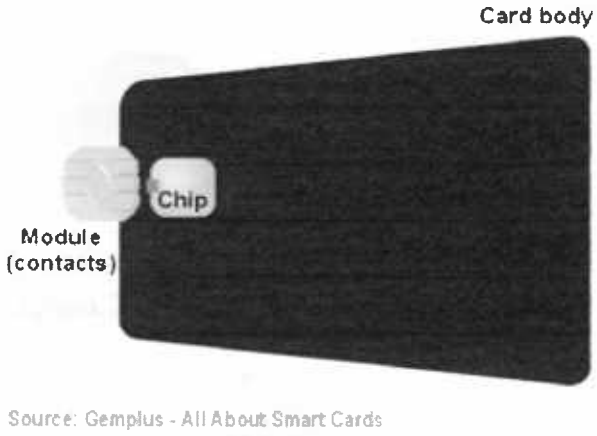
Μια έξυπνη κάρτα είναι ίδια στο μέγεθος με μια χαρακτηριστική πιστωτική κάρτα και είναι αρκετά δύσκολο να πλαστογραφηθεί. Η έξυπνη κάρτα αποθηκεύει και επεξεργάζεται τις πληροφορίες μέσω των ηλεκτρονικών κυκλωμάτων που ενσωματώνονται στο πλαστικό υπόστρωμα της κάρτας. Υπάρχουν δύο βασικοί τύποι έξυπνων καρτών: μνήμης και έξυπνη. Μια κάρτα μνήμης αποθηκεύει τα στοιχεία τοπικά, αλλά δεν περιέχει κεντρική μονάδα επεξεργασίας για την εκτέλεση υπολογισμών. Μια έξυπνη κάρτα περιλαμβάνει έναν μικροεπεξεργαστή και μπορεί να εκτελέσει τους υπολογισμούς στα τοπικά-αποθηκευμένα στοιχεία.

Σε μερικές περιοχές που γίνεται χρήση των έξυπνων καρτών, αυτές είναι απλά κάρτες μνήμης οι οποίες μερικά παρέχουν προστατευμένη μη-ευμετάβλητη αποθήκευση. Οι περισσότερο προηγμένες έξυπνες κάρτες έχουν και μικροεπεξεργαστές και μνήμη, τόσο για ασφαλή επεξεργασία όσο και για αποθήκευση δεδομένων. Αυτές οι κάρτες μπορούν να χρησιμοποιηθούν για εφαρμογές ασφάλειας, οι οποίες χρησιμοποιούν αλγορίθμους δημοσίου ή διαμοιραζόμενου κλειδιού (shared-key). Η μη-ευμετάβλητη μνήμη μέσα σε μια έξυπνη κάρτα είναι ο πιο πολύτιμος πόρος της και μπορεί να χρησιμοποιηθεί για να αποθηκεύει μυστικά κλειδιά και ψηφιακά πιστοποιητικά. Μερικές έξυπνες κάρτες έχουν ξεχωριστό κρυπτογραφικό συνεπεξεργαστή που υποστηρίζει τέτοιους αλγόριθμους όπως RSA, AEC και (3)DES.

Οι έξυπνες κάρτες δεν περιέχουν μπαταρία και γίνονται ενεργές μόνο όταν συνδέονται με έναν αναγνώστη καρτών. Όταν συνδέονται, μετά την εκκίνηση, η κάρτα παραμένει παθητική, περιμένοντας να λάβει μια εντολή από μία εφαρμογή - πελάτη (host).

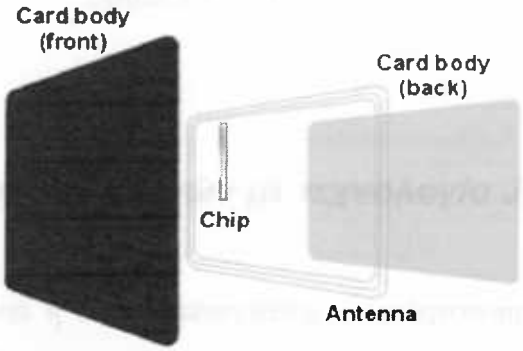
Οι έξυπνες κάρτες μπορούν να είναι contact ή contactless. Όπως τα ονόματα υποδηλώνουν, οι contact κάρτες λειτουργούν επικοινωνώντας μέσω φυσικής επαφής μεταξύ ενός αναγνώστη καρτών και των επαφών των 8-pin των έξυπνων καρτών. Οι contactless κάρτες επικοινωνούν με τη βοήθεια ενός σήματος ραδιοσυχνότητας, με

μια τυπική ακτίνα μικρότερη από 70 εκατοστά. Η ράδιο-επικοινωνία των contactless έξυπνων καρτών βασίζεται σε τεχνολογία παρόμοια με αυτή της Radio Frequency ID (RFID) tags που χρησιμοποιείται στα καταστήματα για τον εντοπισμό των κλεφτών και την ανίχνευση του αποθέματος. Στα σχήματα 2.1.1 και 2.1.2 απεικονίζονται τα δύο είδη έξυπνων καρτών [3].



Source: Gemplus - All About Smart Cards

Σχήμα 2.1.1 – Contact έξυπνη κάρτα



Source: Gemplus - All About Smart Cards

Σχήμα 2.1.2 – Contactless έξυπνη κάρτα

Η Java Card τεχνολογία υπάρχει και σε μορφές διαφορετικές από τις τυπικές έξυπνες κάρτες, όπως έξυπνα κουμπιά (smart buttons) και USB tokens τα οποία φαίνονται στα σχήματα 2.1.3 και 2.1.4. Αυτά μπορούν να χρησιμοποιηθούν όπως οι έξυπνες κάρτες, για την αυθεντικοποίηση χρηστών ή για να κουβαλούν ευαίσθητες πληροφορίες. Τα

έξυπνα κουμπιά, περιλαμβάνουν μπαταρία και είναι contact-based, ενώ τα USB tokens μπορούν να προσαρμοστούν απευθείας σε μία USB πόρτα του PC χωρίς ανάγκη για contact ή contactless αναγνώστη. Η χρήση του αφορά κυρίως το κλείδωμα λογισμικού όπου το ξεκλείδωμά του επιτυγχάνεται μέσω του USB token. Αμφότερα παρέχουν τις ίδιες προγραμματιστικές δυνατότητες, όπως και οι έξυπνες κάρτες .

Σχήμα 2.1.3 – Έξυπνο κουμπί (smart button)

Σχήμα 2.1.4 – USB token

2.2 Προδιαγραφές των καρτών με τεχνολογία Java Card

Πριν από αρκετά χρόνια η Sun Microsystems συνειδητοποίησε τη δυναμική των έξυπνων καρτών καθώς και παρόμοιων συσκευών με περιορισμένους πόρους και ορίστηκε ένα σύνολο από προδιαγραφές για ένα υποσύνολο της Java τεχνολογίας, ώστε να δημιουργηθούν εφαρμογές για αυτές, τα Java Card applets. Έτσι το 1996 δημιουργήθηκε η πρώτη κάρτα που χρησιμοποιούσε την Java τεχνολογία αλλά επειδή αφορούσε πολύ περιορισμένο κομμάτι της αγοράς δεν εξαπλώθηκε πολύ. Μια συσκευή (έξυπνη κάρτα) που υποστηρίζει αυτές τις προδιαγραφές αναφέρεται ως Java Card πλατφόρμα (Java Card platform). Σε μια Java Card πλατφόρμα, μπορούν

να συνυπάρχουν με ασφάλεια πολλαπλές εφαρμογές από διαφορετικούς κατασκευαστές.

Μία τυπική έξυπνη κάρτα βασισμένη στην Java Card τεχνολογία έχει έναν 8 bit ή 16 bit επεξεργαστή με ταχύτητα στα 3.7 Mhz, με 1K RAM και πάνω από 16K μνήμης EEPROM ή Flash. Οι υψηλών προδιαγραφών έξυπνες κάρτες έρχονται με ξεχωριστό επεξεργαστή, κρυπτογραφικό τσιπ και μνήμη για κρυπτογράφηση, μερικές μάλιστα έρχονται και με 32 bit επεξεργαστή.

Η προδιαγραφή της Java Card τεχνολογίας, επί του παρόντος στην έκδοση 2.2 αποτελείται από τρία μέρη :

- Την προδιαγραφή για την Java Card Virtual Machine, η οποία ορίζει ένα υποσύνολο της Java και μία εικονική μηχανή (virtual machine) για έξυπνες κάρτες.

- Την προδιαγραφή για το Java Card Runtime Environment, η οποία περαιτέρω ορίζει την συμπεριφορά κατά των χρόνο εκτέλεσης για τις βασισμένες σε Java έξυπνες κάρτες.

- Την προδιαγραφή για το Java Card API, η οποία ορίζει το βασικό πλαίσιο (framework) , πακέτα επέκτασης και κλάσεις για εφαρμογές έξυπνων καρτών.

Η Sun παρέχει το Java Card Development Kit (JCDK), το οποίο περιλαμβάνει μία υλοποίηση του Java Card Runtime Environment και της Java Card Virtual Machine καθώς και άλλα εργαλεία για την ανάπτυξη Java Card applets.

2.3 Java Card Virtual Machine

Η προδιαγραφή της Java Card Virtual Machine (JCVM) ορίζει ένα υποσύνολο της γλώσσας προγραμματισμού Java και μία Java συμβατή VM για έξυπνες κάρτες, συμπεριλαμβάνοντας δυαδικές αναπαραστάσεις δεδομένων, format αρχείων και σύνολο εντολών του JCVM.

Η VM για την Java Card πλατφόρμα είναι υλοποιημένη σε δύο μέρη, με το ένα μέρος εξωτερικό της κάρτας και το άλλο να τρέχει πάνω στην κάρτα [2][3]. Το μέρος της VM που υλοποιείται πάνω στην κάρτα μεταφράζει bytecode ,διαχειρίζεται κλάσεις και αντικείμενα και ούτω καθεξής. Το εξωτερικό μέρος της Java VM είναι ένα εργαλείο ανάπτυξης, που τυπικά αναφέρεται ως Java Card Converter tool, το οποίο φορτώνει, επαληθεύει και ετοιμάζει περαιτέρω τις Java κλάσεις σε ένα applet για εκτέλεση πάνω στην κάρτα. Τα εξερχόμενα από το Converter tool είναι ένα αρχείο Converter APplet (CAP), το αρχείο που περιέχει όλες τις κλάσεις ενός πακέτου Java μέσα σε μία έτοιμη να φορτωθεί και να εκτελεστεί δυαδική αναπαράσταση. Ο converter επαληθεύει ότι οι κλάσεις συμμορφώνονται με τις προδιαγραφές της Java Card.

Η JCVM υποστηρίζει μόνο ένα περιορισμένο υποσύνολο της γλώσσας προγραμματισμού Java, παρότι διατηρεί πολλά από τα οικία χαρακτηριστικά συμπεριλαμβανομένων των αντικειμένων, της κληρονομικότητας, των πακέτων, της δυναμικής δημιουργίας αντικειμένων, των εικονικών μεθόδων, των διεπαφών (interfaces) και των εξαιρέσεων (exceptions). Η προδιαγραφή της JCVM αποκλείει την υποστήριξη για ένα αριθμό στοιχείων της γλώσσας τα οποία θα χρησιμοποιούσαν μεγάλο κομμάτι από την περιορισμένη μνήμη των έξυπνων καρτών. Στο πίνακα του σχήματος 2.3.1 παρουσιάζονται συνοπτικά τα περισσότερα:

Περιορισμοί της τεχνολογίας Java Card	
Χαρακτηριστικά γλώσσας (language features)	Dynamic class loading, security manager, threads, object cloning και μερικές λειτουργίες ελέγχου πρόσβασης πακέτων δεν υποστηρίζονται
Λέξεις Κλειδιά (keywords)	`native`, `synchronized`, `transient`, `volatile`, `strictfp` δεν υποστηρίζονται
Τύποι δεδομένων (types)	Δεν υπάρχει υποστήριξη για `char`, `double`, `float` και `long` ή για πολυδιάστατους πίνακες.
Κλάσεις (classes) και διεπαφές (interfaces)	Οι Java core API κλάσεις και οι διεπαφές (`java.io`, `java.lang`, `java.util`) δεν υποστηρίζονται εκτός για τα `Object` και `Throwable` και οι περισσότερες μέθοδοι των `Object` και `Throwable` δεν είναι διαθέσιμες.

Εξαιρέσεις (exceptions)	Κάποιες `Exception` και `Error` υποκλάσεις παραβλέπονται επειδή οι εξαιρέσεις και τα λάθη τα οποία περιλαμβάνουν δεν μπορούν να ανακύψουν στην Java Card πλατφόρμα.

Σχήμα 2.3.1 – Σύνοψη των περιορισμών της τεχνολογίας Java Card

Σύμφωνα με τον περιορισμό της μνήμης η προδιαγραφή JCVM ορίζει περιορισμούς σε πολλά προγραμματιζόμενα χαρακτηριστικά. Ο πίνακας του σχήματος 2.3.2 συνοψίζει τους περιορισμούς της JCVM. Θα πρέπει να σημειωθεί ότι πολλοί από αυτούς τους περιορισμούς είναι διαφανείς κατά την ανάπτυξη μιας Java Card εφαρμογής.

Περιορισμοί της Java Card Virtual Machine	
Πακέτα (packages)	• Ένα πακέτο μπορεί να αναφέρεται μέχρι σε 128 άλλα πακέτα • Ένα πλήρες όνομα πακέτου περιορίζεται σε 255 bytes • Ένα πακέτο μπορεί να έχει μέχρι 255 κλάσεις
Κλάσεις (classes)	• Μία κλάση μπορεί άμεσα ή έμμεσα να υλοποιεί μέχρι 15 διεπαφές (interfaces) • Μία διεπαφή μπορεί να κληρονομεί μέχρι από 14 άλλες διεπαφές • Ένα πακέτο μπορεί να έχει το πολύ 256 στατικές μεθόδους αν περιέχει applets ή 255 αν δεν περιέχει • Μία κλάση μπορεί να υλοποιεί μέχρι 128 στιγμιότυπα public ή protected μεθόδων

Σχήμα 2.3.2 – Σύνοψη περιορισμών της Java Card Virtual Machine

Στην Java Card Virtual Machine, όπως και στην J2SE Virtual Machine , τα αρχείων κλάσεων (*.class) είναι βασικά, αλλά στην προδιαγραφή της JCVM ορίζονται δύο άλλα format αρχείων για να επιτύχουν περαιτέρω ανεξαρτησία της πλατφόρμας, τα Converted Applet (*.CAP) τα οποία προαναφέρθηκαν και τα Export (*.exp).

2.5 Java Card API

Η προδιαγραφή του Java Card API ορίζει ένα μικρό υποσύνολο από το παραδοσιακό Java API- ακόμη μικρότερο και από αυτό του J2ME CLDC. Δεν υπάρχει καθόλου υποστήριξη για Strings ή για πολύπλοκα νήματα (threads). Δεν υπάρχουν κλάσεις όπως Boolean και Integer και καθόλου Class ή System κλάσεις.

Επιπρόσθετα στο μικρό υποσύνολο των βασικών κλάσεων της Java, το Java Card Framework ορίζει το δικό του σύνολο με βασικές κλάσεις, ειδικές για να υποστηρίξει εφαρμογές Java Card. Αυτά περιέχονται στα παρακάτω πακέτα :

- `java.io` : ορίζει μία κλάση εξαίρεσης, τη βασική κλάση IOException, για να ολοκληρώσει την ιεραρχία των RMI εξαιρέσεων. Καμία από τις άλλες παραδοσιακές κλάσεις java.io δεν περιλαμβάνονται.

- `java.lang` : ορίζει τις Object και Throwable κλάσεις από τις οποίες εκλείπουν πολλές μέθοδοι από τις αντιστοιχες κλάσεις στην J2SE. Επίσης αυτό ορίζει ένα αριθμό από κλάσεις εξαίρεσης : τη βασική κλάση Exception, ποικίλες runtime εξαιρέσεις και την CardException. Καμία από τις άλλες παραδοσιακές java.lang κλάσεις δεν περιλαμβάνονται.

- `java.rmi` : ορίζει το Remote interface και την κλάση RemoteException. Καμία από τις παραδοσιακές κλάσεις java.rmi δεν περιλαμβάνεται. Η υποστήριξη για Remote Method Invocation (RMI) περιλαμβάνεται για να απλοποιήσει τη μεταφορά και την ενσωμάτωση με συσκευές που χρησιμοποιούν την Java Card τεχνολογία.

- `javacard.framework` : ορίζει τα interfaces, τις κλάσεις και τις εξαιρέσεις που συνθέτουν τη βάση του Java Card Framework. Ορίζει σημαντικές έννοιες, όπως Personal Identification Number (PIN), Application Protocol Data Unit (APDU), Java Card applet (Applet), Java Card System (JCSystem) και την Util κλάση η οποία περιέχει χρήσιμες μεθόδους. Επίσης ορίζει

ποικίλες ISO7816 σταθερές και ποικίλες εξαιρέσεις οι οποίες αφορούν αποκλειστικά την Java Card .

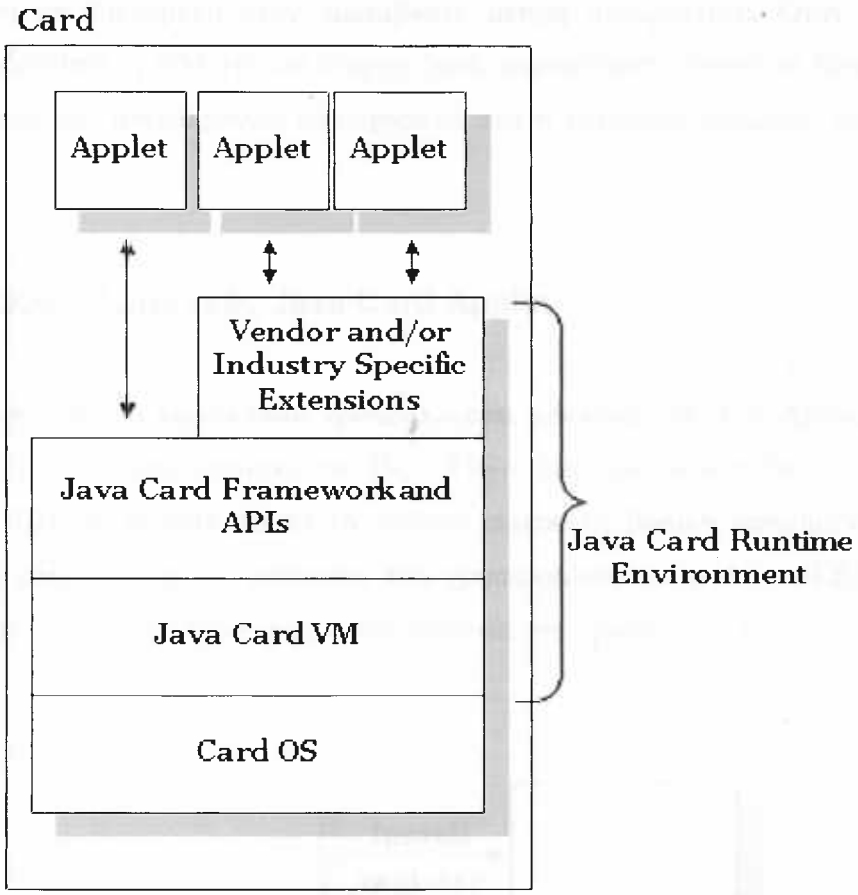
- `javacard.framework.service` : ορίζει τα interfaces, τις κλάσεις και τις εξαιρέσεις για τα *services*. Ένα service επεξεργάζεται εισερχόμενες εντολές με την μορφή των APDU.

- `javacard.security` : ορίζει τις κλάσεις και τα interfaces για το Java Card Security Framework. Η προδιαγραφές τις Java Card ορίζουν ένα εύρωστη security API το οποίο περιλαμβάνει διάφορους τύπους ιδιωτικών και δημόσιων κλειδιών και αλγορίθμων, μεθόδους για τον υπολογισμό των cyclic redundancy checks (CRCs), των message digests και των υπογραφών.

- `javacardx.crypto` , `javacardx.rmi` : αποτελούν πακέτα επέκτασης τα οποία αφορούν το μεν πρώτο λειτουργίες κρυπτογράφησης ενώ το δεύτερο τις κλάσεις του Java Card RMI.

2.6 Java Card Runtime Environment

Η προδιαγραφή του Java Card Runtime Environment ορίζει τον κύκλο ζωής της Java Card VM, τον κύκλο ζωής του applet, πως τα applets επιλέγονται και ξεχωρίζουν το ένα από το άλλο, συναλλαγές και διαμοιρασμός των αντικειμένων. Το JCRE παρέχει μία διεπαφή ανεξάρτητη πλατφόρμας στις υπηρεσίες που παρέχονται από το λειτουργικό σύστημα της κάρτας. Όπως φαίνεται στο σχήμα 2.6.1 το JCRE αποτελείται από τη Java Card Virtual Machine, το Java Card API και τις ιδιαίτερες επεκτάσεις του κάθε κατασκευαστή καρτών [3][8].



Σχήμα 2.6.1 – Η θέση του JCRE στην Java Card

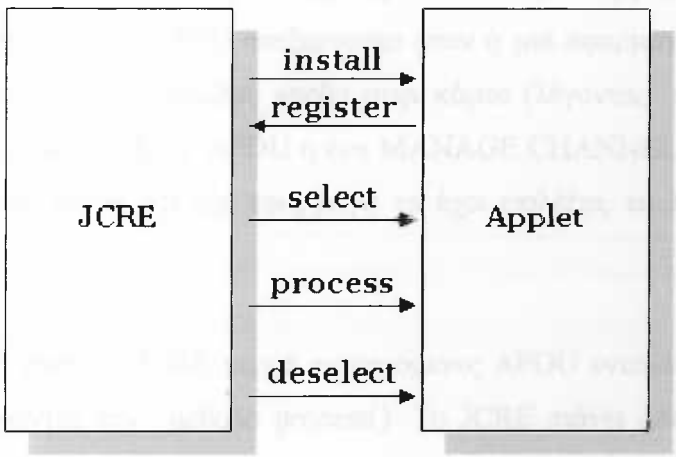
2.6.1 Κύκλος ζωής της Java Card VM

Ο κύκλος ζωής της JCVM συμπίπτει με αυτόν της ίδιας της κάρτας. Ξεκινάει κάποια χρονική στιγμή μετά την κατασκευή και τον έλεγχο της κάρτας και πριν την έκδοσή της σε κάποιο πελάτη και τελειώνει όταν η κάρτα ακυρώνεται και καταστρέφεται. Η JCVM δεν σταματά όταν αφαιρείται η τροφοδοσία από την κάρτα, αφού η κατάστασή της συγκρατείται στην αμετάβλητη μνήμη της κάρτας (EEPROM). Αρχίζοντας η JCVM αρχικοποιεί το JCRE και δημιουργεί όλα τα αντικείμενα του JCRE framework, τα οποία ζουν για όλο το χρόνο ζωής της JCVM. Μετά την εκκίνηση της JCVM, όλες οι αλληλεπιδράσεις με την κάρτα είναι, σε γενικές γραμμές, ελεγχόμενες από ένα από τα applets της κάρτας. Όταν η ισχύς αφαιρείται από την κάρτα, κάθε δεδομένο που περιέχεται στη RAM χάνεται, αλλά κάθε

κατάσταση αποθηκευμένη στην αμετάβλητη μνήμη συγκρατείται. Όταν η κάρτα ανατροφοδοτείται, η VM γίνεται ενεργή ξανά, οποιαδήποτε στιγμή οι καταστάσεις της VM και των αντικειμένων επανέρχονται και η εκτέλεση περιμένει για κάποια είσοδο.

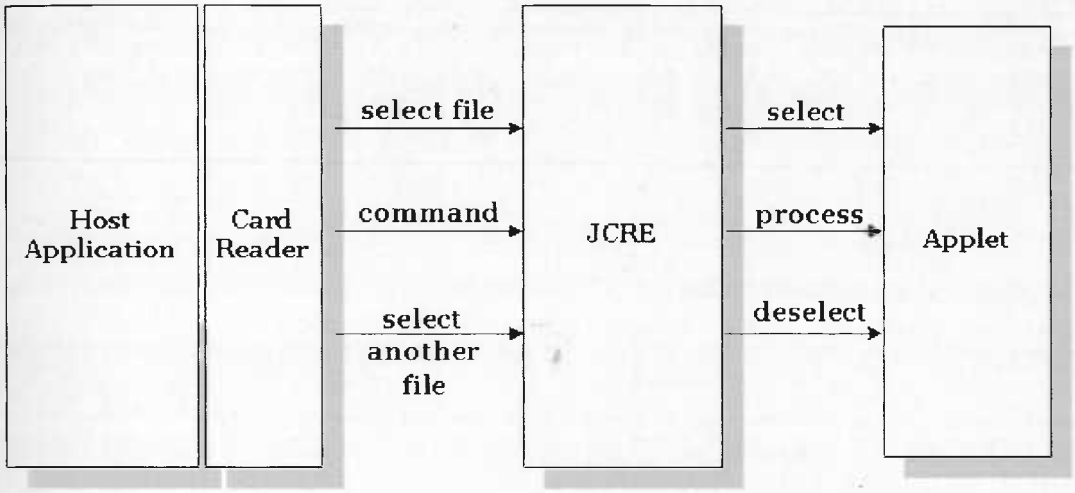
2.6.2 Κύκλος ζωής ενός Java Card Applet

Κάθε applet σε μία κάρτα είναι προσδιορίζεται μοναδικά από ένα Application ID (AID). Ένα AID, όπως ορίστηκε στο ISO 7816-5, είναι μια ακολουθία μεταξύ 5 και 16 bytes. Όλα τα applets πρέπει να κάνουν extend τη βασική αφηρημένη κλάση Applet, η οποία ορίζει τις μεθόδους που χρησιμοποιούνται από το JCRE για να ελέγχει τον κύκλο ζωής του applet, όπως φαίνεται στο σχήμα 2.6.2.1.



Σχήμα 2.6.2.1 – Μέθοδοι που καθορίζουν τον κύκλο ζωής ενός applet

Ο κύκλος ζωής ενός applet αρχίζει όταν το applet φορτώνεται στην κάρτα και το JCRE καλώντας τη στατική μέθοδο του applet Applet.install(), και το applet καταχωρείται στο JCRE καλώντας την Applet.register(). Μόλις το applet είναι εγκατεστημένο και καταχωρημένο, είναι σε μη-επιλεγμένη κατάσταση, διαθέσιμο για επιλογή και APDU επεξεργασία. Στο σχήμα 2.6.2.2 παρατηρούμε την λειτουργία των μεθόδων του applet.



Σχήμα 2.6.2.2 – Χρήση των μεθόδων του Java Card applet

Κατά τη διάρκεια που είναι σε μη-επιλεγμένη κατάσταση, το applet είναι ανενεργό. Ένα applet επιλέγεται για APDU επεξεργασία όταν η μια εφαρμογή ζητάει από το JCRE να επιλέξει ένα συγκεκριμένο applet στην κάρτα (λέγοντας στον αναγνώστη κάρτας να στείλει ένα SELECT APDU ή ένα MANAGE CHANNEL APDU). Για να γνωστοποίηση στο applet ότι μία εφαρμογή το έχει επιλέξει, το JCRE καλεί την μέθοδο select ().

Μόλις η επιλογή γίνει, το JCRE περνά εισερχόμενες APDU εντολές στο applet για επεξεργασία καλώντας την μέθοδο process(). Το JCRE πιάνει όσες εξαιρέσεις το applet αποτυγχάνει τα πιάσει.

Το applet παύει να είναι επιλεγμένο όταν η εφαρμογή λέει στο JCRE να επιλέξει άλλο applet. Το JCRE επισημαίνει στο ενεργό applet ότι έχει δεν είναι επιλεγμένο πλέον καλώντας την μέθοδο deselect(), η οποία τυπικά πραγματοποιεί κάθε λογικό ξεκαθάρισμα και επιστρέφει το applet σε ανενεργή, μη επιλεγμένη κατάσταση.

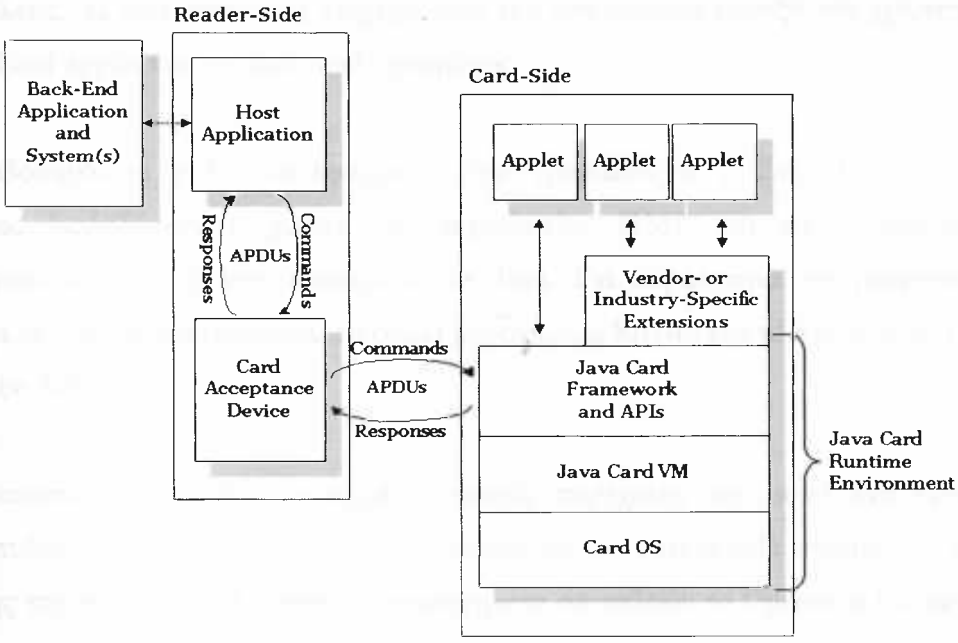
ΚΕΦΑΛΑΙΟ 3

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΩΝ JAVA CARD ΕΦΑΡΜΟΓΩΝ

Κεφάλαιο

3.1 Στοιχεία μιας Java Card εφαρμογής

Μια ολοκληρωμένη εφαρμογή Java Card αποτελείται από μία back-end εφαρμογή και συστήματα, μία host εφαρμογή (εκτός κάρτας), μία συσκευή διεπαφής (interface device – card reader) και το applet που βρίσκεται πάνω στην κάρτα, τα πιστοποιητικά του χρήστη και το συνοδευτικό λογισμικό [3].



Σχήμα 3.1.1 – Η αρχιτεκτονική μιας Java Card εφαρμογής

Μία τυπική εφαρμογή Java Card δεν είναι αυτόνομη αλλά αντιθέτως περιλαμβάνει card-side, reader-side και back-end στοιχεία. Ας δούμε κάθε στοιχείο με περισσότερη λεπτομέρεια.

3.1.1 Back-end εφαρμογές και συστήματα

Οι back-end εφαρμογές παρέχουν υπηρεσίες που υποστηρίζουν τα Java applets που βρίσκονται πάνω στην κάρτα. Για παράδειγμα, μία back-end εφαρμογή μπορεί να παρέχει συνδεσιμότητα με ασφαλή συστήματα, τα οποία μαζί με πιστοποιητικά που βρίσκονται στην κάρτα, εξασφαλίζουν ισχυρή ασφάλεια. Σε ένα ηλεκτρονικό σύστημα πληρωμής, η back-end εφαρμογή θα μπορούσε να παρέχει πρόσβαση σε πιστωτική κάρτα και σε άλλες πληροφορίες πληρωμής.

3.1.2 Η reader-side host εφαρμογή

Η host εφαρμογή βρίσκεται σε ένα τερματικό όπως ένας ηλεκτρονικός υπολογιστής, ένα ηλεκτρονικό τερματικό πληρωμής, ένα κινητό τηλέφωνο ή ένα υποσύστημα

ασφάλειας. Η host εφαρμογή διαχειρίζεται την επικοινωνία μεταξύ του χρήστη, του Java Card applet και της back-end εφαρμογής.

Παραδοσιακά, οι reader-side εφαρμογές ήταν γραμμένες σε C. Στις μέρες μας, είναι ευρέως διαδεδομένη η χρήση της τεχνολογίας J2ME και κάνει πιθανή την πραγματοποίηση της host εφαρμογής σε Java. Για παράδειγμα, θα μπορούσε να τρέχει σε ένα κινητό τηλέφωνο το οποίο υποστηρίζει MIDP, και το Security and Trust Service API.

Οι κατασκευαστές έξυπνων καρτών τυπικά παρέχουν, όχι μόνο ένα εργαλείο ανάπτυξης, αλλά και APIs για να υποστηρίξουν τις reader-side εφαρμογές καθώς επίσης και τα Java Card applets. Παραδείγματα αποτελούν το Opencard Framework, ένα σύνολο από APIs βασισμένα σε Java τα οποία κρύβουν μερικές από τις λεπτομέρειες της αλληλεπίδρασης με αναγνώστες καρτών από διαφορετικούς κατασκευαστές, το μοντέλο κατανεμημένων αντικειμένων Java Card Remote Method Invocation και το Security and Trust Service API (SATSA).

3.1.3 Η reader-side Card Acceptance Device

Η συσκευή αποδοχής κάρτας (card acceptance device - CAD) είναι η συσκευή διασύνδεσης που βρίσκεται ανάμεσα στην host εφαρμογή και στη συσκευή Java Card. Ένα CAD παρέχει ισχύ στην κάρτα, καθώς επίσης και ηλεκτρική ή RF επικοινωνία με αυτή. Ένα CAD μπορεί να είναι ένας αναγνώστης συνδεμένος με έναν ηλεκτρονικό υπολογιστή χρησιμοποιώντας μία σειριακή θύρα ή μπορεί να είναι ενσωματωμένο μέσα σε ένα τερματικό, όπως ένα ηλεκτρονικό τερματικό πληρωμής σε ένα εστιατόριο ή σε ένα πρατήριο καυσίμων. Το CAD προωθεί τις Application Protocol Data Unit (APDU) εντολές από την host εφαρμογή στην κάρτα και τις απαντήσεις από την κάρτα στην host εφαρμογή. Μερικές CADs έχουν πληκτρολόγιο για την εισαγωγή του προσωπικού κωδικού (PIN) και πιθανώς και μία οθόνη.

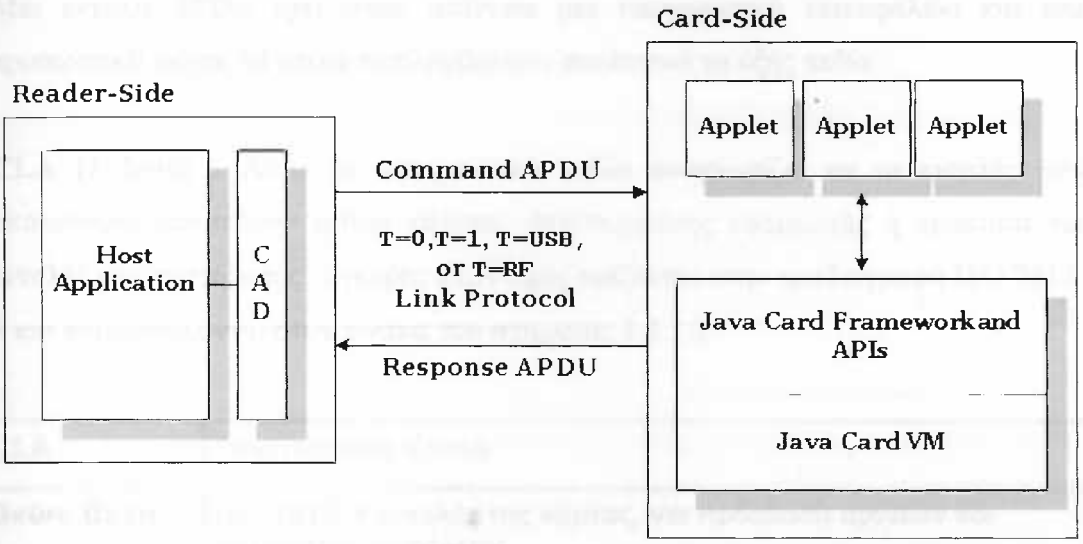
3.1.4 Τα card-side applets και το περιβάλλον

Η πλατφόρμα της Java Card είναι ένα περιβάλλον πολλαπλών εφαρμογών. Όπως παρουσιάζεται και στο σχήμα 3.1.1, ένα ή περισσότερα Java Card Applets μπορούν να ανήκουν στην κάρτα, μαζί με λογισμικό υποστήριξης (το λειτουργικό σύστημα της κάρτας και το Java Card Runtime Environment).

Όλα τα Java Card applets κάνουν extend την βασική κλάση Applet και πρέπει να υλοποιούν τις μεθόδους install() και process(). Το JCRE καλεί την install() όταν εγκαθιστά το applet, και την process() κάθε φορά υπάρχει ένα εισερχόμενο APDU για το applet. Τα applets γίνονται instantiated όταν φορτώνονται και παραμένουν ζωντανά όταν η τροφοδοσία σταματάει. Ένα card applet συμπεριφέρεται σαν server και έχει παθητικό ρόλο. Περιμένει για APDU αιτήσεις αφού πρώτα έχει επιλογή από κάποια host εφαρμογή και αποστέλλει απαντήσεις. Στην παράγραφο 2.6.2 είδαμε αναλυτικά των κύκλο ζωής ενός applet, που περιλαμβάνει όλες τις παραπάνω λειτουργίες.

3.2 Επικοινωνία με τα Java Card applets

Το μοντέλο διαβίβασης μηνυμάτων είναι η βάση για όλες τις επικοινωνίες με Java Cards και απεικονίζεται στο σχήμα 3.2.1. Στο κέντρο του μοντέλου αυτού είναι το Application Protocol Data Unit (APDU), ένα λογικό πακέτο δεδομένων το οποίο ανταλλάσσεται μεταξύ του CAD και του Java Card Framework. Το Java Card Framework λαμβάνει και προωθεί στο κατάλληλο applet κάθε εισερχόμενη εντολή APDU που αποστέλλεται από το CAD. Το applet επεξεργάζεται την APDU εντολή και επιστρέφει μία APDU απάντηση [2][3][7]. Τα APDUs ακολουθούν τα παγκόσμια πρότυπα ISO/IEC 7816-3 και 7816-4 .



Σχήμα 3.2.1 – Το μοντέλο διαβίβασης μηνυμάτων

Η επικοινωνία μεταξύ του αναγνώστη (reader) και της κάρτας βασίζεται συνήθως σε κάποιο από τα δύο πρωτόκολλα , το byte-oriented T=0 και το block-oriented T=1. Εναλλακτικά, μπορούν να χρησιμοποιηθούν και πρωτόκολλα που αναφέρονται ως T=USB και T= RF. Η APDU κλάση του JCRE αποκρύπτει κάποιες λεπτομέρειες του πρωτοκόλλου από την εφαρμογή, αλλά όχι όλες, αφού το πρωτόκολλο T=0 είναι μάλλον πολύπλοκο.

3.2.1 Command APDU

Η δομή μιας εντολής (command) APDU ελέγχεται από την τιμή του πρώτου byte και στις περισσότερες περιπτώσεις έχει τη μορφή του σχήματος 3.2.1.1.

COMMAND APDU						
HEADER (υποχρεωτικό)				**BODY** (προαιρετικό)		
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**

Σχήμα 3.2.1.1 – Η δομή μιας command APDU

Μια εντολή APDU έχει όπως φαίνεται μια υποχρεωτική επικεφαλίδα και ένα προαιρετικό σώμα, τα οποία περιλαμβάνουν αναλυτικά τα εξής πεδία :

CLA (1 byte) : Αυτό το υποχρεωτικό πεδίο αναγνωρίζει αν οι εντολές που ακολουθούν αποτελούν τμήμα κάποιας συγκεκριμένης εφαρμογής ή πρόκυται για εντολές του συστήματος. Έγκυρες CLA τιμές ορίζονται στην προδιαγραφή ISO 7816-4 και παρουσιάζονται στον πίνακα του σχήματος 3.2.1.2.

CLA	Instruction Class
0x0n, 0x1n	ISO 7816-4 εντολές της κάρτας, για πρόσβαση αρχείων και λειτουργίες ασφάλειας
20 to 0x7F	Κρατημένες (Reserved)
0x8n or 0x9n	ISO/IEC 7816-4 format, μπορούμε να τις χρησιμοποιήσουμε για τις εντολές δικών μας εφαρμογών
0xAn	Εντολές ειδικά για την εφαρμογή ή τον κατασκευαστή
B0 to CF	ISO/IEC 7816-4 format , μπορούμε να τις χρησιμοποιήσουμε για τις εντολές δικών μας εφαρμογών
D0 to FE	Εντολές ειδικά για την εφαρμογή ή τον κατασκευαστή
FF	Κρατημένη για την επιλογή του τύπου πρωτοκόλλου

Σχήμα 3.2.1.2 – Οι κλάσεις εντολών που καθορίζει η τιμή του CLA

INS (1 byte) : Αυτό το υποχρεωτικό πεδίο υποδεικνύει μία συγκεκριμένη εντολή μέσα στην κλάση εντολών που αναγνωρίζεται στο πεδίο CLA. Το πρότυπο ISO 7816-4 καθορίζει τις βασικές εντολές που χρησιμοποιούνται για πρόσβαση σε δεδομένα που βρίσκονται στην κάρτα. Επιπρόσθετες λειτουργίες έχουν καθοριστεί αλλού στο πρότυπο, μερικές από τις οποίες είναι λειτουργίες ασφάλειας. Στον πίνακα του σχήματος 3.2.1.3 παρουσιάζονται κάποιες εντολές ISO 7816, που αφορούν λειτουργίες της κάρτας. Μπορούμε να καθορίσουμε δικές μας τιμές στο INS μόνο αν έχουμε χρησιμοποιήσει και την κατάλληλη τιμή στο CLA, σύμφωνα με το σχήμα 3.2.1.2.

INS Value	Command Description
0E	Erase Binary
20	Verify
70	Manage Channel
82	External Authenticate
84	Get Challenge
88	Internal Authenticate
A4	Select File
B0	Read Binary
B2	Read Record(s)
C0	Get Response
C2	Envelope
CA	Get Data
D0	Write Binary
D2	Write Record
D6	Update Binary
DA	Put Data
DC	Update Record
E2	Append Record

Σχήμα 3.2.1.3 – Τιμές του INS όταν η τιμή του CLA=0x

P1 (1 byte) : Αυτό το υποχρεωτικό πεδίο ορίζει την παράμετρο 1 της εντολής. Μπορεί να χρησιμοποιηθεί για να χαρακτηριστεί το πεδίο INS ή για δεδομένα εισόδου.

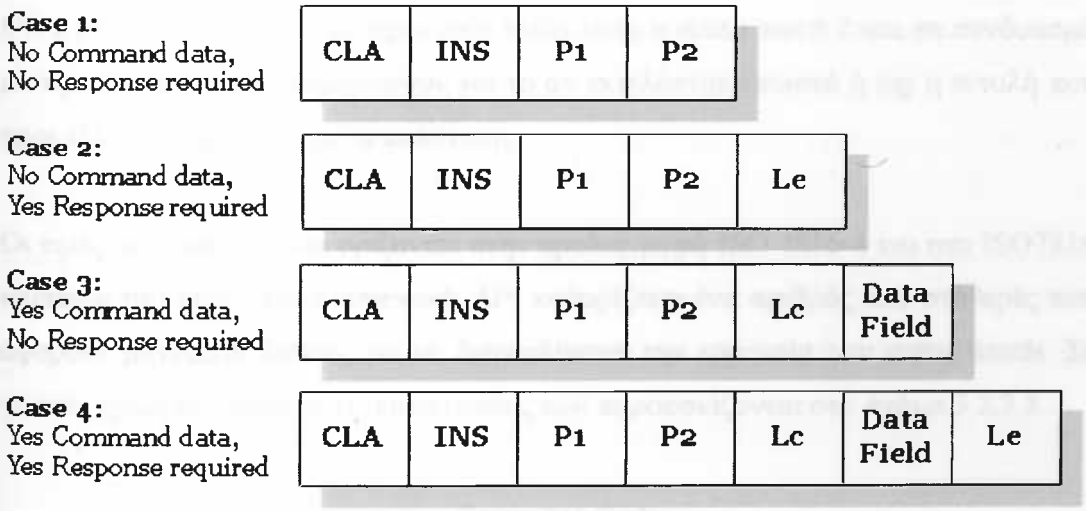
P2 (1 byte) : Αυτό το υποχρεωτικό πεδίο ορίζει την παράμετρο 2 της εντολής. Μπορεί να χρησιμοποιηθεί για να χαρακτηριστεί το πεδίο INS ή για δεδομένα εισόδου.

LC (1 byte) : Αυτό το προαιρετικό πεδίο είναι ο αριθμών των bytes στο πεδίο δεδομένων της εντολής.

Data field (Lc bytes) : Αυτό το προαιρετικό πεδίο κρατάει τα δεδομένα της εντολής και έχει μήκος τόσα bytes όσα ορίζονται στο πεδίο Lc.

Le (1byte) : Αυτό το προαιρετικό πεδίο καθορίζει το μέγιστο αριθμό bytes στο Data field της προσδοκώμενης απάντησης.

Βασιζόμενη στην παρουσία ή μη των δεδομένων της εντολής και στο εάν απαιτείται μια απάντηση, υπάρχουν τέσσερις παραλλαγές της εντολής APDU αν χρησιμοποιείται το πρωτόκολλο T=0, όπως φαίνεται στο σχήμα 3.2.1.4. Μία τυπική εφαρμογή θα χρησιμοποιήσει ποικίλες APDU εντολές με διαφορετική δομή η κάθε μία.



Σχήμα 3.2.1.5 – Πιθανές δομές μιας εντολής APDU

3.2.2 Response APDU

Η δομή μας απάντησης (response) APDU είναι πολύ πιο απλή από την αντίστοιχη της εντολής όπως φαίνεται στο σχήμα 3.2.2.1.

RESPONSE APDU		
BODY (προαιρετικό)	**TRAILER** (υποχρεωτικό)	
Data field	**SW1**	**SW2**

Σχήμα 3.2.2.1 - Η δομή μιας response APDU

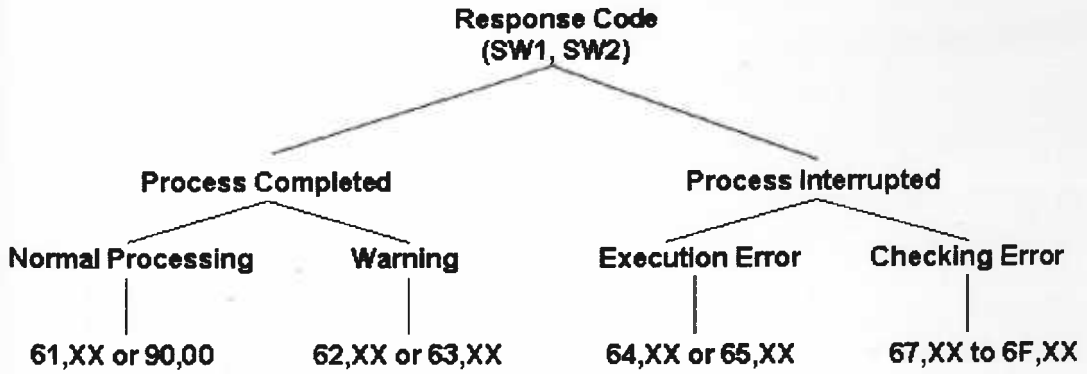
Μια APDU απάντηση έχει και αυτή προαιρετικά και υποχρεωτικά πεδία :

Data field (Le bytes) : Αυτό το προαιρετικό πεδίο περιέχει τα δεδομένα που επιστρέφονται από το applet

SW1 (1 byte) : Αυτό το υποχρεωτικό πεδίο είναι η status word 1 και σε συνδυασμό με την status word 2 ενημερώνουν για το αν εκτελέστηκε σωστά ή όχι η εντολή που προκάλεσε την συγκεκριμένη απάντηση.

SW1 (1 byte) : Αυτό το υποχρεωτικό πεδίο είναι η status word 2 και σε συνδυασμό με την status word 1 ενημερώνουν για το αν εκτελέστηκε σωστά ή όχι η εντολή που προκάλεσε την συγκεκριμένη απάντηση.

Οι τιμές των status words ορίζονται στην προδιαγραφή ISO 7816-4 και στο ISO7816 interface του Java Card Framework API καθορίζεται ένα αριθμός από σταθερές που αφορούν μηνύματα λάθους για να διευκολύνουν την ερμηνεία των status words. Σε γενικές γραμμές αφορούν τις καταστάσεις που παρουσιάζονται στο σχήμα 3.2.2.3.



Σχήμα 3.2.2.3 – Οι κωδικοί των status της απάντησης

3.2.3 Επεξεργασία των APDUs

Αφού μία εντολή APDU φθάσει στην κάρτα αυτή την επεξεργάζεται για μετά στέλνει την ανάλογη απάντηση. Πιο συγκεκριμένα, κάθε φορά που υπάρχει μία εισερχόμενη APDU για κάποιο επιλεγμένο applet, το JCRE καλεί την μέθοδο process(), περνώντας της την εισερχόμενη APDU σαν όρισμα. Το applet πρέπει να αναλύσει την APDU, να επεξεργαστεί τα δεδομένα, να δημιουργήσει μία απαντητική APDU και να επιστρέψει τον έλεγχο στο JCRE ακολουθώντας τα επόμενα βήματα :

1. Εξάγει από την APDU τα CLA και INS πεδία
2. Ανακτά τις παραμέτρους P1 και P2 καθώς και το πεδίο δεδομένων
3. Επεξεργάζεται τα δεδομένα της APDU
4. Δημιουργεί και στέλνει μία απάντηση
5. Το JCRE αποστέλλει την κατάλληλη status word πίσω στην host εφαρμογή

ΚΕΦΑΛΑΙΟ 4

ΤΕΡΜΑΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΚΑΙ JAVA CARD APPLETS

Κεφάλαιο

4. ΤΕΡΜΑΤΙΚΕΣ ΕΦΑΡΜΟΓΕΣ ΚΑΙ JAVA CARD APPLETS

Όπως προαναφέραμε οι εντολές APDU αποτελούν την βάση για όλες τις επικοινωνίας που έχουν να κάνουν με την Java Card. Έτσι, στη συνέχεια θα δούμε κάποιους τρόπους οι οποίοι, μας διευκολύνουν στο να επικοινωνήσει μια τερματική εφαρμογή που αναπτύσσουμε με μία Java Card, χρησιμοποιώντας πάντα APDUs.

4.1 Java Card RMI

Το μοντέλο επικοινωνίας Java Card RMI (JCRMI) βασίζεται σε ένα υποσύνολο του J2SE RMI μοντέλου κατανεμημένων αντικειμένων. Στο μοντέλο RMI μία εφαρμογή εξυπηρετητή δημιουργεί και κάνει προσβάσιμα απομακρυσμένα αντικείμενα και μία εφαρμογή πελάτη εξασφαλίζει απομακρυσμένες αναφορές σε απομακρυσμένα αντικείμενα ενώ μετά καλεί απομακρυσμένες μεθόδους σε αυτά. Στο JCRMI, το Java Card applet είναι ο εξυπηρετητής και η host εφαρμογή ο πελάτης [3].

Το JCRMI παρέχεται στο πακέτο επέκτασης javacardx.rmi απο την κλάση RMIService. Τα μηνύματα JCRMI περιλαμβάνονται μέσα στο APDU αντικείμενο που περνά στη μέθοδο RMIService. Με άλλα λόγια, το JCRMI παρέχει ένα μηχανισμό μοντέλου κατανεμημένων αντικειμένων πάνω από μοντέλο μηνυμάτων

που βασίζεται στις APDUs, με το οποίο ο εξυπηρετητής και ο πελάτης επικοινωνούν, περνώντας πληροφορίες μεθόδων, ορίσματα και επιστρέφουν τιμές ο ένας στον άλλο.

4.2 Security And Trust Services API

Το Security And Trust Services API SATSA ορίζεται στο JSR (Java Specification Request) 177 και καθορίζει ένα προαιρετικό πακέτο το οποίο παρέχει ένα ασφαλές και έμπιστο API για J2ME. Αυτό το API παρέχει πρόσβαση σε υπηρεσίες που παρέχονται από ένα στοιχείο ασφάλειας (όπως μία έξυπνη κάρτα), συμπεριλαμβάνοντας ασφαλή αποθήκευση και ανάκτηση των ευαίσθητων πληροφοριών, καθώς επίσης και υπηρεσίες κρυπτογράφησης και αυθεντικοποίησης [3].

Το SATSA προωθεί το Generic Connection Framework (GCF) ορισμένο στην έκδοση 1.0 του Connected Limited Device Configuration (CLDC), να παρέχει ένα περισσότερο αφηρημένο interfacesε σχέση με το βασικό μοντέλο διαβίβασης μηνυμάτων και το JCRMI.. Για να υποστηρίζει το SATSA τις αποστολές APDU μηνυμάτων, ορίζει στην κλάση javax.microedition.apdu το interface APDUConnection ενώ για την υποστήριξη του JCRMI ορίζει στην κλάση javax.microedition.jcrmi το interface JavaCardRMIConnection [4].

Το SATSA είναι και ο μόνος τρόπος επικοινωνίας μιας εφαρμογής γραμμένης σε J2ME απευθείας με μία έξυπνη κάρτα που χρησιμοποιεί Java Card.

4.3 OpenCard Framework

Το OpenCard Framework είναι ένα στάνταρ framework που ανακοινώθηκε από ένα βιομηχανικό consortium , μεγάλων εταιριών που δραστηριοποιούνται στις έξυπνες κάρτες και παρέχει δια-λειτουργικότητα μεταξύ πολλών hardware and software πλατφόρμων. Το OpenCard Framework είναι ένα ανοιχτό στάνταρ που προσφέρει μία αρχιτεκτονική και ένα σύνολο από APIs τα οποία διευκολύνουν αυτούς που αναπτύσσουν εφαρμογές και τους παρόχους υπηρεσιών να χτίζουν και να διανέμουν

λύσεις βασισμένες σε έξυπνες κάρτες και πιο συγκεκριμένα τις τερματικές εφαρμογές που είναι γραμμένες σε Java, σε οποιοδήποτε OpenCard συμβατό περιβάλλον [5][6].

Τα μέλη του consortium τα οποία ανέπτυξαν το framework είναι τα ακόλουθα :

- 3-G International
- American Express Travel Related Services
- Bull
- First Access
- Gemplus
- Giesecke & Devrient
- IBM
- Toshiba Corporation
- TOWITOKO
- Schlumberger
- Siemens
- Sun Microsystems
- UbiQ Inc.
- Visa International
- XAC Automation

Είναι φανερό πως το opencard framework είναι πλέον ένα στάνταρ για Java Cards αφού στο consortium συμμετέχει η SUN καθώς και κάποιες από τις μεγαλύτερες εταιρείες κατασκευής έξυπνων καρτών γι' αυτό και θα το αναλύσουμε σε μεγαλύτερο βάθος στις παραγράφους που ακολουθούν.

Το OpenCard Framework είναι ανεξάρτητο του λειτουργικού συστήματος που χρησιμοποιείται αφού είναι υλοποιημένο σε Java. Υποστηρίζει κάρτες οι οποίες περιέχουν πολλαπλές εφαρμογές, όπως είναι και οι Java Cards και οι πλατφόρμες που στοχεύει είναι δικτυακοί υπολογιστές, web browsers ή οποιαδήποτε άλλη πλατφόρμα η οποία τρέχει Java και πρέπει να αλληλεπιδράσει με έξυπνες κάρτες. Τελικά το OCF είναι Java στο τερματικό του χρήστη που μιλάει με μία έξυπνη κάρτα και οι Java εφαρμογές που τρέχουν σε έναν προσωπικό υπολογιστή μπορούν να

χρησιμοποιήσουν το OCF για να έχουν πρόσβαση σε έξυπνες κάρτες ανεπτυγμένες με την Java Card τεχνολογία.

4.3.1 Στόχοι του Opencard Framework

Κοιτώντας τις εφαρμογές έξυπνων καρτών από την οπτική γωνία του προγραμματιστή εφαρμογών, φαίνονται τρία μέρη που παίζουν σημαντικό ρόλο και αυτά είναι τα ακόλουθα :

Κατασκευαστές αναγνωστών καρτών

Οι κατασκευαστές αναγνωστών καρτών παρέχουν τους αναγνώστες καρτών ή αλλιώς τις συσκευές υποδοχής καρτών ή ακόμα στην ορολογία του OCF τα τερματικά καρτών. Κάθε κατασκευαστής, προσφέρει ένα μικρό ή μεγάλο φάσμα από αναγνώστες καρτών, οι οποίοι μπορεί να είναι από πολύ απλοί έως πολύ σύνθετοι (ενσωματωμένο πληκτρολόγιο ή οθόνη). Δυστυχώς, οι κατασκευαστές δεν έχουν συμφωνήσει σε ένα κοινό στάνταρ interface και πολλά διαφορετικά πρωτόκολλα επιτρέπονται.

Παροχείς λειτουργικών συστημάτων καρτών

Υπάρχουν επίσης πολυάριθμες ανταγωνιστικές εταιρίες που προσφέρουν πολλά διαφορετικά λειτουργικά συστήματα για κάρτες και διαφορετικά APIs. Αυτό έχει σαν αποτέλεσμα μεγάλη ποικιλία από κώδικες εντολών και απαντήσεων.

Εκδότες καρτών

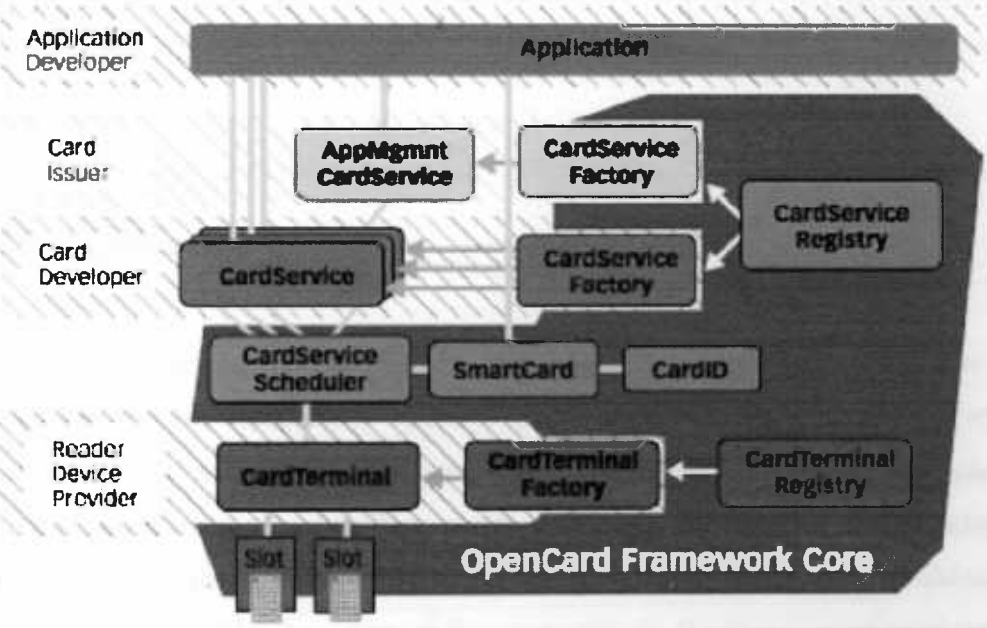
Υπάρχουν ακόμα οι οντότητες που εκδίδουν έξυπνες κάρτες στους πελάτες και είναι εκείνες που αποφασίζουν που θα τοποθετηθούν οι εφαρμογές στις κάρτες που εκδίδουν. Έτσι , η τοποθεσία του κώδικα μπορεί να ποικίλει ανάλογα.

Ο στόχος είναι να μειωθεί η εξάρτηση μεταξύ αυτών των μερών. Αν επιτευχθεί αυτό, ούτε ο τύπος του αναγνώστη, ούτε η μάρκα της έξυπνης κάρτας θα έχει κάποια σημασία. Έτσι λοιπόν οι στόχοι του OCF αφορούν την ανεξαρτησία του κατασκευαστή αναγνωστών καρτών, την ανεξαρτησία των παρόχων λειτουργικών

συστημάτων και την ανεξαρτησία των εκδοτών καρτών. Επιπροσθέτως , στόχος είναι το OCF να είναι εύκολο στην χρήση και επεκτάσιμο.

4.3.2 Αρχιτεκτονική του Open Card Framework

Τα τρία βασικότερα στοιχεία της αρχιτεκτονικής του OCF, παρουσιάζονται στο σχήμα 4.3.2.1, και προσπαθούν να πετύχουν καθένα ξεχωριστά τους στόχους που τέθηκαν στην προηγούμενη παράγραφο. Αυτά είναι τα CardService, AppletAccessCardService και CardTerminal.



Σχήμα 4.3.2.1 – Η αρχιτεκτονική του OCF

4.3.2.1 CardTerminal

Οι κατασκευαστές αναγνωστών καρτών που θέλουν να κάνουν τους δικούς τους αναγνώστες διαθέσιμους στις εφαρμογές που χρησιμοποιούν το OCF , πρέπει να παρέχουν μία κλάση CardTerminal η οποία περιλαμβάνει την συμπεριφορά του αναγνώστη και μία CardTerminalFactory κλάση. Η CardTerminalFactory χρησιμοποιείται από το OCF για να δημιουργήσει στιγμιότυπα της CardTerminal

όταν το Framework αρχικοποιηθεί. Το CardTerminalFactory αναγνώστη που είναι συνδεδεμένος σε έναν υπολογιστή πρέπει να καταχωρηθεί στο CardTerminalRegistry.

4.3.2.2 AppletAccessCardService

Με την εισαγωγή της έννοιας των πολλαπλών εφαρμογών στην ίδια έξυπνη κάρτα, νέες εξαρτήσεις , όπως ποιες εφαρμογές είναι διαθέσιμες στην κάρτα ή που βρίσκεται η φυσική θέση μιας εφαρμογής και των δεδομένων της, δημιουργήθηκαν. Αυτός είναι και ο ρόλος του AppletAccessCardService που είναι ικανή να εντοπίζει και αν επιλέγει εφαρμογές που βρίσκονται σε μία κάρτα, να επιστρέφει τις εφαρμογές που μία συγκεκριμένη κάρτα υποστηρίζει και να εγκαθιστά και να απεγκαθιστά εφαρμογές.

4.3.2.3 CardService

Το CardService επίπεδο του OCF υλοποιεί ένα στάνταρτ interface για χρήση από τον προγραμματιστή της εφαρμογής κρύβοντας τις ιδιαιτερότητες της κάθε έξυπνης κάρτας. Μία CardService προσφέρει μία συγκεκριμένη λειτουργικότητα μιας κάρτας σε αυτόν που αναπτύσσει μία τερματική εφαρμογή. Η CardService δημιουργεί APDUs και επικοινωνεί με την κάρτα για να υλοποιήσει λειτουργίες υψηλότερου επιπέδου. Για κάθε applet θα πρέπει να δημιουργηθεί μία CardService για αυτό καθώς επίσης και ένα CardServiceFactory για το OCF. Θα πρέπει να υπάρχει αντιστοιχία ένα προς ένα μεταξύ του applet και του OCF CardService.

Το OCF προσφέρει μεθόδους και κλάσεις για CardService για την πρόσβαση στην κάρτα. Ένα CardServiceFactory είναι συνδεδεμένο με κάθε υλοποίηση CardService και είναι ικανό να το κατασκευάζει. Το CardServiceFactory αναγνωρίζει την κάρτα ή τις κάρτες για τις οποίες το CardService κατασκευάστηκε. Όταν μία έξυπνη κάρτα εισάγεται στον αναγνώστη, το OCF κοιτάζει στην λίστα του από τις καταχωρημένα CardServiceFactory και αρχικοποιεί τα card services που αντιστοιχούν στην κάρτα.

ΚΕΦΑΛΑΙΟ 5

ΣΧΕΔΙΑΣΜΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΠΟΡΤΟΦΟΛΙΟΥ ΜΕ ΤΕΧΝΟΛΟΓΙΑ JAVA CARD

Κεφάλαιο

5. ΣΧΕΔΙΑΣΜΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΠΟΡΤΟΦΟΛΙΟΥ ΜΕ ΤΕΧΝΟΛΟΓΙΑ JAVA CARD

Στο κεφάλαιο αυτό θα αναφέρουμε τα βήματα προκειμένου να αναπτύξουμε το Java Card applet για ένα ηλεκτρονικό πορτοφόλι, το οποίο θα έχει την δυνατότητα να αυθεντικοποιεί τον χρήστη μέσω επαλήθευσης PIN, να ανανεώνει το PIN με κάποιο νέο που εισάγει ο κάτοχος, να προσθέτει κάποιο ποσό, να αφαιρεί κάποιο ποσό και τέλος να επιστρέφει το διαθέσιμο υπόλοιπο.

5.1 Ανάλυση και Σχεδιασμός

Όπως σε κάθε άλλη περίπτωση ανάπτυξης λογισμικού εφαρμογών έτσι και για την ανάπτυξη ενός Java Card applet , θα πρέπει πριν ξεκινήσουμε την συγγραφή του κώδικα, να προηγηθεί μια φάση σχεδιασμού όπου θα καθορίσουμε την αρχιτεκτονική του applet. Η φάση σχεδιασμού θα μπορούσε να αποτελείται από τρία βασικά βήματα:

1. Καθορισμός των λειτουργιών του applet.

2. Απόδοση κάποιου AID στο applet αλλά και στο πακέτο (package) που το περιέχει.

3. Καθορισμός του interface μεταξύ του applet και της τερματικής εφαρμογής.

5.1.1 Καθορισμός Λειτουργιών applet

Όπως προαναφέραμε, στόχος είναι η κατασκευή ενός πορτοφολιού το οποίο θα πρέπει να αποθηκεύει ηλεκτρονικό χρήμα και να υποστηρίζει λειτουργίες χρέωσης, πίστωσης και ερώτησης του διαθέσιμου υπολοίπου.

Για την αποφυγή χρησιμοποίησης της κάρτας από μη εξουσιοδοτημένα άτομα, θα πρέπει να περιλαμβάνεται ένας αλγόριθμος ασφάλειας. Ο αλγόριθμος αυτός θα δέχεται από τον χρήστη ένα PIN (personal identification number), 4 ψηφίων. Ο χρήστης της κάρτας θα πληκτρολογεί τον προσωπικό του PIN από το πληκτρολόγιο του CAD. Ο αλγόριθμος που θα χρησιμοποιηθεί κλειδώνει την κάρτα αν ο χρήστης πληκτρολογήσει τρεις φορές λάθος το PIN του. Το default PIN της κάρτας ορίζεται το 0000. Είναι αυτονόητο ότι για να πραγματοποιηθεί μια οποιαδήποτε πράξη πίστωσης ή χρέωσης, θα πρέπει πρώτα να έχει γίνει επαλήθευση του PIN του χρήστη.

Επίσης για να κάνουμε πιο λειτουργική την εφαρμογή, τοποθετούμε και μία λειτουργία ανανέωσης του PIN, δηλαδή αντικατάστασης του παλιού με ένα νέο PIN επιλογής του χρήστη. Και σε αύτή την περίπτωση εννοείται ότι πρώτα θα πρέπει να έχει γίνει επαλήθευση του PIN του χρήστη.

Τελικά συνοψίζοντας, βλέπουμε ότι χρειαζόμαστε το applet μας να πραγματοποιεί πέντε διαδικασίες : επαλήθευση κωδικού, ανανέωση κωδικού, πίστωση, χρέωση, ερώτηση υπολοίπου.

5.1.2 Καθορισμός AID

Οι περισσότερες παραδοσιακές εφαρμογές, αναγνωρίζονται συνήθως από ένα όνομα. Στην Java Card τεχνολογία ένα applet αναγνωρίζεται και επιλέγεται για χρήση από ένα AID (Application ID), στο οποίο αναφερθήκαμε όταν μιλήσαμε για τον κύκλο ζωής του Java Card Applet στο κεφάλαιο 2. Επίσης ένα AID χρησιμοποιείται για την αναγνώριση κάθε Java πακέτου. Ένα AID μπορεί να έχει μήκος από 5 έως 16 bytes, το οποίο αποτελείται από ένα μέρος που αντιστοιχεί σε αναγνωριστικό του πάροχου της εφαρμογής και ένα αναγνωριστικό της εφαρμογής του πάροχου και αυτό φαίνεται στο σχήμα 5.1.2.1.

AID	
RID (National registered application provider)	**PIX (Proprietary application identifier extension)**
5 bytes	**0 έως 11 bytes**

5.1.2.1 - Η μορφή του AID

Ο ISO ελέγχει και είναι υπεύθυνος για την απονομή των RID's σε εταιρίες , η οποία έχει το δικό της , μοναδικό RID, ενώ κάθε εταιρία διαχειρίζεται μόνη της τα PIX's του AID.

Για την ανάπτυξη του ηλεκτρονικού πορτοφολιού εμείς θα χρησιμοποιήσουμε μια τυχαία AID , αφού προορίζεται για πειραματική χρήση. Σε αντίθετη περίπτωση θα έπρεπε να μας είχε εκχωρηθεί ένα RID και βάση αυτού να δημιουργούσαμε την AID. Το AID που θα χρησιμοποιήσουμε για το πακέτο είναι το 0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x2 ενώ το AID για το applet είναι το 0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1 . Παρατηρούμε ότι τα πρώτα 5 bytes (RID) είναι τα ίδια για το πακέτο και για το applet ενώ διαφέρουν μόνο στο τελευταίο byte.

5.1.3 Ορισμός του interface μεταξύ του Applet και της τερματικής εφαρμογής

Ο ορισμός του interface μεταξύ του Applet και της τερματικής εφαρμογής, στη ουσία αποτελεί τον τρόπο που το applet μιλάει με τον έξω κόσμο. Όπως έχουμε αναφέρει με λεπτομέρεια σε προηγούμενο κεφάλαιο, οποιαδήποτε τερματική εφαρμογή επικοινωνεί με μία Java Card με την βοήθεια APDU's . Κάθε APDU είναι ένα ζεύγος εντολής και απάντησης. Η κάρτα έχει παθητικό ρόλο περιμένοντας για κάποια command APDU και όταν την λάβει απαντάει με μία response APDU.

Για κάθε μία λειτουργία που ορίσαμε στην παράγραφο 5.1.1 θα πρέπει να ορίσουμε και μία APDU, μέσω της οποίας θα επικοινωνεί η εκάστοτε τερματική εφαρμογή με το applet στην Java Card καθώς και τις απαντήσεις που θα στέλνει η κάρτα στην εφαρμογή. Στο σχήμα 5.1.3.1 παρουσιάζονται αναλυτικά οι APDU's [7].

VERIFY APDU command						
Command APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x20	0x0	0x0	Μήκος του PIN	PIN data	N/A

- CLA byte προσδιορίζει τη δομή της APDU
- INS byte (0x20) προσδιορίζει την λειτουργία Verify
- P1 and P2 δεν χρησιμοποιούνται
- To data field παρίεχει την τιμή του PIN

Response APDU		
Optional data	**Status word**	**Περιγραφή**
N/A	0x9000	Successful processing
	0x6300	Verification failed

UPDATE PIN APDU command						
Command APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0x0	0x25	0x0	0x0	Μήκος του PIN	PIN data	N/A
• Το data field παρίεχει την τιμή του νέου PIN						
Response APDU						
Optional data		**Status word**		**Περιγραφή**		
No data		0x9000		Successful processing		
		0x6301		PIN verification required		
		0x6A86		Invalid PIN length		

CREDIT APDU command						
Command APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x30	0x0	0x0	1	Credit amount	N/A
• Το data field περιλαμβάνει το ποσό της πίστωσης						
Response APDU						
Optional data		**Status word**		**Περιγραφή**		
N/A		0x9000		Successful processing		
		0x6301		PIN verification required		
		0x6A83		Invalid credit amount		
		0x6A84		Exceed the maximum amount		

DEBIT APDU command						
Command APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x40	0x0	0x0	1	Debit amount	N/A

- Το data field περιλαμβάνει το ποσό της χρέωσης

Response APDU		
Optional data	**Status word**	**Περιγραφή**
N/A	0x9000	Successful processing
	0x6301	PIN verification required
	0x6A83	Invalid debit amount
	0x6A85	Negative balance

GET BALANCE APDU command						
Command APDU						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x50	0x0	0x0	N/A	N/A	2

- Η απάντηση θα περιλαμβάνει το διαθέσιμο υπόλοιπο

Response APDU		
Optional data	**Status word**	**Περιγραφή**
N/A	0x9000	Successful processing

Σχήμα 5.1.3.1 – Οι APDU's των λειτουργιών της κάρτας

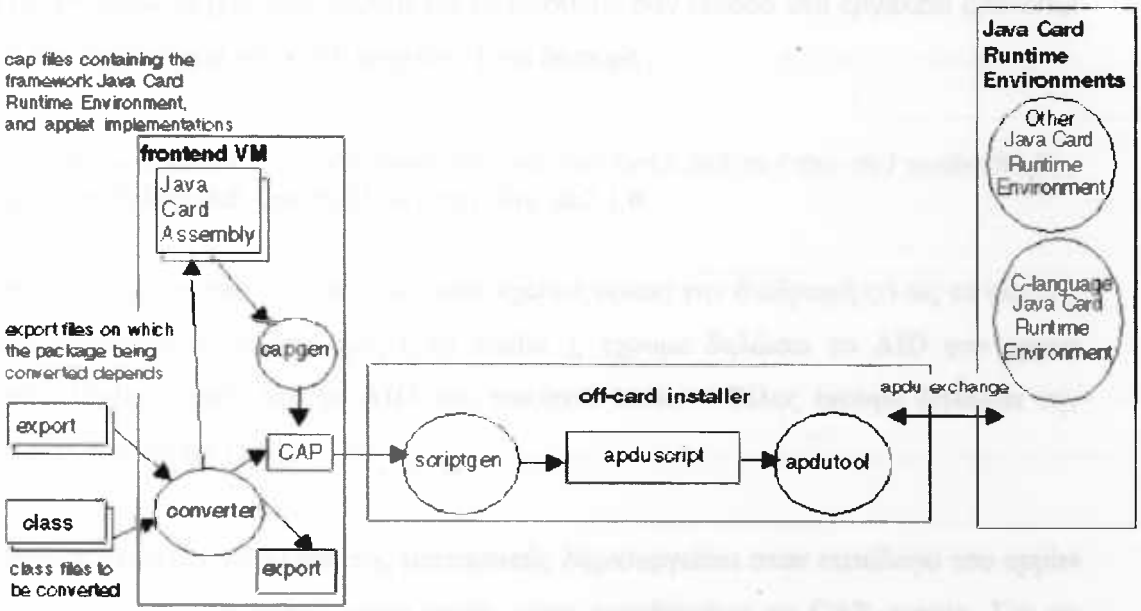
Επιπροσθέτως των status words που θα δημιουργηθούν από εμάς για τις ανάγκες του συγκεκριμένου applet, το interface javacard.framework.ISO7816 παρέχει επιπλέον status words για συνηθισμένα λάθη τα οποία προκύπτουν συχνά, όπως για παράδειγμα ένα APDU command λάθος σύνταξης κ.α.

5.2 Υλοποίηση

Για την υλοποίηση της Java Card του ηλεκτρονικού πορτοφολιού χρησιμοποιήθηκε το Java Card Development Kit 2.2.1 το οποίο διατίθεται δωρεάν από τη SUN. Το JCDK παρέχει και μία σειρά από εργαλεία που μας επιτρέπουν την ανάπτυξη applets που βασίζονται στο Java Card API. Για να λειτουργήσει το JCDK απαιτείται να είναι εγκατεστημένο στο σύστημα το J2SE SDK. Αφού λοιπόν εγκαταστήσουμε το JCDK και γράψουμε τον κώδικα για το applet μας πρέπει να ακολουθήσουμε μία σειρά βημάτων για να το κάνουμε λειτουργικό.

5.2.1 Εγκατάσταση ενός Java Card applet

Στο σχήμα 5.2.1.1 βλέπουμε τη ροή των δεδομένων ώστε να καταλήξουμε σε ένα applet που μπορεί να εγκατασταθεί και λειτουργήσει σε μία πραγματική Java Card. Αρχικά το Java source (πηγαίος κώδικας), γίνεται compiled και περνάει σαν είσοδος στον Converter. Ο Converter , μετατρέπει κλάσεις που περιλαμβάνουν κάποιο Java πακέτο σε ένα converted applet (CAP) είτε σε ένα Java Card Assembly αρχείο. Ένα CAP αρχείο είναι μία δυαδική αναπαράσταση ενός converted Java πακέτου ενώ το Java Card Assembly είναι ένα αναγνώσιμο text αρχείο που βοηθά στην αποσφαλμάτωση και τον έλεγχο. Ένα Java Card Assembly αρχείο μπορεί να μπει σαν είσοδος στο εργαλείο capgen για να παραχθεί το CAP αρχείο [9].



5.2.1.1 – Βήματα για την δημιουργία ενός Java Card applet

Το CAP αρχείο, δέχεται επεξεργασία από έναν off-card installer , το εργαλείο scriptgen το οποίο παράγει ένα APDU script αρχείο το οποίο με τη σειρά του αποτελεί είσοδο στο εργαλείο apdutool. Το εργαλείο apdutool είναι πολύ χρήσιμο πέρα από αυτή την φάση της εγκατάστασης ενός applet στην κάρτα αφού στέλνει APDU's τα οποία βρίσκονται σε ένα text αρχείο στην εκάστοτε υλοποίηση του Java Card Runtime Environment. Όπως εξετάσαμε και σε προηγούμενο κεφάλαιο, μια υλοποίηση JCRE περιλαμβάνει μία Java Card Virtual Machine, το Java Card API και άλλες υπηρεσίες υποστήριξης. Το Java Card Development Kit 2.2.1 το συνοδεύει ένα JCRE (Cref) γραμμένο σε C το οποίο θα χρησιμοποιήσουμε και εμείς για να προσομοιώσουμε το περιβάλλον μιας πραγματικής κάρτας.

5.2.2 Εγκατάσταση του wallet applet

Ας δούμε όμως πως εφαρμόσαμε τα παραπάνω βήματα στην περίπτωση του ηλεκτρονικού πορτοφολιού. Αφού δημιουργήσαμε το Wallet.java το οποίο περιέχει τον πηγαίο κώδικα σε Java για το applet μας και το οποίο θα δούμε με λεπτομέρεια στην παράγραφο της συντήρησης, το μεταγλωττίσαμε με τη βοήθεια του JDK1.3 μιας και οι πιο καινούργιες εκδόσεις της J2SE παρουσίασαν πρόβλημα στη συνέχεια.

Έτσι το .class αρχείο που πήραμε θα το δώσουμε σαν είσοδο στο εργαλείο converter για την δημιουργία του CAP αρχείου. Έτσι δίνουμε :

converter -classdir c:\ -applet 0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x1 wallet.Wallet wallet 0xa0:0x0:0x0:0x0:0x0:0x62:0x3:0x1:0xc:0x2 1.0

Στην παραπάνω σύνταξη του converter έχουμε ορίσει την διαδρομή c:\ ως το φάκελο που βρίσκεται το πακέτο μας (το wallet), έχουμε δηλώσει το AID του applet wallet.Wallet καθώς και το AID του πακέτου wallet . Τέλος έχουμε δηλώσει την έκδοση του applet (1.0) .

Μετά την επιτυχή εκτέλεση της μετατροπής δημιουργείται στον κατάλογο του applet ένας κατάλογος 'javacard' στον οποίο μέσα τοποθετείται το CAP αρχείο. Για να πάρουμε από το wallet.cap το script αρχείο wallet.scr με τις APDU εντολές δίνουμε :

scriptgen wallet.cap –o wallet.scr

Έτσι πριν χρησιμοποιήσουμε το apdutool για να στείλουμε το wallet.scr στην κάρτα, πρέπει να προσθέσουμε στο wallet.scr τέσσερις έντολές, μία για να θέσουμε σε λειτουργία την κάρτα, μία για να δηλώσουμε τον τερματισμό, μία για να επιλέξουμε των on-card installer και μία για να δημιουργήσουμε ένα στιγμιότυπο του applet. Έτσι το wallet.scr θα διαμορφωθεί ως εξής :

powerup;

// Installer Applet
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;

//CAP begin

……

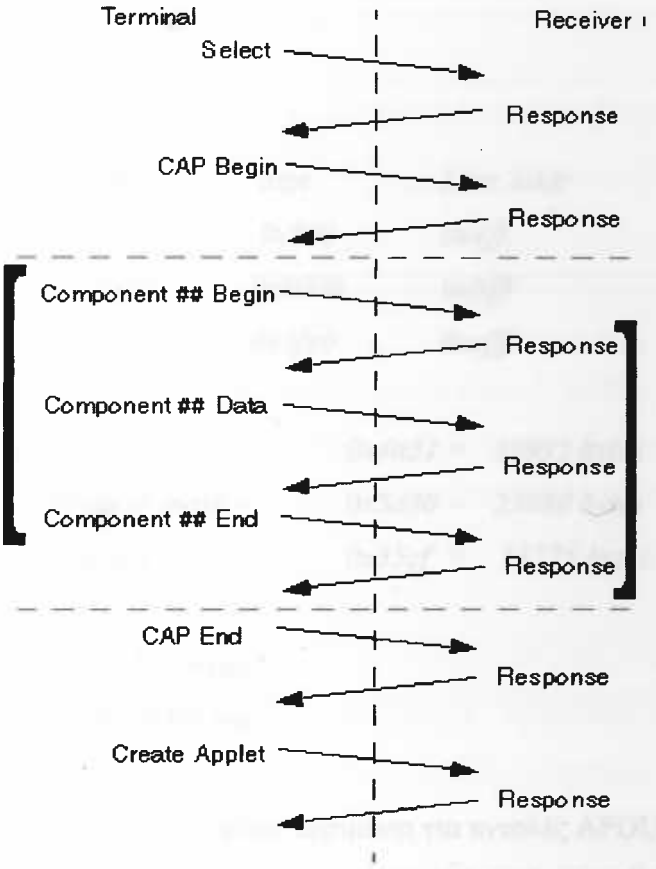
//CAP end

//Create instance

0x80 0xB8 0x00 0x00 0x0A 0x09 0xa0 0x0 0x0 0x0 0x62 0x3 0x1 0xc 0x1 0x7F;

powerdown;

Τα βήματα της εγκατάστασης του applet από την αποστολή του script αρχείου, φαίνονται στο σχήμα 5.2.2.1.



5.2.2.1 – Ανταλλαγή μηνυμάτων για την εγκατάσταση του applet

Τώρα πια είμαστε έτοιμοι να στείλουμε το wallet.scr με το apdutool στην κάρτα. Εμείς όπως προαναφέραμε θα χρησιμοποιήσουμε το cref το οποίο προσομοιώνει το περιβάλλον μιας Java Card που βρίσκεται μέσα σε κάποιον Card reader. Σημαντικό στοιχείο του cref είναι ότι μπορεί να κρατάει σε ένα αρχείο και να ανακτήσει από αυτό την κατάσταση που βρίσκεται μια κάρτα μια δεδομένη στιγμή δηλαδή να κρατάει τα περιεχόμενα της EEPROM της κάρτας. Έτσι λοιπόν σε ένα command prompt τρέχουμε το cref :

cref -o image

και το cref μας ενημερώνει για την διαθέσιμη μνήμη της κάρτας και ότι τα περιεχόμενα της EEPROM θα αποθηκευτούν στο αρχείο image :

Java Card 2.2 C Reference Implementation Simulator (version 0.41)
Copyright 2002 Sun Microsystems, Inc. All rights reserved.

Memory configuration

Type	*Base*	*Size*	*Max Addr*
RAM	*0x0*	*0x500*	*0x4ff*
ROM	*0x1000*	*0x8000*	*0x8fff*
E2P	*0x9020*	*0x3fe0*	*0xcfff*

ROM Mask size =	*0x4a31 =*	*18993 bytes*
Highest ROM address in mask =	*0x5a30 =*	*23088 bytes*
Space available in ROM =	*0x35cf =*	*13775 bytes*

EEPROM will be saved in file "image"
Mask has now been initialized for use

Το cref τώρα μέσω ενός socket interface περιμένει για εντολές APDU εντολές. Έτσι σε κάποιο άλλο command prompt στέλνουμε με το apdutool το wallet.scr :

apdutool wallet.scr

Το apdutool μας ενημερώνει αν η διαδικασία ήταν επιτυχής για κάθε apdu εντολή που στάλθηκε μέσω των απαντήσεων που λαμβάνει από το cref και τέλος το cref και το apdutool τερματίζουν τη λειτουργία τους. Τώρα πια δίνοντας :

cref –i image

έχουμε την Java Card του ηλεκτρονικού μας πορτοφολιού την οποία μπορούμε να τη διαχειριστούμε στέλνοντας APDU εντολές που αντιστοιχούν στις λειτουργίες που καθορίσαμε στο applet μας, λειτουργίες δηλαδή επαλήθευσης κωδικού, ανανέωσης κωδικού, πίστωσης, χρέωσης, ερώτησης υπολοίπου. Πέρα από το apdutool το οποίο μπορούμε να χρησιμοποιήσουμε για τον έλεγχο του applet μας, θα πρέπει να φτιάξουμε κάποια τερματική εφαρμογή η οποία να διαχειρίζεται και να επικοινωνεί με την κάρτα. Δύο σενάρια τέτοιων εφαρμογών εξετάζουμε και υλοποιούμε στα κεφάλαια 6 και 7 .

5.3 Έλεγχος Λειτουργίας

Στην παράγραφο αυτή θα ελέγξουμε την λειτουργία της Java Card που κατασκευάσαμε στέλνοντας APDU εντολές με το εργαλείο apdutool και εξετάζοντας τις απαντήσεις που θα στείλει η κάρτα. Όπως αναφέραμε και όταν μελετήσαμε τον κύκλο ζωής του applet πριν αποστείλουμε σε αυτό κάποια APDU θα πρέπει να το επιλέξουμε , με τη λογική ότι σε μια κάρτα μπορούν να υπάρχουν περισσότερα του ενός applets και θα πρέπει το JCRE να γνωρίζει σε ποιο από όλα απευθύνονται οι APDU εντολές που στέλνουμε.

Ας υποθέσουμε λοιπόν ότι θέλουμε να προσθέσουμε 10 χρηματικές μονάδες ,να αφαιρέσουμε 2 και να πάρουμε το διαθέσιμο υπόλοιπο. Θα πρέπει να δημιουργήσουμε ένα αρχείο εντολών APDU το οποίο καταρχάς να περιέχει την APDU εντολή επιλογής του wallet applet , έπειτα επαλήθευση του κωδικού PIN ώστε να επιτραπούν οι δοσοληψίες, ερώτηση υπολοίπου πριν τις συναλλαγές πίστωση 10 μονάδων , χρέωση 2 μονάδων και τέλος ερώτηση του διαθέσιμου υπολοίπου. Ένα τέτοιο αρχείο ας το ονομάσουμε apdu.scr και η μορφή του θα ήταν η ακόλουθη :

```
powerup;

//Select wallet applet
0x00 0xA4 0x04 0x00 0x09 0xa0 0x0 0x0 0x0 0x62 0x3 0x1 0xc 0x1 0x7F;

//Verify PIN APDU
```

0xB0 0x20 0x00 0x00 0x03 0x00 0x00 0x00 0x00 0x7F;

//Get Balance
0xB0 0x50 0x00 0x00 0x00 0x7F;

//Credit 10
0xB0 0x30 0x00 0x00 0x1 0xa 0x7F;

//Debit 2
0xB0 0x40 0x00 0x00 0x1 0x2 0x7F;

//Get Balance
0xB0 0x50 0x00 0x00 0x00 0x7F;

powerdown;

Τρέχοντας το cref σε ένα command prompt στέλνοντας το apdu.scr με το apdutool από ένα άλλο command prompt θα πάρουμε τα ακόλουθα :

c:\wallet\javacard>apdutool apdu.scr
Java Card 2.2 ApduTool (version 0.20)
Copyright 2002 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
Connected.
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 0c, 01, Le: 00, SW1: 90, SW2: 00
CLA: b0, INS: 20, P1: 00, P2: 00, Lc: 04, 00, 00, 00, 00 Le: 00, SW1: 90, SW2: 00
CLA: b0, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 02, 00, 00, SW1: 90, SW2: 00
CLA: b0, INS: 30, P1: 00, P2: 00, Lc: 01, 0a, Le: 00, SW1: 90, SW2: 00
CLA: b0, INS: 40, P1: 00, P2: 00, Lc: 01, 02, Le: 00, SW1: 90, SW2: 00
CLA: b0, INS: 50, P1: 00, P2: 00, Lc: 00, Le: 02, 00, 08, SW1: 90, SW2: 00

Το apdutool μας ενημερώνει ότι συνδέθηκε με την κάρτα στην πόρτα 9025 (στην συγκεκριμένη περίπτωση με το cref μέσω sockets που ακούει στην πόρτα 9025). Στη συνέχεια το cref στέλνει μία ATR (Answer To Reset), η οποία διαφέρει για κάθε τύπο κάρτας και κατ' επέκταση και αναγνώστη καρτών. Οι υπόλοιπες γραμμές αφορούν τις APDU που στείλαμε και την απάντηση που έλαβε ο apdutool. Παρατηρούμε ότι όλες έχουν στην status word της απάντησης το 9000 που σημαίνει ότι η εντολή που στάλθηκε ικανοποιήθηκε με επιτυχία. Η πρώτη εντολή αφορούσε την επιλογή του applet με τη βοήθεια του AID του, η δεύτερη την επαλήθευση του PIN (0000), η Τρίτη την ερώτηση υπολοίπου (διαθέσιμο υπόλοιπο 0) , η τέταρτη και η πέμπτη την προσθήκη 10μονάδων ('a' στο δεκαεξαδικό σύστημα που χρησιμοποιείται) και την χρέωση 2 και τέλος πάλι ερώτηση υπολοίπου του οποίου η απάντηση έδωσε σωστά 8 μονάδες.

5.4 Συντήρηση

Στην παράγραφο αυτή θα αναλύσουμε κάποια σημαντικά κομμάτια κώδικα του Wallet applet, του σημαντικότερου τμήματος της κάρτας.

5.4.1 Δήλωση Σταθερών

```
// code of CLA byte in the command APDU header
final static byte Wallet_CLA =(byte)0xB0;

// codes of INS byte in the command APDU header
final static byte VERIFY = (byte) 0x20;
final static byte UPDATE_PIN = (byte) 0x25;
final static byte CREDIT = (byte) 0x30;
final static byte DEBIT = (byte) 0x40;
final static byte GET_BALANCE = (byte) 0x50;

// maximum balance
final static short MAX_BALANCE = 0x7FFF;

// maximum transaction amount
final static byte MAX_TRANSACTION_AMOUNT = 127;

// maximum number of incorrect tries before the
// PIN is blocked
final static byte PIN_TRY_LIMIT =(byte)0x03;

// maximum size PIN
final static byte MAX_PIN_SIZE =(byte)0x08;

// signal that the PIN verification failed
final static short SW_VERIFICATION_FAILED =0x6300;

// signal the the PIN validation is required
// for a credit or a debit transaction
final static short SW_PIN_VERIFICATION_REQUIRED =0x6301;
```

```
// signal invalid transaction amount
// amount > MAX_TRANSACTION_AMOUNT or amount < 0
final static short SW_INVALID_TRANSACTION_AMOUNT = 0x6A83;

// signal that the balance exceed the maximum
final static short SW_EXCEED_MAXIMUM_BALANCE =0x6A84;

// signal the the balance becomes negative
final static short SW_NEGATIVE_BALANCE = 0x6A85;

// signal invalid pin length
final static short SW_INVALID_PIN_LENGTH = 0x6A86;
```

Σημαντικό κομμάτι του κώδικα είναι αυτό της δήλωσης σταθερών που χωρίς να είναι υποχρεωτικό διευκολύνει πολύ την συγγραφή και την ανάγνωση του κώδικα αφού είναι πολύ ευκολότερο να χρησιμοποιούμε περιγραφικά ονόματα από δεκαεξαδικούς αριθμούς. Αρχικά δηλώνονται οι τιμές που θα παίρνουν στο πεδίο INS οι διάφορες μέθοδοι του applet καθώς και η τιμή του πεδίου CLA που θα είναι κοινή για όλες τις APDU του wallet applet. Έπειτα δηλώνουμε κάποιες σταθερές με κάποια ονόματα και τέλος κάποιες απαντήσεις που θα στέλνει η κάρτα, με APDU απαντήσεις, σε περιπτώσεις λάθους.

5.4.2 Μέθοδος debit

```
private void debit(APDU apdu) {

    // access authentication
    if ( ! pin.isValidated() )
       ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);

    byte[] buffer = apdu.getBuffer();

    byte numBytes = (byte)(buffer[ISO7816.OFFSET_LC]);

    byte byteRead = (byte)(apdu.setIncomingAndReceive());

    if ( ( numBytes != 1 ) || (byteRead != 1) )
       ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

    // get debit amount
    byte debitAmount = buffer[ISO7816.OFFSET_CDATA];

    // check debit amount
    if ( ( debitAmount > MAX_TRANSACTION_AMOUNT) ||  ( debitAmount < 0 ) )
       ISOException.throwIt(SW_INVALID_TRANSACTION_AMOUNT);

    // check the new balance
    if ( (short)( balance - debitAmount ) < (short)0 )
         ISOException.throwIt(SW_NEGATIVE_BALANCE);

    balance = (short) (balance - debitAmount);

  } // end of debit method
```

Η μέθοδος debit είναι ίσως το πιο χαρακτηριστικό παράδειγμα μεθόδου της κάρτας αφού δέχεται δεδομένα, τα επεξεργάζεται και αν υπάρξει πρόβλημα στέλνει ανάλογη απάντηση. Αρχικά γίνεται έλεγχος για το αν ο χρήστης έχει επαληθεύσει τον κωδικό PIN και εφόσον το έχει κάνει (υπάρχει η μέθοδος isValidated() που παίρνει τιμές true/false) αποθηκεύεται σε ένα buffer η εισερχόμενη APDU εντολή. Από εδώ και στο εξής χρησιμοποιούνται σταθερές ISO7816 για την μετακίνηση στα περιεχόμενα του buffer. Για παράδειγμα η πρώτη σταθερά `ISO7816.OFFSET_LC` μας μεταφέρει στο πεδίο Lc που περιέχει τον αριθμό των bytes των δεδομένων που στάλθηκαν με την συγκεκριμένη APDU. Αφού ελεγχθεί και το κατά πόσο τα bytes που διαβάστηκαν ήταν τόσα όσα τα αναμενόμενα, το ποσό της χρέωσης εκχωρείται στην μεταβλητή debitAmount ώστε να χρησιμοποιηθεί για περαιτέρω ελέγχους. Βλέπουμε για παράδειγμα ότι αν το υπόλοιπο δεν επαρκεί για να πραγματοποιηθεί η συναλλαγή , τότε επιστρέφεται σαν απάντηση η `SW_NEGATIVE_BALANCE` η οποία ισοδυναμεί με τον αριθμό `0x6A85` και ο οποίος τελικά θα είναι αυτός που θα αποτελέσει την status word της απάντησης. Αν τελικά όλα πάνε καλά ενημερώνεται το υπόλοιπο και αποστέλλεται η στάνταρ status word όταν δεν υπάρχει κανέναν πρόβλημα η `0x9000`.

5.4.3 Μέθοδος getBalance

```
private void getBalance(APDU apdu) {
    byte[] buffer = apdu.getBuffer();

    // inform system that the applet has finished
    // processing the command and the system should
    // now prepare to construct a response APDU
    // which contains data field
    short le = apdu.setOutgoing();

    if ( le < 2 )
       ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

    //informs the CAD the actual number of bytes returned
    apdu.setOutgoingLength((byte)2);

    // move the balance data into the APDU buffer
    // starting at the offset 0
    buffer[0] = (byte)(balance >> 8);
```

```
    buffer[1] = (byte)(balance & 0xFF);

    // send the 2-byte balance at the offset
    // 0 in the apdu buffer
    apdu.sendBytes((short)0, (short)2);
  }
```

Η μέθοδος getBalance η οποία επιστρέφει το υπόλοιπο είναι και η μόνη μέθοδος που εκτός από τα status words αποστέλλει και δεδομένα σαν απάντηση. Έτσι αρχικά δηλώνεται ότι η απάντηση θα περιέχει πεδίο με δεδομένα το οποίο θα αποτελείται από 2 bytes και έπειτα τοποθετείται το υπόλοιπο μέσα στα δύο πρώτα bytes του buffer της APDU της απάντησης.

ΚΕΦΑΛΑΙΟ 6

ΔΙΑΧΕΙΡΙΣΗ JAVA CARD ΑΠΟ PC

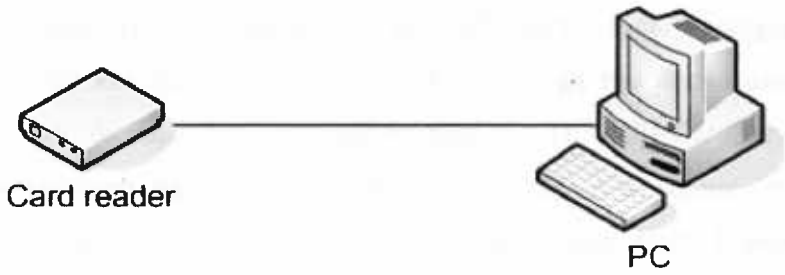
Κεφάλαιο 6

6. ΔΙΑΧΕΙΡΙΣΗ JAVA CARD ΑΠΟ PC

Στο κεφάλαιο αυτό θα αναπτύξουμε μία εφαρμογή για PC γραμμένη σε Java , η οποία με τη βοήθεια του OpenCard Framework θα μπορεί να διαχειρίζεται το applet του ηλεκτρονικού πορτοφολιού που παρουσιάσαμε στο κεφάλαιο 5. Η εφαρμογή αυτή θα παρέχει ένα γραφικό περιβάλλον για την αλληλεπίδραση με τον χρήστη και την πραγματοποίηση των λειτουργιών της κάρτας, ενώ θα επικοινωνεί με την κάρτα μέσω των μηχανισμών του OpenCard Framework .

6.1 Αρχιτεκτονική

Στην παράγραφο αυτή θα δούμε τα βασικά στοιχεία του συστήματος , τον τρόπο λειτουργίας και τον τρόπο επικοινωνίας των επιμέρους μερών. Στο σχήμα 6.2.1 αναπαριστάται το σενάριο διαχείρισης κάρτας από PC.



Σχήμα 6.2.1 – Διαχείρισης κάρτας από PC

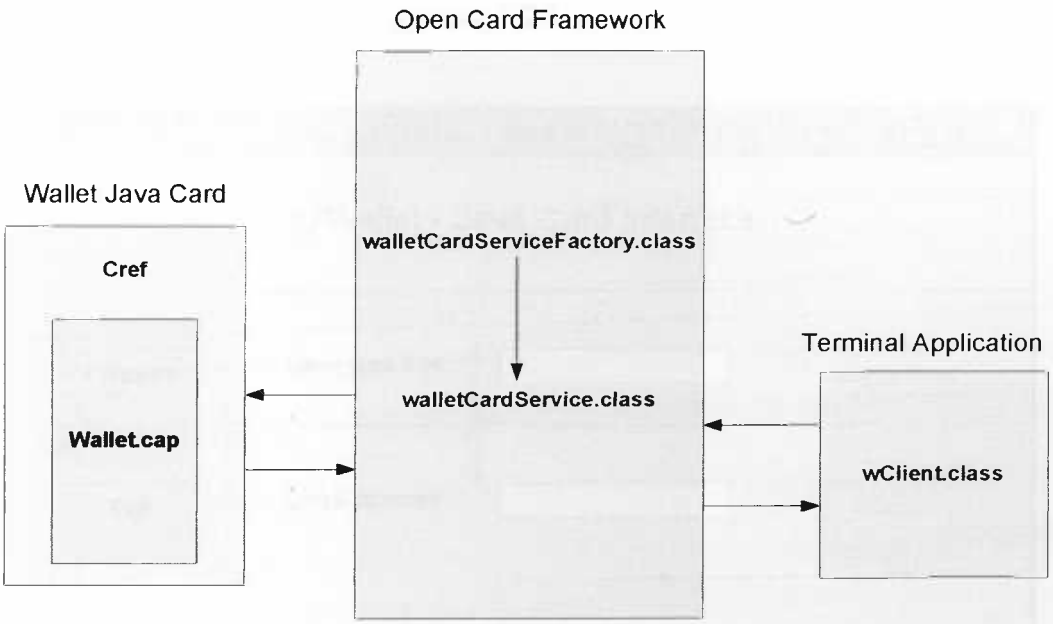
Τα τμήματα του συστήματος που είναι αντιληπτά από το χρήστη , είναι η Java Card και η τερματική εφαρμογή. Για να επικοινωνήσουν όμως αυτά τα δύο συστατικά με ένα τρόπο ώστε να είναι ανεξάρτητος από τον αναγνώστη καρτών ή τον τύπο της κάρτας , ενώ θα μπορεί να χρησιμοποιηθεί και από άλλου τύπου τερματικές εφαρμογές (εφαρμογή κινητού τηλεφώνου – Κεφάλαιο 7), θα πρέπει να μεσολαβήσει το Open Card Framework.

Στο κεφάλαιο 4 αναφερθήκαμε με λεπτομέρεια στην χρησιμότητα και την αρχιτεκτονική του Open Card Framework και εδώ θα δούμε πρακτικά πώς ακριβώς δουλεύει. Για να στήσουμε το framework για το applet μας θα χρειαστεί να αναπτύξουμε δύο συστατικά , το walletCardService και το walletCardServiceFactory. Ενώ για να επιτύχει το εγχείρημα θα πρέπει να δημιουργήσουμε και το opencard.properties αρχείο, το οποίο είναι ένα απλό αρχείο κειμένου, απαραίτητο όμως για τη λειτουργία του Open Card στο οποίο δηλώνουμε το service factory που χρησιμοποιούμε καθώς και τον driver του αναγνώστη καρτών που μας παρέχεται από τον κατασκευαστή. Στην περίπτωση του wallet applet που χρησιμοποιούμε το walletCardServiceFactory και σαν αναγνώστη χρησιμοποιούμε το cref, το opencard.properties θα έχει την ακόλουθη μορφή :

OpenCard.services = wallet.walletCardServiceFactory

OpenCard.terminals=
com.sun.javacard.crefterminal.CrefCardTerminalFactory|cref|CREF|localhost:9025

Σε γενικές γραμμές το walletCardServiceFactory έχει καταχωρημένο το walletCardService και φροντίζει για την επικοινωνία με την κάρτα ανάλογα με τις παραμέτρους που δώσαμε στο opencard.properties. Το walletCardService από την μεριά του αντιστοιχεί στο wallet applet (στο κεφάλαιο 4 είπαμε ότι ένα applet αντιστοιχεί ένα προς ένα με ένα CardService του Open Card Framework) και υλοποιεί όλες τις λειτουργίες της κάρτας, καθώς μιλάει απευθείας με την κάρτα. Μπορούμε να σκεφτούμε το CardService σαν ένα πιο ευέλικτο και δυναμικό apdutool που χρησιμοποιήσαμε στην 5.3. Το walletCardService επομένως είναι υπεύθυνο να επιλέξει το wallet applet στην κάρτα και να αποστείλει τις APDU επαλήθευσης κωδικού, ανανέωσης κωδικού, πίστωσης, χρέωσης και ερώτησης υπολοίπου όταν του ζητηθεί από την εκάστοτε τερματική εφαρμογή.



Σχήμα 6.1.1 – Αρχιτεκτονική του Συστήματος

Στο σχήμα 6.1.1 βλέπουμε την αρχιτεκτονική του συστήματος και την ροή της πληροφορίας. Όταν μια κάρτα εισαχθεί, το Open Card Framework ψάχνει τα δηλωμένα Card Service Factories που βρίσκονται στην Card Service Registry και δημιουργεί ένα στιγμιότυπο του Card Service που αντιστοιχεί στη συγκεκριμένη κάρτα. Η τερματική εφαρμογή συνδέεται με το walletCardService και επικαλείται

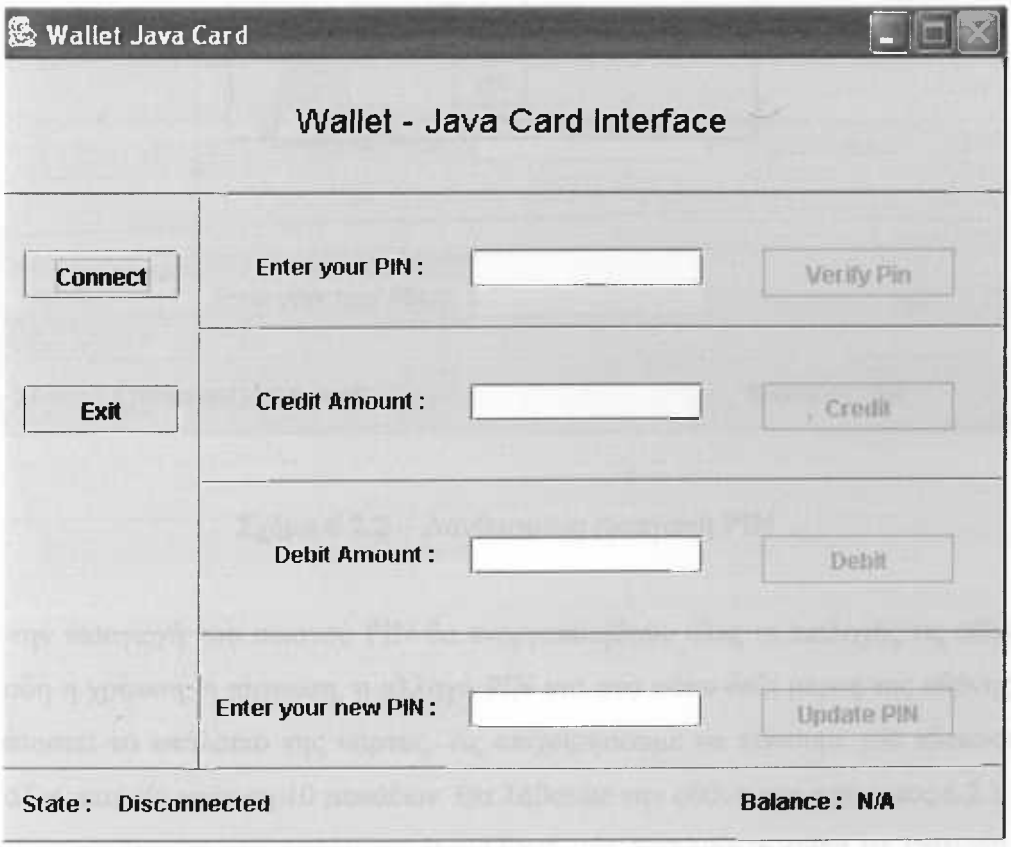
μεθόδους του walletCardService οι οποίες έχουν ως αποτέλεσμα την αποστολή APDUs στην κάρτα. Στη συνέχεια η κάρτα απαντάει στο walletCardService και η τερματική εφαρμογή ενημερώνεται για την απάντηση αυτή. Στην παράγραφο της συντήρησης θα δούμε αναλυτικά των κώδικα που κρύβεται πίσω από αυτά τα συστατικά.

6.2 Χρήση Εφαρμογής

Για την λειτουργία της εφαρμογής θα πρέπει καταρχάς να έχουμε τρέξει σε ένα command prompt το cref που προσομοιώνει το περιβάλλον της κάρτας. Έτσι υποθέτοντας ότι το applet βρίσκεται στο αρχείο image, δίνοντας *cref –i image* η κάρτα περιμένει για αιτήσεις. Σε ένα άλλο command prompt τρέχουμε την τερματική μας εφαρμογή :

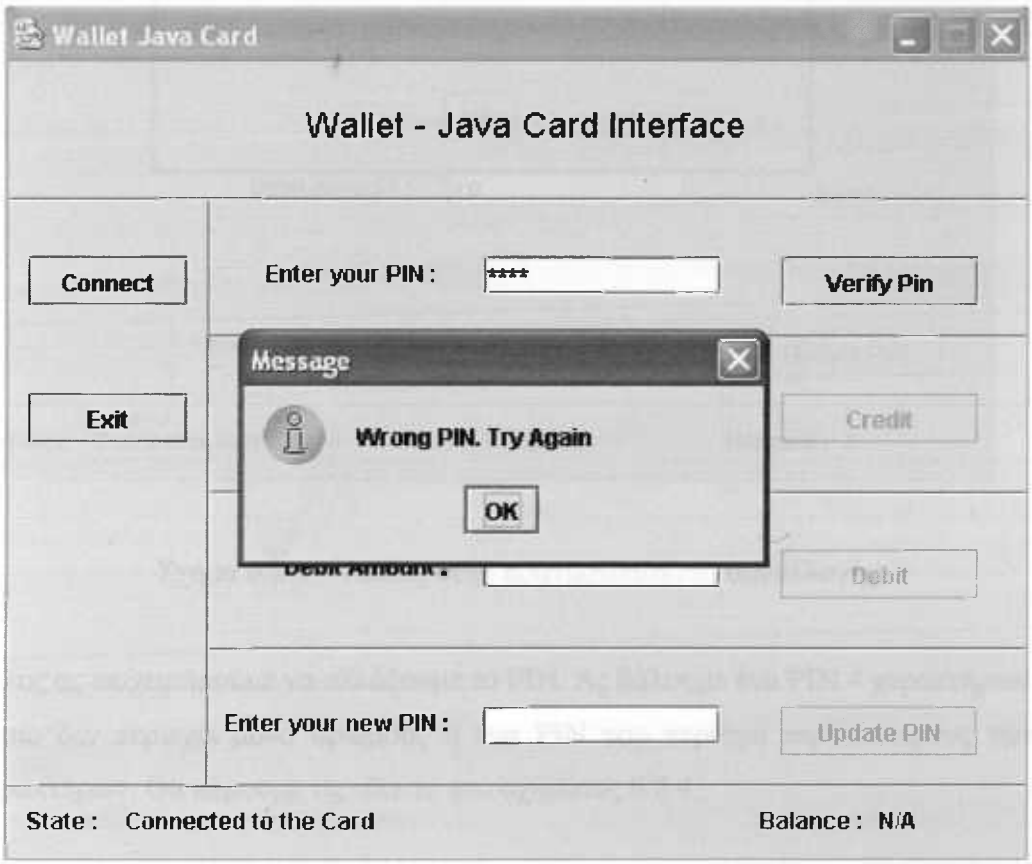
java wClient

και παίρνουμε την οθόνη του σχήματος 6.2.1.



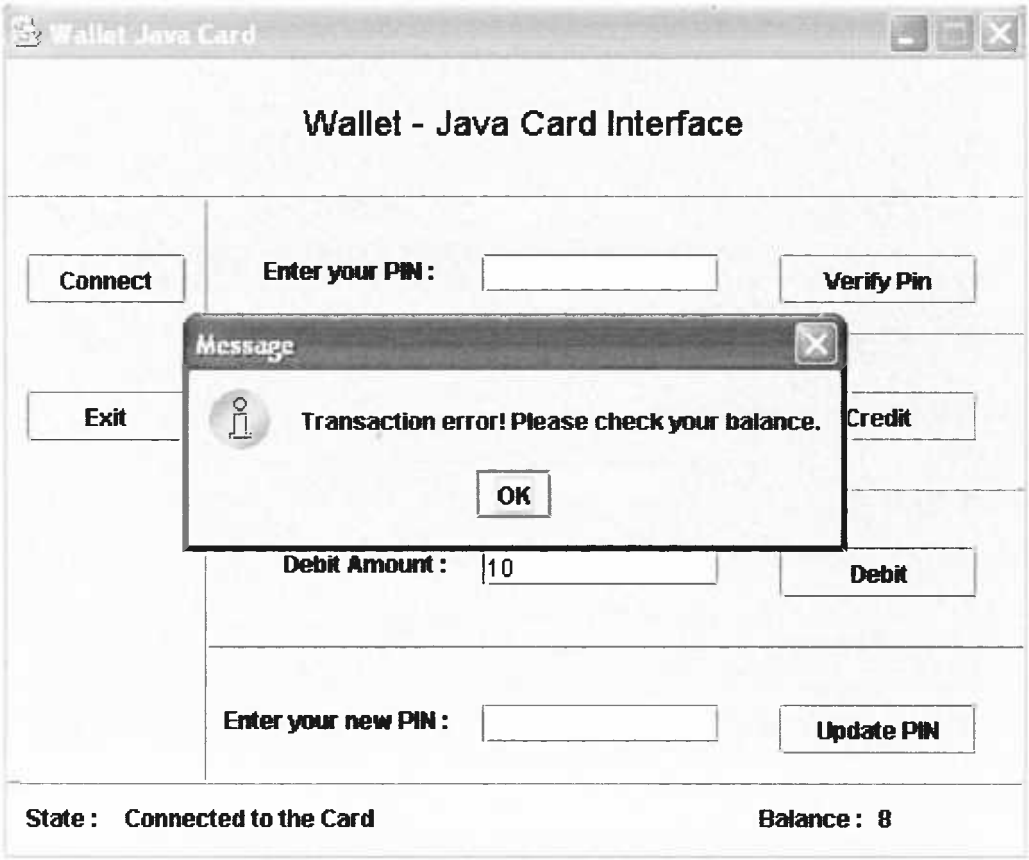
Σχήμα 6.2.1 – Αρχική οθόνη εφαρμογής

Αρχικά θα πρέπει να συνδεθούμε με την κάρτα πατώντας το κουμπί 'Connect' . Αν η σύνδεση είναι επιτυχής, όπως επίσης και αν υπάρξει κάποιο πρόβλημα κατά την σύνδεση, θα ενημερωθούμε στο κάτω μέρος του παραθύρου στο πεδίο της κατάστασης "state", Αν η σύνδεση είναι επιτυχής , θα κληθούμε να δώσουμε τον κωδικό μας PIN. Αν ο PIN είναι σωστός θα ειδοποιηθούμε με ανάλογο μήνυμα αλλιώς θα πάρουμε το μήνυμα της οθόνης του σχήματος 6.2.2.



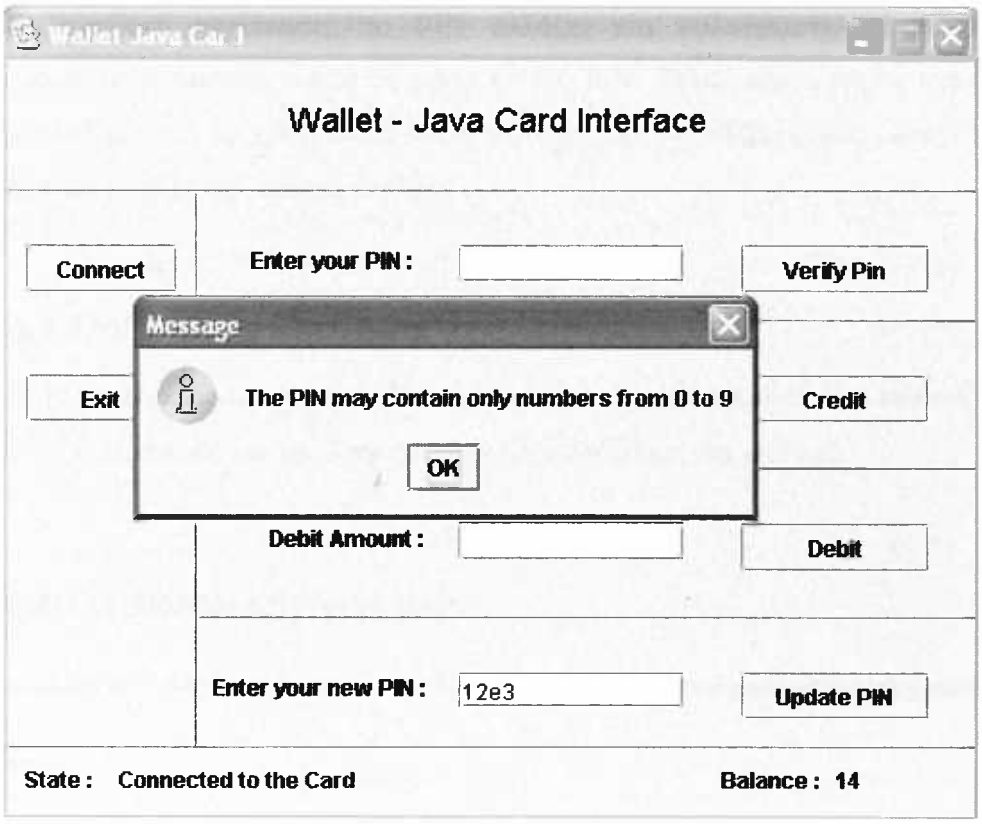
Σχήμα 6.2.2 – Λανθασμένη εισαγωγή PIN

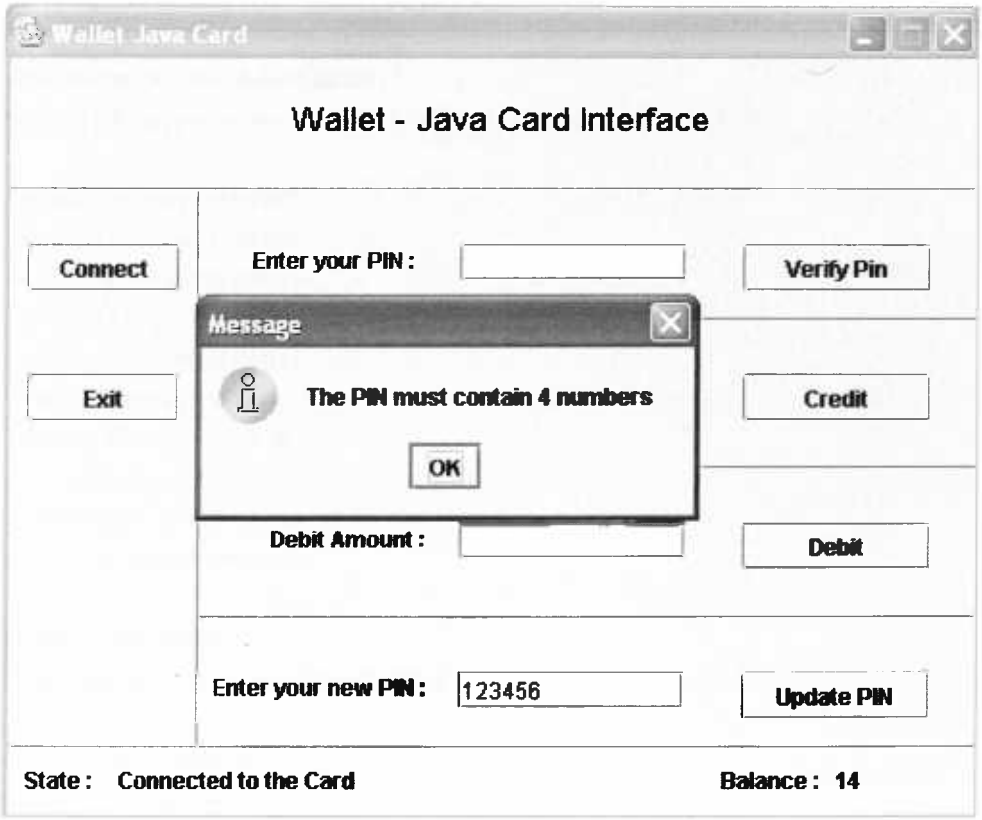
Με την εισαγωγή του σωστού PIN θα ενεργοποιηθούν όλες οι επιλογές τις οθόνης, δηλαδή η χρέωση, η πίστωση, η αλλαγή PIN και στο κάτω δεξί μέρος της οθόνης θα εμφανιστεί το υπόλοιπο της κάρτας. Ας επιχειρήσουμε να κάνουμε μία πίστωση 8 μονάδων και μία χρέωση 10 μονάδων. Θα λάβουμε την οθόνη του σχήματος 6.2.3. Σε αντίθετη περίπτωση και εφόσον η συναλλαγή μας πραγματοποιηθεί με επιτυχία θα λάβουμε ανάλογο μήνυμα.



Σχήμα 6.2.3 – Λάθος στην πραγματοποίηση συναλλαγής

Τέλος ας επιχειρήσουμε να αλλάξουμε το PIN. Ας βάλουμε ένα PIN 4 χαρακτήρων το οποίο δεν περιέχει μόνο αριθμούς ή ένα PIN που περιέχει περισσότερους των 4 χαρακτήρων. Θα πάρουμε τις οθόνες του σχήματος 6.2.4.





Σχήμα 6.2.4 - Μη αποδεκτό PIN

Σε αντίθετη περίπτωση, το PIN αλλάζει και καλούμαστε να κάνουμε πάλι επαλήθευση δίνοντας αυτή τη φορά το νέο PIN. Τέλος πατώντας το κουμπί "Exit" βγαίνουμε από το πρόγραμμα και κλείνουμε και τη σύνδεση μας με την κάρτα (το cref τερματίζει την λειτουργία του)

6.4 Συντήρηση Συστήματος

Στην παράγραφο αυτή θα αναλύσουμε κάποια σημαντικά κομμάτια κώδικα του Open Card Framework και της Java εφαρμογής διαχείρισης της κάρτας.

6.4.1 walletCardService.debit

```
public int debit(int amount) throws CardServiceUnexpectedResponseException {

try{

//Check if applet is selected
if (!appletSelected) selectApplet();

//Construct the debit APDU
byte[] debit=new byte[(7)];

 debit[0]=(byte)0xB0;
 debit[1]=(byte)0x40;
 debit[2]=(byte)0x00;
 debit[3]=(byte)0x00;
 debit[4]=(byte)0x01;
 debit[5]=(byte)amount;
 debit[6]=(byte)0x7F;

//Allocate channel to send the APDU
allocateCardChannel();

//Send the APDU
ResponseAPDU response=sendAPDU(debit);

//Get the response
switch (response.sw() & 0xFFFF) {

//If everything is OK
case OK:
```

```
return 1;

//If there is a problem
default:
throw new CardServiceUnexpectedResponseException(" Unexpected Status : 0x" +
Integer.toHexString((short) (response.sw() & 0xFFFF)));
}

} catch(Exception e) { e.printStackTrace(); return 0;}

finally { releaseCardChannel();
}
}
```

Ας δούμε αναλυτικά την μέθοδο debit της κλάσης walletCardService που είναι κομμάτι του Open Card Framework. Η μέθοδος δέχεται σαν είσοδο το ποσό που θέλουμε να χρεωθεί και επιστρέφει 1 αν η διαδικασία ήταν επιτυχής αλλιώς επιστρέφει 2. Αρχικά ελέγχεται αν έχει επιλεγεί το Wallet Applet. Αν όχι εκτελείτε η μέθοδος selectApplet() η οποία στέλνει το APDU για να επιλεγεί το applet μας. Ύστερα πρέπει να κατασκευάσουμε την APDU εντολή για την χρέωση η οποία θα έχει τα τέσσερα πρώτα bytes σταθερά και θα ακολουθήσει το ποσό της συναλλαγής. Δεσμεύουμε ένα κανάλι πάνω από το οποίο θα στείλουμε την APDU και στη συνέχεια την αποστέλλουμε κρατώντας και την απάντηση που επιστρέφεται. Τέλος , ανάλογα με την απάντηση η μέθοδος επιστρέφει 1 αν η απάντηση είναι 0χ9000 ή σε οποιαδήποτε άλλη περίπτωση επιστρέφει 0.

6.4.2 wClient.connect

```
public void connect(){

try{

//Initialize Open Card Framework
SmartCard.start();

//Wait until a card is inserted to the Card Terminal
cr=new CardRequest(CardRequest.ANYCARD,null,walletCardService.class);

myCard=SmartCard.waitForCard(cr);

//Get the card service walletCardService
```

```
wcs=(walletCardService)myCard.getCardService(walletCardService.class,true);
stateLabel.setText("Connected to the Card");
verifyButton.setEnabled(true);
}

catch (Exception e) {
stateLabel.setText("Cannot Connect to the Card");
e.printStackTrace();}
finally{}

    }
```

Η μέθοδος connect() της εφαρμογής διαχείρισης της κάρτας συνδέει στην ουσία την Java εφαρμογή και το Open Card Framework ώστε να μπορεί αυτή να επικοινωνεί με την Java Card μέσω του OCF. Πάντα πριν χρησιμοποιηθεί το OCF πρέπει να γίνεται εκκίνηση του Framework με την `SmartCard.start()`. Έπειτα, περιμένουμε μέχρι να μπει η κάρτα που περιέχει το applet που αντιστοιχεί στο walletCardService στον card reader . Όταν συμβεί αυτό , τότε δημιουργούμε ένα στιγμιότυπο της κλάσης walletCardService ώστε να μπορούμε να καλούμε μεθόδους τις και κατ' επέκταση να επικοινωνούμε με την κάρτα. Για οποιαδήποτε πρόβλημα σε αυτή τη σύνδεση δημιουργείται εξαίρεση και ο χρήστης της εφαρμογής ενημερώνεται.

6.4.3 wClient.verifyButtonActionPerformed

```
private void verifyButtonActionPerformed(java.awt.event.ActionEvent evt) {

//Check if the given PIN has length=4
if (pinText.getText().toCharArray().length==4)
   {

      try{

    //Convert the given PIN to char array
    char ca[]= pinText.getText().toCharArray();

    //Convert the char array into a byte array
    byte b1= (new Integer(String.valueOf(ca[0]))).byteValue();
       byte b2= (new Integer(String.valueOf(ca[1]))).byteValue();
         byte b3= (new Integer(String.valueOf(ca[2]))).byteValue();
           byte b4= (new Integer(String.valueOf(ca[3]))).byteValue();

byte pin[]={(byte)b1,(byte)b2,(byte)b3,(byte)b4};

//Call the verfyPIN method of the walletCardService class
//and check if everything was OK or not
if (wcs.verifyPIN(pin)==1)
{
creditButton.setEnabled(true);
debitButton.setEnabled(true);
updateButton.setEnabled(true);
```

```
pinText.setText("");
showMsg("PIN Correct!");

try{
        //Call the getBalance method of the walletCardService class
        int balance= wcs.getBalance();
        labelBalance.setText(String.valueOf(balance));
        }

        catch(Exception e){}
}

else
{
    showMsg("Wrong PIN. Try Again");
}
        }

     //if the PIN doesn't contain only numbers
     catch (Exception e) {
  showMsg("The PIN may contain only numbers from 0 to 9");
e.printStackTrace();}
        }

        //if the PIN contains more than 4 numbers
        else
        {
            showMsg("The PIN must contain 4 numbers") ;
        }
    }
```

Το παραπάνω απόσπασμα κώδικα αφορά τις ενέργειες που γίνονται όταν πατιέται το κουμπί 'verify' τη εφαρμογής. Αρχικά γίνεται ο έλεγχος αν το PIN που έχει εισαγάγει ο χρήστης έχει μήκος 4. Εάν πράγματι το PIN έχει μήκος 4 χαρακτήρες τότε ακολουθούν κάποιες μετατροπές ώστε ο PIN από ένα απλό String να γίνει πίνακας από bytes. Έτσι μπορούμε να καλέσουμε την μέθοδο verifyPIN του walletCardService και να του περάσουμε σαν όρισμα τον πίνακα bytes που περιέχει τον κωδικό. Αν ο κωδικός είναι σωστός η μέθοδος θα επιστρέψει 1 και ο χρήστης θα ενημερωθεί για την επιτυχή κατάληξη της διαδικασίας. Σε αντίθετη περίπτωση ο χρήστης ενημερώνεται, ενώ αν προκύψει κάποιο πρόβλημα στην κλήση της μεθόδου πράγμα που σημαίνει ότι το PIN δεν αποτελούταν μόνο από νούμερα δημιουργείται μία εξαίρεση και ενημερώνεται ο χρήστης. Εάν τέλος η διαδικασία πραγματοποιηθεί με επιτυχία, τότε καλείται και η μέθοδος getBalance του walletCardService ώστε να ενημερωθεί ο χρήστης αυτόματα για την ανανέωση του υπολοίπου.

Και οι υπόλοιπες λειτουργίες της κάρτας , υλοποιούνται με τον ίδιο τρόπο, με επίκληση δηλαδή μεθόδων του walletCardService και με κατάλληλο χειρισμό των εξαιρέσεων και των σφαλμάτων που μπορούν να προκύψουν.

ΚΕΦΑΛΑΙΟ 7

ΔΙΑΧΕΙΡΙΣΗ JAVA CARD ΑΠΟ ΚΙΝΗΤΗ ΣΥΣΚΕΥΗ

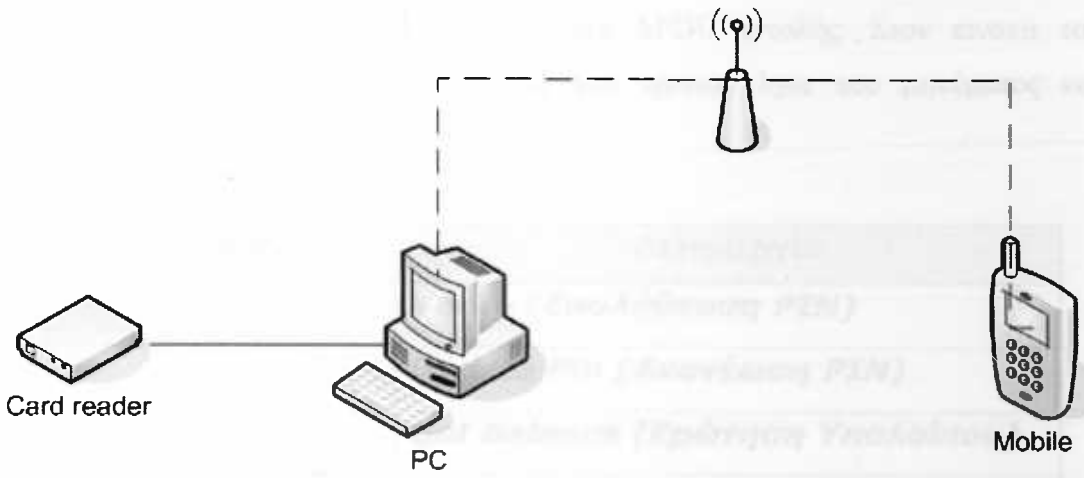
Κεφάλαιο

7. ΔΙΑΧΕΙΡΙΣΗ JAVA CARD ΑΠΟ ΚΙΝΗΤΗ ΣΥΣΚΕΥΗ

Στο κεφάλαιο αυτό θα αναπτύξουμε μία εφαρμογή για κινητή συσκευή με την τεχνολογία J2ME, η οποία με τη βοήθεια του OpenCard Framework θα μπορεί να διαχειρίζεται το applet του ηλεκτρονικού πορτοφολιού που παρουσιάσαμε στο κεφάλαιο 5. Η εφαρμογή της κινητής συσκευής θα επικοινωνεί με sockets με μία εφαρμογή εξυπηρετητή σε ένα PC το οποίο με τη σειρά του θα είναι συνδεδεμένο με τον αναγνώστη καρτών.

7.1 Αρχιτεκτονική

Στην παράγραφο αυτή θα δούμε τα βασικά στοιχεία του συστήματος , τον τρόπο λειτουργίας και τον τρόπο επικοινωνίας των επιμέρους μερών. Στο σχήμα 7.1.1 αναπαριστάται το σενάριο διαχείρισης κάρτας από κινητή συσκευή μέσω PC.



Σχήμα 7.1.1 – Διαχείρισης κάρτας από κινητή συσκευή

Στο κεφάλαιο 6 παρουσιάσαμε μία εφαρμογή που χρησιμοποιώντας το Open Card Framework διαχειρίζεται την Java Card του ηλεκτρονικού πορτοφολιού. Και σε αυτό το σενάριο λοιπόν θα χρησιμοποιήσουμε το Open Card Framework, επωφελούμενοι από τα οφέλη που μας παρέχει και αποδεικνύοντας έμπρακτα την χρήση του ως διεπάφη με την κάρτα για μια πληθώρα εφαρμογών.

Συνεπώς και για την υλοποίηση και αυτού του σεναρίου, θα χρησιμοποιήσουμε τα συστατικά της Java Card και του Open Card Framework που έχουμε ήδη υλοποιήσει. Τα δύο επιπλέον συστατικά που θα χρησιμοποιήσουμε είναι μία εφαρμογή για την κινητή συσκευή γραμμένη σε J2ME και μία εφαρμογή java η οποία θα αναλάβει να επικοινωνεί με το Open Card Framework και να προωθεί τις αιτήσεις που θα καταφθάνουν από την εφαρμογή της κινητής συσκευής.

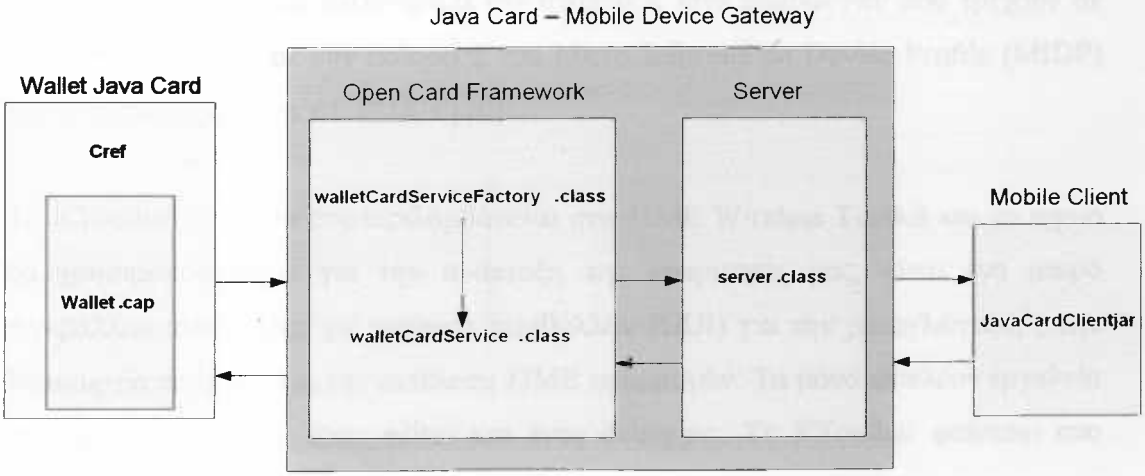
Η επικοινωνία μεταξύ της κινητής συσκευής και του PC θα γίνει μέσω sockets όπου η J2ME εφαρμογή θα παίζει τον ρόλο του πελάτη (client) ενώ η εφαρμογή το PC τον ρόλο του εξυπηρετητή. Η εφαρμογή J2ME μέσω GPRS ή με κάποιον άλλο τρόπο που θα τις παρέχει σύνδεση στο internet θα χρησιμοποιεί την IP του PC που φιλοξενεί τον server για να του αποστείλει μηνύματα τα οποία θα ενεργοποιούν κάποιες APDU εντολές ενώ ο server θα αποστέλλει στην IP που έχει δοθεί στην κινητή συσκευή τις απαντήσεις αυτών των APDU εντολών. Στο μήνυμα που θα στέλνεται στο server το πρώτο byte θα προσδιορίζει την κάθε λειτουργία ενώ τα επόμενα τα δεδομένα που

τοποθετούνται στο data field της εκάστοτε APDU εντολής. Στον πίνακα του σχήματος 7.1.2 φαίνεται η αντιστοιχία του πρώτου byte του μηνύματος και λειτουργιών.

Πρώτο Byte	Λειτουργία
0	***Verify (Επαλήθευση PIN)***
1	***Update Pin (Ανανέωση PIN)***
2	***Get Balance (Ερώτηση Υπολοίπου)***
3	***Debit (Χρέωση)***
4	***Credit (Πίστωση)***

Σχήμα 7.1.2 – Αντιστοιχία Λειτουργιών

Η αρχιτεκτονική του συστήματος φαίνεται στο σχήμα 7.1.3, όπου φαίνονται και ποιες κλάσεις αποτελούν ποια συστατικά.



Σχήμα 7.1.3 - Αρχιτεκτονική του Συστήματος

Ο χρήστης επιλέγει κάποια λειτουργία από την διεπαφή της εφαρμογής της κινητής συσκευής και εκείνη στέλνει αντίστοιχο μήνυμα στον server μέσω sockets. Ο server επικοινωνεί με το Open Card Framework και βάση του μηνύματος που λαμβάνει εκτελεί ανάλογη μέθοδο του walletCardService που αντιστοιχεί στην αποστολή

κάποιο APDU στην κάρτα. Η κάρτα απαντάει στην εντολή και ακολουθείται η αντίστροφη διαδικασία. Η κάρτα στέλνει την απάντησή της στο Open Card Framework το οποίο με τη σειρά του επιστρέφει για την μέθοδο που είχε κληθεί αντίστοιχη απάντηση. Ο server παίρνει την απάντηση του Open Card Framework και στέλνει κατάλληλο μήνυμα στην J2ME εφαρμογή. Στην παράγραφο της συντήρησης θα δούμε αναλυτικά των κώδικα που κρύβεται πίσω από αυτά τα συστατικά.

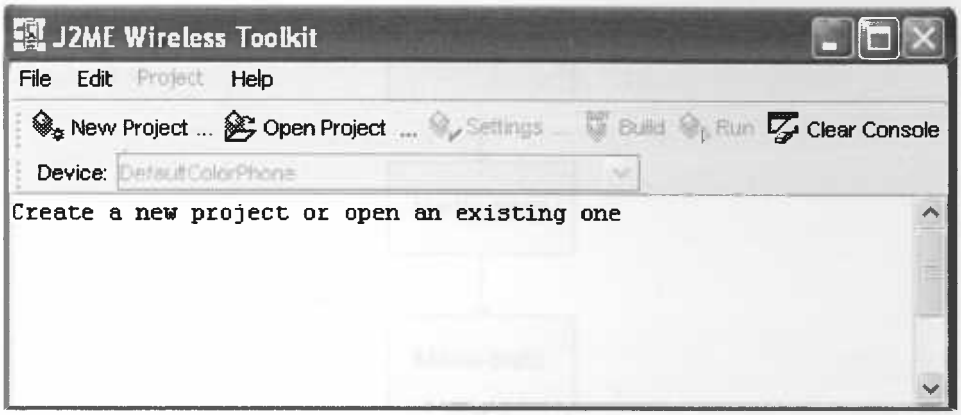
7.2 Υλοποίηση

Ας δούμε με ποιον τρόπο θα υλοποιήσουμε τα επιμέρους συστατικά του συστήματος, δηλαδή την J2ME εφαρμογή και την Java εφαρμογή του server.

7.2.1 J2ME Java Card Client

Για την υλοποίηση της J2ME εφαρμογής θα χρησιμοποιήσουμε το Java 2 Platform Micro Edition Wireless Toolkit 2.0 το οποίο διατίθεται δωρεάν από τη SUN. Το J2ME Wireless Toolkit υποστηρίζει την ανάπτυξη Java εφαρμογών που τρέχουν σε συσκευές συμβατές με την έκδοση 2 του Micro Information Device Profile (MIDP) όπως κινητά τηλέφωνα και PDA's [10].

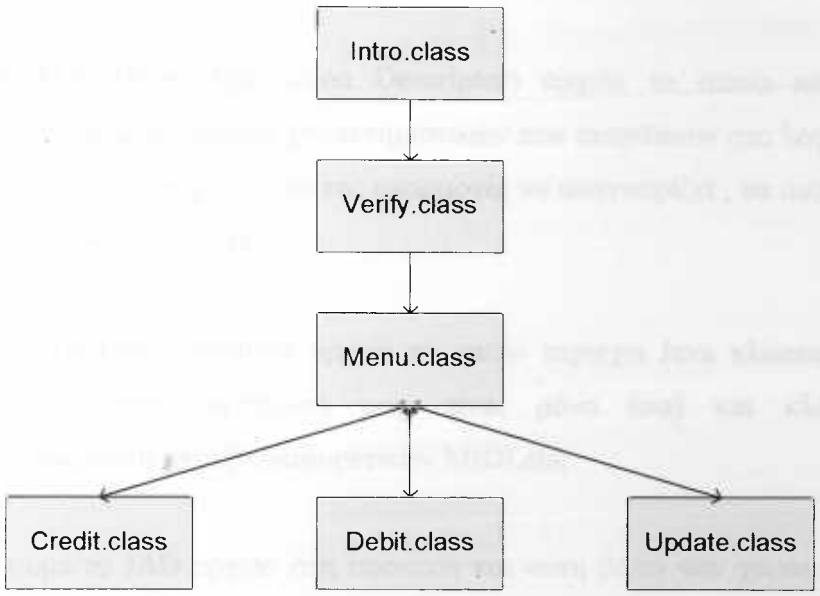
Το KToolbar το οποίο συμπεριλαμβάνεται στο J2ME Wireless Toolkit και το οποίο θα χρησιμοποιήσουμε για την ανάπτυξη της εφαρμογής μας, είναι ένα μικρό περιβάλλον ανάπτυξης με γραφικό περιβάλλον (GUI) για την μεταγλώττιση , την δημιουργία πακέτων και την εκτέλεση J2ME εφαρμογών. Τα μόνο επιπλέον εργαλεία που χρειάζονται είναι ένας editor και ένας debugger. Το KToolbar φαίνεται στο σχήμα 7.2.1.1.



Σχήμα 7.2.1.1 – Η αρχική οθόνη του KToolbar

Εμείς αρχικά θα δημιουργήσουμε ένα project JavaCardClient και αυτόματα θα δημιουργηθεί μία δομή φακέλων μέσα σε ένα φάκελο με το όνομα του project και θα περιέχει φακέλους που θα τοποθετηθούν ο πηγαίος κώδικας , οι κλάσεις και το τελικό πακέτο των κλάσεων. Έτσι είμαστε έτοιμοι να αρχίσουμε να δημιουργούμε τις κλάσεις μας μέσα στο φάκελο 'src' αφού πρώτα έχουμε κάνει και τις απαραίτητες ρυθμίσεις στα settings του project μας , όπως για παράδειγμα ποια είναι η πρώτη κλάση της εφαρμογής που θα τρέξει καθώς και άλλες ρυθμίσεις που αφορούν εκδόσεις , όνομα εφαρμογής, δικαιώματα κ.τ.λ.

Για την ανάπτυξη το Java Card Client θα υλοποιήσουμε επτά διαφορετικές κλάσεις. Η σχέση τους αποτυπώνεται στο σχήμα 7.2.1.2.



Σχήμα 7.2.1.2 – Σχέση των κλάσεων του MIDLet

Η κλάση Intro αποτελεί την αρχική οθόνη η οποία θα μας εισάγει στην εφαρμογή και θα μας μεταφέρει στην κλάση Verify η οποία είναι υπεύθυνη για την επαλήθευση του κωδικού PIN. Αν ο κωδικός είναι σωστός επιτρέπεται η πρόσβαση στην κλάση Menu η οποία αποτελεί και το βασικό μενού , όπου ο χρήστης θα πραγματοποιεί τις λειτουργίες της χρέωσης , της πίστωσης , της ανανέωσης κωδικού και της ερώτησης υπολοίπου. Σε αυτές τις λειτουργίες αντιστοιχούν οι κλάσεις Debit, Credit και Update. Για την ερώτηση υπολοίπου δεν κατασκευάστηκε ξεχωριστή κλάση μιας και η λειτουργικότητά της ενσωματώνεται στην Menu , όπου κάποιος μπορεί ενώ βρίσκεται στο μενού να πάρει και το διαθέσιμο υπόλοιπο. Τέλος , πολύ σημαντική κλάση η οποία δεν φαίνεται στο σχήμα 7.2.1.2 αλλά είναι απαραίτητη για την λειτουργία της εφαρμογής είναι η κλάση Sender. Οι κλάσεις που πρέπει να στείλουν κάποιο μήνυμα στον server, το στέλνουν μέσω της Sender , η οποία αναλαμβάνει να δημιουργεί το socket και να αποστέλλει στον server το εκάστοτε μήνυμα.

Υποθέτοντας πως έχουμε τελειώσει με την υλοποίηση της εφαρμογής μας μπορούμε να την τρέξουμε στον προσομοιωτή που μας παρέχεται και μετά για να τη φέρουμε σε μορφή να εγκατασταθεί σε μία πραγματική κινητή συσκευή θα πρέπει από το μενού project του KToolbar να δημιουργήσουμε το πακέτο των κλάσεων. Η ενέργεια αυτή θα έχει ως αποτέλεσμα την κατασκευή ενός MIDLet Suite που περιέχει δύο αρχεία:

- Ένα JAD (Java Application Descriptor) αρχείο το οποίο περιέχει ένα προκαθορισμένο σύνολο χαρακτηριστικών που επιτρέπουν στο λογισμικό της συσκευής που διαχειρίζεται της εφαρμογές να αναγνωρίζει , να ανακτά και να εγκαθιστά τα MIDLets.

- Ένα JAR (Java Archive) αρχείο το οποίο περιέχει Java κλάσεις για κάθε MIDLet (στην περίπτωσή μας, είναι μόνο ένα) και κλάσεις που διαμοιράζονται μεταξύ διαφορετικών MIDLets.

Έτσι στέλνουμε το JAD αρχείο στη συσκευή και αυτή βάση των χαρακτηριστικών που βρίσκονται μέσα στο αρχείο κατεβάζει και εγκαθιστά το JAR αρχείο.

7.2.2 Java Socket Server

Για την υλοποίηση του server όπως προαναφέραμε θα χρησιμοποιήσουμε Java. Ο server θα υλοποιηθεί ως ένα thread που τρέχει συνεχώς και θα είναι σύγχρονος περιμένοντας αιτήσεις και απαντώντας σε αυτές στην πόρτα 4445.

Ο server συνδέεται με το Open Card Framework ενώ βασική δουλειά του είναι να μπορεί να διακρίνει τα μηνύματα που του έρχονται από τον εκάστοτε πελάτη, διαβάζοντας το πρώτο byte και έτσι αναγνωρίζοντας ποια λειτουργία αφορά. Για κάθε λειτουργία λοιπόν επικαλείται την αντίστοιχη μέθοδο του walletCardService περνώντας της σαν όρισμα (αν δέχεται), το υπόλοιπο μήνυμα (αν υπάρχει) που έχει φτάσει από τον client. Τέλος αφού αποστείλει την κατάλληλη απάντηση πίσω στον πελάτη καταγράφει κάποιες βασικές πληροφορίες για την επικοινωνία, όπως την IP διεύθυνση και την πόρτα που επικοινώνησε ο πελάτης, το μήνυμα που έλαβε, το μήνυμα που έστειλε ως απάντηση και την ημερομηνία και ώρα της επικοινωνίας.

7.3 Χρήση Εφαρμογής

Για την λειτουργία της εφαρμογής θα πρέπει καταρχάς να έχουμε τρέξει σε ένα command prompt το cref που προσομοιώνει το περιβάλλον της κάρτας. Έτσι

υποθέτοντας ότι το applet βρίσκεται στο αρχείο image, δίνοντας *cref –i image* η κάρτα περιμένει για αιτήσεις. Σε ένα άλλο command prompt τρέχουμε τον server ο οποίος θα δέχεται αιτήσεις από την κινητή συσκευή :

java server

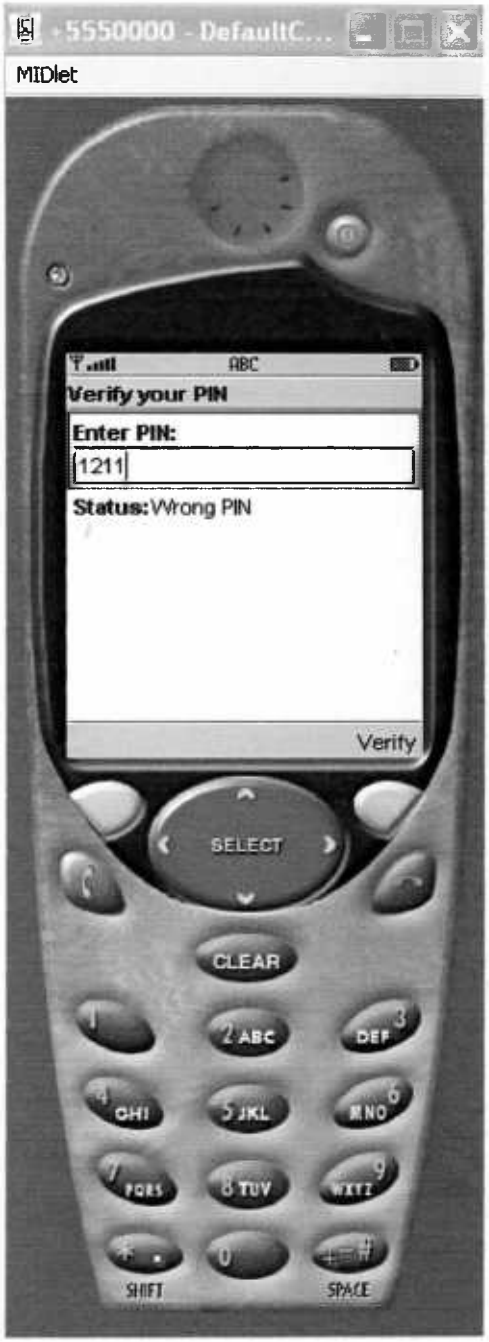
και παίρνουμε την οθόνη του σχήματος 7.3.1.

Σχήμα 7.3.1 – Εκκίνηση Server

Έπειτα χρησιμοποιούμε το J2ME Wireless Toolkit για να προσομοιώσουμε το περιβάλλον του τηλεφώνου και να τρέξουμε την J2ME εφαρμογή. Τρέχοντας λοιπόν το JavaCardClient.jar το οποίο περιέχει όλες τις κλάσης της εφαρμογής μας παίρνουμε την οθόνη του σχήματος 7.3.2.

Σχήμα 7.3.2 – Αρχική οθόνη της εφαρμογής

Πατώντας Start ξεκινάμε την εφαρμογή μας και πάμε στην οθόνη επαλήθευσης του κωδικού, θα κληθούμε να δώσουμε τον κωδικό μας PIN. Αν ο PIN είναι σωστός θα προχωρήσουμε στο κυρίως μενού αλλιώς θα πάρουμε ανάλογο μήνυμα όπως φαίνεται στο σχήμα 7.3.3.

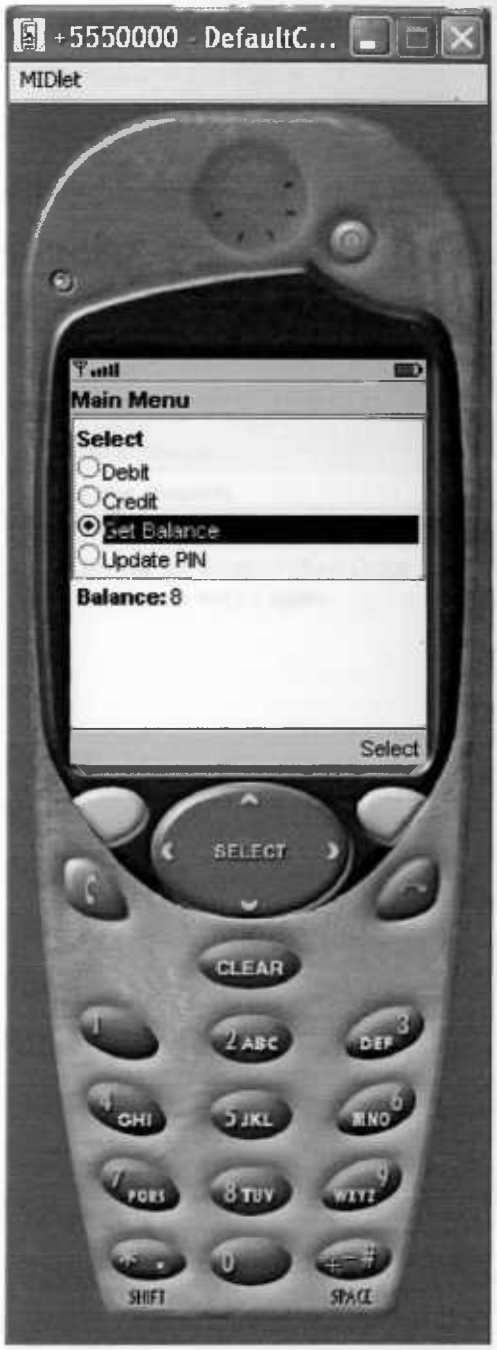


Σχήμα 7.3.3 – Εισαγωγή λάθος PIN

Από την μεριά του ο server καταγράφει τις αιτήσεις που δέχεται και εξυπηρετεί. Στο σχήμα 7.3.4 φαίνεται η κατάσταση του server μετά από δύο προσπάθειες επαλήθευσης κωδικού, μία αποτυχημένη και μία επιτυχημένη.

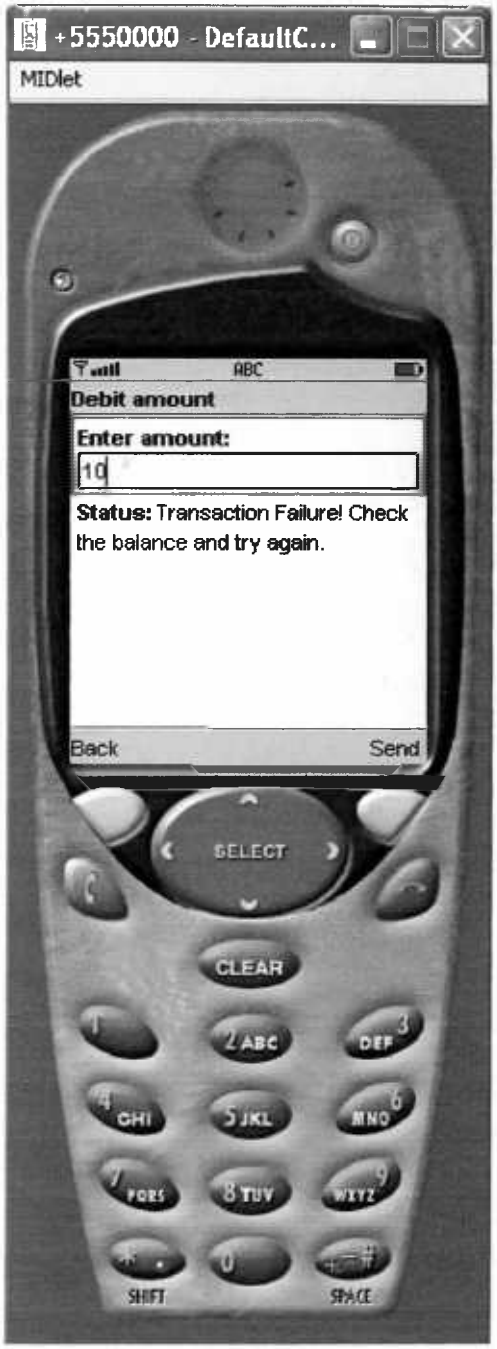
Σχήμα 7.3.4 – Κατάσταση server

Στο κυρίως μενού υπάρχουν οι λειτουργίες της χρέωσης , της πίστωσης ,της ερώτησης υπολοίπου και της αλλαγής PIN. Έχοντας επιλέξει την επιλογή 'Get Balance' θα πάρουμε την οθόνη του σχήματος 7.3.5.



Σχήμα 7.3.5 – Κυρίως μενού

Ας επιλέξουμε να κάνουμε μία χρέωση με την επιλογή 'Debit' και ας επιχειρήσουμε να κάνουμε μία χρέωση 10 μονάδων στην νέα οθόνη που εμφανίζεται. Θα λάβουμε την οθόνη του σχήματος 7.3.6. Σε αντίθετη περίπτωση και εφόσον η συναλλαγή μας πραγματοποιηθεί με επιτυχία θα λάβουμε ανάλογο μήνυμα. Πατώντας την επιλογή 'Back' επιστρέφουμε στο κυρίως μενού.



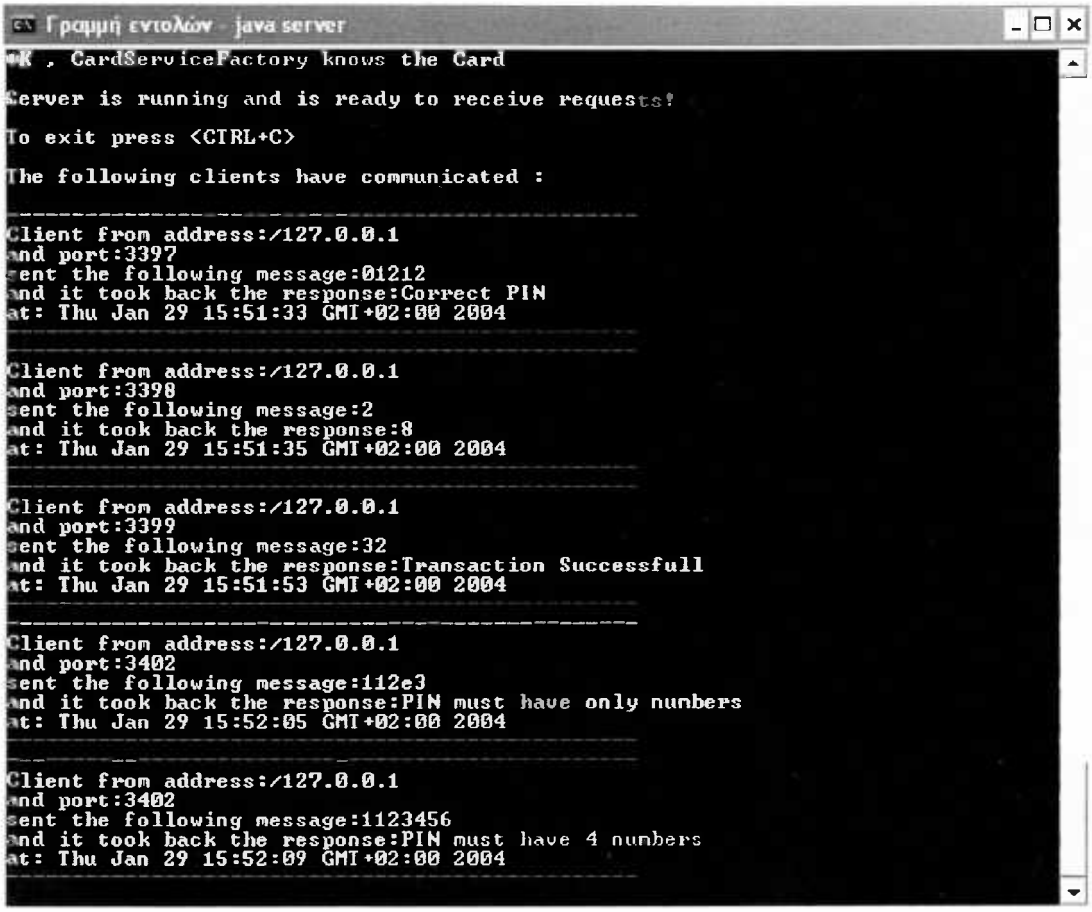
Σχήμα 7.3.6 - Λάθος στην πραγματοποίηση συναλλαγής

Η λειτουργία της πίστωσης ακολουθεί ακριβώς την ίδια διαδικασία. Τέλος ας επιχειρήσουμε να αλλάξουμε το PIN. Επιλέγουμε την επιλογή 'Update PIN' και στην νέα οθόνη ας βάλουμε ένα PIN τεσσάρων χαρακτήρων το οποίο δεν περιέχει μόνο

αριθμούς ή ένα PIN που περιέχει περισσότερους των 4 χαρακτήρων. Θα πάρουμε τις οθόνες του σχήματος 7.3.7.

Σχήμα 7.3.7 – Μη αποδεκτό PIN

Σε αντίθετη περίπτωση , το PIN αλλάζει και ενημερωνόμαστε με ανάλογο μήνυμα. Τέλος ας δούμε στο σχήμα 7.3.8 ,την καταγραφή του server για μια σειρά από αιτήσεις σαν αυτές που περιγράψαμε παραπάνω.



Σχήμα 7.3.8 – Καταγραφή αιτήσεων από τον server

7.4 Συντήρηση Συστήματος

Στην παράγραφο αυτή θα αναλύσουμε κάποια σημαντικά κομμάτια κώδικα του server και της J2ME εφαρμογής.

7.4.1 serverthread.debit

```
public void debit(String receivedString){

//Convert received message to character array
char ca[]= receivedString.trim().toCharArray();

// Remove the first byte to remain the amount
```

```
char temp[]=new char[ca.length-1];

int i;

for (i=1;i<ca.length;i++)

{temp[i-1]=ca[i];}

//Get the amount
String receivedAmount=String.valueOf(temp);

try{

    //Convert the amount from String to integer
    int debitAmount= Integer.parseInt(receivedAmount);

  //Call the debit method of the walletCardService class
  //and check if everything was OK or not
  //then send the response
  if (wcs.debit(debitAmount)==1) {response="Transaction Successfull";}

  //if the debit amount is bigger than balance
  else {response="Transaction Failure! Check the balance and try again.";}
}

//if the format of amount is invalid
catch (Exception e) {response="The amount format is invalid! Try again!";}

}
```

Το παραπάνω απόσπασμα κώδικα αφορά την μέθοδο verify του server η οποία παίρνει σαν όρισμα το μήνυμα που έρχεται από την κινητή συσκευή και έχει σαν πρώτο χαρακτήρα το '3' το οποίο σηματοδοτεί την λειτουργία της χρέωσης ποσού. Αρχικά μετατρέπεται το μήνυμα σε έναν πίνακα χαρακτήρων, έτσι ώστε να αφαιρεθεί ο πρώτος χαρακτήρας και να απομείνει το καθαρό ποσό. Έπειτα καλείται η μέθοδος debit του walletCardService και ανάλογα με την έκβαση της συναλλαγής συντάσσεται η απάντηση η οποία θα αποσταλεί πίσω στον πελάτη.

7.4.2 JavaCardClient.jar – Debit.run

```
public void run() {

try {

//Create new Datagram connection
DatagramConnection dc = (DatagramConnection)
Connector.open("datagram://localhost:4445");

 si.setText("Connected to server");

       //Create a new Sender instance
```

```
        sender = new Sender(dc);

            while (true) {

        //Receive server response
        Datagram dg = dc.newDatagram(100);
        dc.receive(dg);

         // Show the response of the server
         if (dg.getLength() > 0) {
                    si.setText(new String(dg.getData(), 0, dg.getLength()));
                }
            }
         }

catch (ConnectionNotFoundException cnfe) {
            Alert a = new Alert("Client", "Cannot establish connection!",
                                null, AlertType.ERROR);
            a.setTimeout(Alert.FOREVER);
            display.setCurrent(a);
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
```

Το παραπάνω κομμάτι κώδικα είναι η μέθοδος run της κλάσης debit του πακέτου JavaCardClient, της εφαρμογής ουσιαστικά που βρίσκεται στην κινητή συσκευή. Αρχικά δημιουργείται μία σύνδεση με τον server και χρησιμοποιείται η κλάση sender η οποία χρησιμοποιεί αυτή την σύνδεση για να στείλει το μήνυμα που έχει οριστεί. Έπειτα παίρνουμε την απάντηση για το μήνυμα που αποστείλαμε στον server και το εμφανίζουμε στον χρήστη. Στο σημείο αυτό γίνεται και ένας έλεγχος αν το μήνυμα έχει περιεχόμενο ή είναι κενό. Σε περίπτωση που προκύψει κάποιο πρόβλημα με την σύνδεση με τον server ενημερωνόμαστε με ανάλογο μήνυμα.

7.4.3 Διαχείριση κουμπιών κινητής συσκευής

```
  public void commandAction(Command c, Displayable s) {

        //If send or back button is pressed
        if (c == sendCommand && !parent.isPaused() ) {
            sender.send(null, "3"+tf.getString());
```

```
        }
        else if (c == backCommand) {
           Menu menu=new Menu(parent);
            menu.start();
        }
    }
```

Το παραπάνω τμήμα κώδικα αφορά την μέθοδο η οποία διαχειρίζεται τις ενέργειες των κουμπιών για την χρέωση κάποιου ποσού. Αν λοιπόν πατήσουμε το 'send' τότε χρησιμοποιούμε τον sender για να αποστείλουμε το ποσό της χρέωσης , μόνο που προσθέτουμε μπροστά το '3' ώστε να καταλάβει ο server όταν παραλάβει το μήνυμα ότι το μήνυμα αφορά την λειτουργία της χρέωσης. Αν πάλι πατήσουμε το 'Back' τότε πηγαίνουμε πάλι στην οθόνη του κεντρικού μενού.

ΚΕΦΑΛΑΙΟ 8

ΣΥΜΠΕΡΑΣΜΑΤΑ

Κεφάλαιο

8. ΣΥΜΠΕΡΑΣΜΑΤΑ

8.1 Συμπεράσματα

Στην παρούσα εργασία πετύχαμε τον συνδυασμό διαφορετικών τεχνολογιών με την δημιουργία ενός συστήματος που συνδυάζει την Java Card τεχνολογία, την Java και την τεχνολογία J2ME. Χρησιμοποιήσαμε μια σχετικά απλή εφαρμογή για να δείξουμε την λειτουργικότητα του συστήματος και τον τρόπο που ένα τέτοιο σύστημα μπορεί να υλοποιηθεί. Αφού αναφερθήκαμε αναλυτικά στο θεωρητικό κομμάτι των τεχνολογιών και κυρίως της Java Card, υλοποιήσαμε ένα σχετικά απλό σενάριο χρήσης της έξυπνης κάρτας από ένα PC , σενάριο όμως που αποτέλεσε την βάση για να υλοποιήσουμε το πιο σύνθετο σενάριο διαχείρισης της κάρτας από κινητή συσκευή μέσω PC.

Με την ανάπτυξη του ηλεκτρονικού εμπορίου και γενικότερα των νέων τεχνολογιών , γίνεται ακόμα πιο εμφανής η ανάγκη για αύξηση της προσωπικής ασφάλειας αλλά και για μεταφορά πληροφορίας σε ένα φορητό μέσο που διαθέτει το ίδιο την δυνατότητα αυθεντικοποίησης και επεξεργασίας δεδομένων. Ένα τέτοιο μέσο είναι η έξυπνη κάρτα και μάλιστα ιδιαίτερο ενδιαφέρον παρουσιάζει μία συνεχώς εξελισσόμενη τεχνολογία, η Java Card τεχνολογία. Η Java Card, τα επόμενα χρόνια θα κυριαρχήσει στην αγορά έξυπνων καρτών, αφού συνδυάζει τα πολλά πλεονεκτήματα που κληρονομεί από την γλώσσα Java και την εύκολη συνδεσιμότητά της με τις υπόλοιπες ήδη καταξιωμένες Java τεχνολογίες.

Στην εργασία αυτή εκτός από την Java Card τεχνολογία, χρησιμοποιήσαμε και την J2ME. Η τεχνολογία αυτή χρησιμοποιείται σε κινητές συσκευές για να αυξήσει τη λειτουργικότητά τους. Σε μια εποχή που η χρήση κινητών συσκευών και κατά κύριο λόγο των κινητών τηλεφώνων, γνωρίζει κατακόρυφη άνοδο, μπορούμε να φανταστούμε την χρησιμότητα, του να προσθέσουμε λειτουργικότητα σε κάτι που ο καθένας μας έχει επάνω του ανά πάσα στιγμή. Αυτομάτως αυξάνουμε τις δυνατότητες που μας παρέχονται και διευκολύνουμε καθημερινές συνήθειες και ανάγκες, που θα μπορούμε να ικανοποιούμε όποτε και όπου βρεθούμε.

Αυτές τις δύο τεχνολογίες σε συνδυασμό, ανοίγουν νέους δρόμους συναθροίζοντας τα πλεονεκτήματά τους και δίνοντας τη δυνατότητα για κατασκευή ασφαλών εφαρμογών που μπορούμε να τις έχουμε πάντα μαζί μας , αξιοποιώντας στο μέγιστο τις νέες τεχνολογίες. Μπορούμε να σκεφτούμε για παράδειγμα την πλοήγησή μας σε ένα ηλεκτρονικό κατάστημα μέσω του κινητού μας τηλεφώνου και την αυθεντικοποίηση και πληρωμή μέσω μιας έξυπνης κάρτας που βρίσκεται μέσα σε έναν αναγνώστη καρτών, συνδεδεμένο με το κινητό με κάποιο τρόπο (ενσύρματο, Bluetooth, υπέρυθρες). Ακόμα καλύτερα μπορούμε να σκεφτούμε το προηγούμενο παράδειγμα με την έξυπνη κάρτα ενσωματωμένη στο κινητό τηλέφωνο. Άλλο πιθανό σενάριο είναι η χρήση της έξυπνης κάρτας ως ένα ασφαλές αποθηκευτικό μέσο, και την απομακρυσμένη χρήση της μέσω ενός ηλεκτρονικού υπολογιστή από το κινητό μας τηλέφωνο. Τα σενάρια χρήσης αυτών των τεχνολογιών είναι ανεξάντλητα και μερικά από αυτά θα είναι πραγματικότητα στο εγγύς μέλλον.

8.2 Περαιτέρω Έρευνα

Η υλοποίηση των συστημάτων που παρουσιάστηκαν σε αυτή την εργασία , είχε περισσότερο στόχο την μελέτη της τεχνικής πλευράς και τους τρόπους χρήσης διαφορετικών τεχνολογιών από κοινού και έτσι δεν λήφθηκαν υπόψη ζητήματα ασφάλειας στη μεταφορά δεδομένων μεταξύ των επιμέρους συστατικών. Έτσι θα ήταν χρήσιμο βάση των μεθόδων που προσφέρονται από τα J2SE , J2ME και Java Card να υλοποιηθεί η ασφαλής μεταφορά δεδομένων με τη βοήθεια ανταλλαγής κλειδιών και έτσι να έχουμε ένα σύστημα ασφαλές από κάθε άποψη.

Με την κυκλοφορία του JSR (Java Specification Request) 177 που περιέχει το Security And Trust Services API SATSA, και το οποίο είναι ακόμα υπό ανάπτυξη , θα ανοίξει ο δρόμος για την απευθείας επικοινωνία συσκευών που χρησιμοποιούν J2ME και έξυπνων καρτών με την τεχνολογία Java Card. Έτσι θα μπορέσουν να αναπτυχθούν πολύ ενδιαφέρουσες εφαρμογές ακόμα και να ενσωματωθούν μέσα στη κάρτα SIM ενός κινητού τηλεφώνου κάποια Java Card applets.

ΑΝΑΦΟΡΕΣ

1. "Java 2 Platform, Micro Edition Datasheet" , Sun Microsystems

2. "JavaCard and OpenCard framework : A Tutorial" , Fodor O. and Hassler V.

3. "An Introduction to Java Card Technology" , C. Enrique Ortiz

4. "Security And Trust Services API" , Java Community Process

5. "OpenCard Framework General Information Web Document" , IBM

6. "OpenCard Framework 1.2 Programmer's Guide" , IBM

7. "How to write a Java Card applet: A developer's guide" , Zhiqun Chen

8. "Understanding Java Card 2.0" , Zhiqun Chen

9. "Development Kit User's Guide" , Sun Microsystems

10. "User's Guide, Wireless Toolkit Version 2.0 , Java 2 Platform Micro Edition", Sun Microsystems

ΠΑΡΑΡΤΗΜΑ - Συνοδευτικό CD

Στο CD που συνοδεύει την διπλωματική εργασία, υπάρχει ο κώδικας των εφαρμογών, μαζί με επεξηγηματικά σχόλια, η διπλωματική εργασία σε ηλεκτρονική μορφή καθώς και τα βασικότερα συστατικά για να λειτουργήσουν τα συστήματα που υλοποιήθηκαν.

Πιο συγκεκριμένα, στον φάκελο 'wallet Java Card, OCF,Client, Javacard-Mobile Gateway' υπάρχει ο φάκελος 'wallet', ο οποίος περιέχει τον κώδικα αλλά και έτοιμες τις κλάσεις για το wallet Java Card (Wallet.java), το OpenCard Framework για την wallet Java Card (walletCardServiceFactory.java, walletCardService.java, opencard.properties) , την τερματική εφαρμογή για PC (wClient.java) και τον server ο οποίος σε συνδυασμό με το OpenCard Framework αποτελούν το Java Card-Mobile device Gateway (server.java, serverthread.java). Επίσης στον φάκελο 'wallet' υπάρχει ο φάκελος 'javacard' ο οποίος περιέχει τα αρχεία CAP και EXP για την wallet Java Card (wallet.cap, wallet.exp) και το αρχείο που περιέχει τις APDU εντολές για την εγκατάσταση της wallet Java Card σε μία κάρτα (wallet.scr).

Στον φάκελο 'Mobile Client' υπάρχει ο φάκελος 'JavaCardClient' ο οποίος περιέχει την J2ME εφαρμογή και πιο συγκεκριμένα στον υποφάκελο 'src' περιέχει τον κώδικα των κλάσεων που χρησιμοποιήθηκαν, στον 'classes' τις κλάσεις, και στον 'bin' τα απαραίτητα αρχεία για την εγκατάσταση της εφαρμογής σε μία κινητή συσκευή (JavaCardClient.jad, JavaCardClient.jar).

Στον φάκελο 'Library' υπάρχει η έκδοση 1.4 της Java, το Java Card Development Kit, το Wireless Toolkit και οι βασικές κλάσεις του OpenCard Framework.

Τέλος, στον φάκελο 'Thesis' υπάρχει η διπλωματική εργασία σε ηλεκτρονική μορφή.