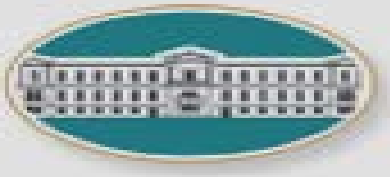
ΤΕΙ ΑΡΤΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

20 ΣΕΠΤΕΜΒΡΙΟΥ

ΑΝΑΠΤΥΞΗ ΤΡΑΠΕΖΙΚΟΥ ΛΟΓΙΣΜΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ

της Ευθυμίας-Σουλτάνας Αρζόγλου

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ :ΑΝΤΩΝΙΑΔΗΣ ΝΙΚΟΛΑΟΣ



EFSI – BANK

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πίνακας Περιεχομένων

Κεφάλαιο 1 : Εισαγωγή

1.1Αντικείμενο διπλωματικής.

Η διπλωματική αυτή εργασία θα ασχοληθεί με την ανάπτυξη ενός **σύγχρονου και ευέλικτου τραπεζικού λογισμικού προγράμματος.**

Οι τράπεζες τόσο διεθνώς όσο και στην Ελλάδα έχουν επιδοθεί σε ένα αγώνα ανάπτυξης πρωτοβουλιών στον τεχνολογικό τομέα ,προκειμένου να τονώσουν την ανταγωνιστικότητα τους στην απαιτητική αγορά.

Δεν πρόκειται φυσικά για ‘μόδα ‘.Ανταγωνισμός και τεχνολογία σχετίζονται άμεσα . Δεν υπάρχει αποτελεσματικός ανταγωνισμός εντός και εκτός συνόρων χωρίς την υποστήριξή της τεχνολογίας.

Η τάση που κυριαρχεί στις επενδύσεις πληροφορικής σήμερα οδηγεί στον προσανατολισμό και την μετατόπιση του κέντρου βάρους σε νέες μεθόδους ,στην δημιουργία λογισμικών εφαρμογών(application software),στην κατάρτιση και στην στελέχωση των τραπεζών με εξειδικευμένο προσωπικό.

Η τεχνολογία μετασχηματίζει την τραπεζική βιομηχανία και της δίνει καινούργιες διαστάσεις . Πρόσβαση σε νέες αγορές ,νέα κανάλια διανομής προϊόντων ,εξάλειψη των προβλημάτων που δημιουργούνται από τις γεωγραφικές αποστάσεις.

Ο κύριος λόγος προσανατολισμού των τραπεζών προς την τεχνολογία, είναι και η σημαντική μείωση του κόστους. Οι τράπεζες που επενδύουν στην τεχνολογία έχουν σημαντικά οικονομικά οφέλη και ένα ανταγωνιστικό πλεονέκτημα στην συνεχώς εξελισσόμενη απαιτητική αγορά.

1.2 Οργάνωση τόμου:

Ποια στάδια θα ακολουθήσουμε για την ανάπτυξη του λογισμικού.

Στάδιο 1:Με την βοήθεια ερωτηματολογίων θα εντοπίσουμε τις αδυναμίες που υπάρχουν αυτή τη στιγμή στα είδη υπάρχοντα τραπεζικά συστήματα.

Στάδιο 2:Στην συνέχεια θα θέσουμε τους στόχους, που θα μας καθοδηγούν σε όλη την διάρκεια ανάπτυξης του λογισμικού.

Στάδιο 3:Ακολούθως θα θέσουμε τις απαιτήσεις για την ανάπτυξη του τραπεζικού μας συστήματος.

Στάδιο 4:Αφού θέσουμε τις απαιτήσεις, θα ξεκινήσουμε την ανάλυση και την σχεδίαση του λογισμικού μας προγράμματος.

Στάδιο 5:Στην συνέχεια έχει σειρά η ανάπτυξη .Θα επιλέξουμε τις πλατφόρμες και τα προγραμματιστικά εργαλεία που πρόκειται να χρησιμοποιήσουμε.

Στάδιο 6:Αφου γίνει η υλοποίηση έχει σειρά ο έλεγχος ,ώστε να διαπιστώσουμε αν το σύστημα μας πληρεί όλες τις προϋποθέσεις

Στάδιο 7:Μελλοντικές προεκτάσεις του λογισμικού μας προγράμματος.

1.3 Στόχοι:

Οι στόχοι ,θα αποτελούν τις βασικές αρχές , που θα ακολουθήσουμε σε όλη την διάρκεια ανάπτυξης του συστήματος. Για κάθε επιλογή που πρόκειται να πάρουμε θα προσπαθήσουμε να οδηγηθούμε στην υλοποίηση των παρακάτω στόχων.

Βασικός στόχος:

Ανάπτυξη ενός σύγχρονού και ευέλικτου τραπεζικού συστήματος.

Επιμέρους στόχοι:

Ασφάλεια: Το σύστημα πρέπει να διασφαλίζει ότι πρόσβαση θα έχουν μόνο οι χρήστες που έχουν δικαιώματα.

Αξιοπιστία: Να διασφαλίζει στους χρήστες του συστήματος ότι δεν θα παρουσιάζει προβλήματα.

Ταχύτητα: Το σύστημα να ολοκληρώνει όλες τις συναλλαγές με την απαιτούμενη ταχύτητα.

Ευελιξία: Το σύστημα μας πρέπει να έχει την δυνατότητα να προσαρμόζεται εύκολα και γρήγορα στις εξελίξεις της τεχνολογίας.

Κόστος: Το κόστος του συστήματος μας πρέπει να είναι χαμηλό.

Κεφάλαιο 2: Απαιτήσεις

Οι απαιτήσεις δίνουν τα πρώτα χαρακτηριστικά στο πρόγραμμα μας .Είναι το πρώτο στάδιο της υλοποίησης .Δίνουμε τις προϋποθέσεις ώστε στη συνέχεια η σχεδίαση να γίνει πιο απλή.

2.1 Απαιτήσεις :

Γενικές απαιτήσεις :

Γραφική διεπαφή χρήστη:	Η γραφική διεπαφή χρήστη (Graphical User Interface) προσφέρει στον χρήστη μια εύκολη στη χρήση διασύνδεση.Με γραφική διεπαφή (GUI),οι χρήστες δεν έχουν να κάνουν τίποτα περισσότερο από το να <<σημειώσουν και να επιλέξουν >>για να κάνουν την δουλεία τους. Οι χρήστες μπορούν να αλληλεπιδράσουν με γραφικές απεικονίσεις γρηγορότερα και ευκολότερα από το έχουν να αντιμετωπίσουν μόνο κείμενο.
Εγχειρίδιο:	Δημιουργία ενός εγχειριδίου ,το οποίο θα μας καθοδηγεί κατά την διάρκεια χρήσης του λογισμικού .Συγκεκριμένα θα μας καθοδηγεί και θα λύνει όλα τα προβλήματα που πιθανόν να αντιμετωπίσουμε κατά την χρήση του.

Απαιτήσεις Ασφαλείας:

Πρόσβαση μόνο χρήστες :	Στο σύστημα μας θα έχουν τη δυνατότητα πρόσβασής μόνο οι χρήστες του συστήματος. Κάθε χρήστης θα έχει το δικό του όνομα και κωδικό πρόσβασης ,τα οποία θα είναι μοναδικά για κάθε χρήστη.
Δημιουργία χρηστών μόνο από διαχειριστές:	Το δικαίωμα δημιουργίας νέων χρηστών θα έχει μόνο ο διαχειριστής του συστήματος. Ο διαχειριστής του συστήματος θα έχει μοναδικό κωδικό πρόσβασης που θα του επιτρέπει να έχει πρόσβαση σε αυτές τις λειτουργίες.

Απαιτήσεις συστήματος:

Δημιουργία χρηστών :	Δημιουργία χρηστών με δικό τους όνομα και κωδικό πρόσβασης.
Καταχώρηση πελατών:	Δυνατότητα καταχώρησης των προσωπικών στοιχείων των πελατών.
Δημιουργία λογαριασμών:	Δημιουργία σε κάθε πελάτη δύο τύπων λογαριασμών τρεχούμενο και ταμιευτηρίου.

Αναζήτηση :	Αναζήτηση των προσωπικών στοιχείων του κάθε πελάτη.
Συναλλαγές:	Μπορούν να υλοποιηθούν οι εξής συναλλαγές: ➢ Επιλογή τύπου λογαριασμού: Ταμιευτηρίου-Τρεχούμενου. ➢ Κατάθεση χρημάτων ➢ Ανάληψη χρημάτων ➢ Εμφάνιση συνολικού υπολοίπου ➢ Μεταφορά χρημάτων.

<u>Απαιτήσεις,χρηστών,διαχειριστών,πελατών.</u>

Χρήστες :	Η χρηστές θα είναι οι κύριοι διαχειριστές του συστήματος. Θα έχουν την δυνατότητα να χρησιμοποιούν σχεδόν όλες τις δυνατότητες του.
Διαχειριστές:	Οι διαχειριστές του συστήματός θα έχουν την δυνατότητα πρόσβασης σε όλες τις λειτουργίες του συστήματος.
Πελάτες :	Οι πελάτες θα μπορούν να εξυπηρετούνται μόνο μέσω της συνεργασίας του με τους χρήστες.

Κεφάλαιο 3 : Αρχιτεκτονική του συστήματος Ανάλυση και Σχεδίαση

3.1 Ορισμός της αρχιτεκτονικής

Αρχιτεκτονική μας δίνει μια σφαιρική άποψη για την εφαρμογή την οποία θα εφαρμόσουμε .Περιγράφει στρατηγικές επιλογές ,οι οποίες προσδιορίζουν την ποιότητα και την αξιοπιστία ,την προσαρμοστικότητα και την εγγύηση της καλής λειτουργίας της εφαρμογής καθώς επίσης πρέπει να αφήνει περιθώρια για παρεμβάσεις ώστε να ικανοποιήσει τις ανάγκες που θα προκύψουν στην φάση της ανάπτυξης.

Η αρχιτεκτονική μας θα πρέπει να χαρακτηρίζεται από :

- Απλότητα
- Ευκολία στην κατανόηση
- Περιορισμός γενικεύσεων
- Σαφή διαχωρισμό ανάμεσα στις προδιαγραφές και στην υλοποίηση κάθε επιπέδου.

3.2 Ανάλυση και Σχεδίαση του συστήματος μας:

Εδώ θα κάνουμε μια αναλυτική περιγραφή των δυνατοτήτων του συστήματος μας :

Αρχικά θα δημιουργήσουμε μια κεντρική φόρμα οι οποία θα μας δίνει τις εξής επιλογές :

- Θα μας εισάγει στο κεντρικό σύστημα.

  Κάθε χρήστης για να έχει πρόσβαση στο σύστημα ,θα δίνει το όνομα και τον κωδικό του. Ο κωδικός θα είναι μοναδικός και μυστικός για κάθε χρήστη.

- Θα μας δίνει την δυνατότητα δημιουργίας νέων χρηστών.

  Την δυνατότητα δημιουργίας νέων χρηστών δεν θα έχουν όλοι οι χρήστες .Μόνο οι διαχειριστές του συστήματος,θα έχουν την δυνατότητα δημιουργίας νέων χρηστών.

Αφού εισαχθούμε στο σύστημα θα έχουμε τις εξής επιλογές:

- Καταχώρηση πελατών
- Δημιουργία λογαριασμών για κάθε πελάτη.
- Αναζήτηση των στοιχείων των πελατών και τον αριθμό ταυτότητας κάθε τύπου λογαριασμού.
- Συναλλαγές.

Στην πρώτη επιλογή ,θα έχουμε την δυνατότητα καταχώρησης όλων των προσωπικών στοιχείων για κάθε πελάτη ,δηλαδή όνομα ,επίθετο ,διεύθυνση ,τηλέφωνο ,αριθμό ταυτότητας .

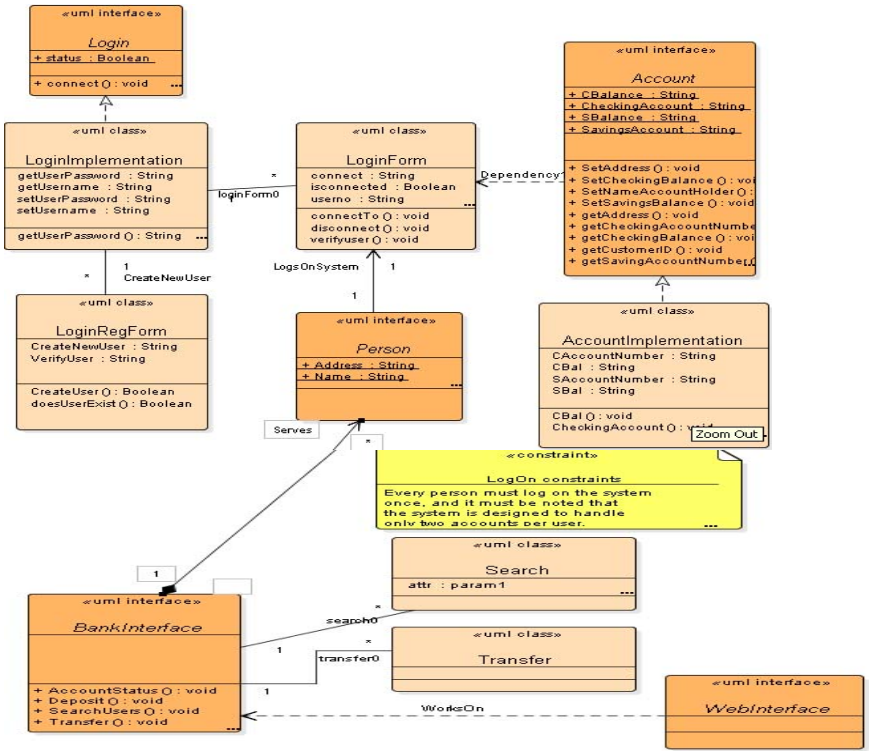
Στην δεύτερη επιλογή ,θα έχουμε την δυνατότητα δημιουργίας λογαριασμών .Κάθε πελάτης θα έχει την δυνατότητα να έχει δύο τύπους λογαριασμών ,τρεχούμενου και ταμιευτηρίου.

Στη τρίτη επιλογή ,Θα έχουμε την δυνατότητα αναζήτησης όλων των προσωπικών στοιχείων των πελατών και τον αριθμό ταυτότητας κάθε λογαριασμού .Η αναζήτηση θα γίνεται μέσω της μοναδικής ταυτότητας κάθε πελάτη .

Στην τέταρτη επιλογή ,κάθε χρήστης θα μπορεί να διεκπαιρεώσει τις εξής επιλογές

- Επιλογή λογαριασμού Τρεχούμενου ,Ταμιευτηρίου.
- Ανάληψη χρημάτων .
- Κατάθεση χρημάτων
- Εμφάνιση συνολικού ποσού κάθε λογαριασμού.
- Μεταφορά χρημάτων από τον ένα λογαριασμό στον άλλο .Ο χρήστης θα έχει την δυνατότητα να μεταφέρει χρήματα από το λογαριασμό του ενός πελάτη στο λογαριασμό του άλλου πελάτη, ανεξάρτητα από τον τύπο του λογαριασμού.

Σχήμα 1:Περιγραφή του συστήματός σε UML.



Ανάλυση του UML σχεδιαγράμματος.

Login :Είναι <<UML interface>>, η τάξη αυτή δεν έχει υλοποίηση ,η υλοποίηση της γίνεται από την τάξη <<UML class>>,η οποία είναι το LoginImplementation.

LoginImplementation:Είναι ένα <<UML class>>,το οποίο πραγματοποιεί την σύνδεση ,η οποία ορίζεται στο Login.

LoginRegForm : Είναι ένα <<UML class>> , η τάξη αυτή ,δημιουργεί νέους χρήστες και ελέγχει αν είδη υπάρχει καταχωρημένος ο καινούργιος χρήστης.

LoginForm: Είναι ένα <<UML class>>,η τάξη αυτή επιβεβαιώνει αν ο κωδικός είναι σωστός .Αν είναι , γίνεται η σύνδεση αλλιώς γίνεται αποσύνδεση.

Person: Είναι ένα <<UML interface>>,η τάξη αυτή αντιπροσωπεύει το χρήστη που συνδέεται στο σύστημα.

Account :Είναι ένα <<UML interface>>,η υλοποίηση του γίνεται από το AccountImplementation .

AccountImplementation:Είναι ένα <<UML class>>,η τάξη αυτή, δημιουργεί δύο τύπους λογαριασμών ,τρεχούμενο και ταμιευτηρίου. Καταχωρεί τα στοιχεία των πελατών και δείχνει στον χρήστη σε ποιον πελάτη ανήκει κάθε λογαριασμός.

BankInterface : Είναι ένα <<UML interface>> , η τάξη αυτή δεν έχει υλοποίηση ,οι υλοποιήσεις της τάξης αυτής, πραγματοποιούνται από τα εξής <<UML class>>,οι οποίες είναι, η αναζήτηση, η μεταφορά ,WebInterface.

Search <<UML class>>, πραγματοποιεί την αναζήτηση των στοιχείων των πελατών.

Transfer <<UML class>>, πραγματοποιεί την μεταφορά χρημάτων από τον ένα λογαριασμό στον άλλο.

Το webInterface <<UML class>>

Συσχετίσεις:

LoginImplementation-LoginForm: Σχέση πολλαπλότητας, πολλοί χρήστες μπορούν να έχουν πρόσβαση στο σύστημα**.**

LoginImplementation-LoginRegForm: Σχέση ένα προς πολλά .Μόνο ο διαχειριστής του συστήματος θα έχει τη δυνατότητα δημιουργίας νέων χρηστών.

LoginForm- Account: Σχέση εξάρτησης, για να υλοποιηθεί η τάξη account θα πρέπει πρώτα να υλοποιηθεί η τάξη LoginForrm.

LoginForm-Person: Σχέση ένα προς πολλά, κάθε χρήστης θα μπορεί να ενωθεί μόνο μια φορά στο σύστημα.

BankInterface-Person: Σχέση σύνθεσης (composition), δηλαδή η τάξη BankInterface δεν μπορεί να έχει υλοποίηση αν προηγουμένως δεν έχουν υλοποιηθεί οι τάξεις στις οποίες ανήκει. Επίσης υπάρχει σχέση ένα προς πολλά, κάθε τραπεζικό σύστημα μπορεί να εξυπηρετεί πολλούς χρήστες ταυτόχρονα.

BankInterface- Search: Σχέση ένα προς πολλά, κάθε τραπεζικό σύστημα θα έχει την δυνατότητα να υλοποιεί πολλές αναζητήσεις.

BankInterface-Transfer: Σχέση ένα προς πολλά, κάθε τραπεζικό σύστημα θα έχει την δυνατότητα να υλοποιεί πολλές μεταφορές χρημάτων.

3.3 Ανάλυση των Βάσεων δεδομένων :

Αρχικά έχουμε τον πίνακα συναλλαγών(current) .Στον πίνακα αυτό αποθηκεύονται όλες οι συναλλαγές των πελατών ,καταθέσεις(deposit),αναλήψεις(withdraw),μεταφορά από (Transfer from),μεταφορά σε (Transfer to),ποσό μεταφοράς(Transfer Amount) και συνολικό ποσό κάθε λογαριασμού (Balance). Το πρωτεύον κλειδί(primary key) σε αυτό τον πίνακα είναι το CurrentID και το ξένο κλειδί είναι το CustomerID.

Current

CurrentID	CustomerID	Deposit	Withdraw	Transfer from	Transfer To	Amt	Bal	CASA

Primary Key ***Foreign Key***

Στην συνέχεια έχουμε τον πίνακα των πελατών(customers).Σε αυτόν τον πίνακα αποθηκεύονται τα στοιχεία των πελατών ,όνομα ,επίθετο ,διεύθυνση και το αριθμό ταυτότητας κάθε λογαριασμού. Το πρωτεύον κλειδί(primary key) σε αυτόν το πίνακα είναι το CustomerID.

Customers

CustomerID	Name	Surname	Address	Telephone	CurrentAcNo	SavingsAcNo

Primary Key

Και τέλος έχουμε τον πίνακα των χρηστών (Login).Σε αυτόν τον πίνακα αποθηκεύονται τα στοιχεία των χρηστών ,το όνομα (user) και ο κωδικός(password).Το πρωτεύον κλειδί (primary key) είναι το password.

Login

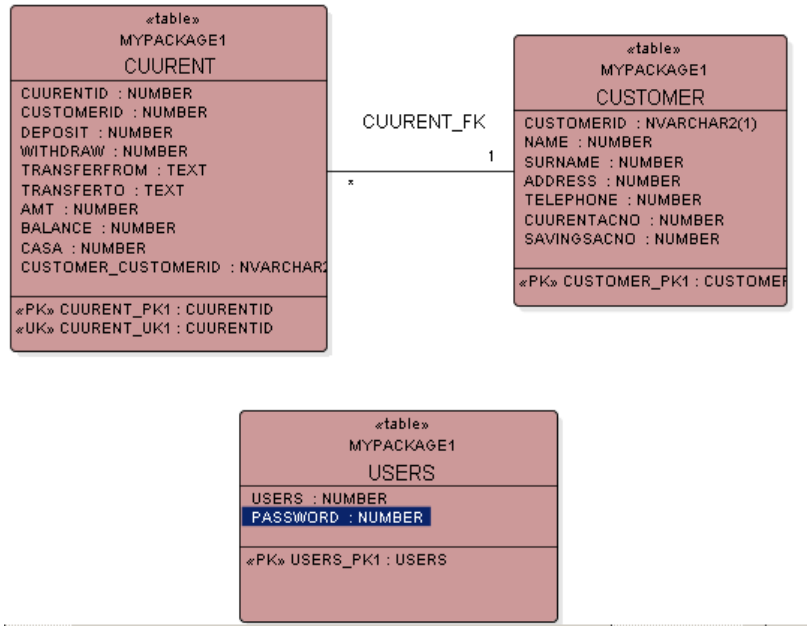
User	Password

Primary Key

Συσχέτιση πινάκων :

Υπάρχει μια συσχέτιση μεταξύ των πινάκων current και customer.Η σχέση είναι ένα προς πολλά ,κάθε πελάτης έχει την δυνατότητα να δημιουργήσει πολλούς λογαριασμούς. Η σύνδεση των δύο πινάκων γίνεται με την βοήθεια του ξένου κλειδιού (foreign key), το CustomerID

Σχήμα :Βάσης.


«table»
MYPACKAGE1
CUURENT
CUURENTID : NUMBER
CUSTOMERID : NUMBER
DEPOSIT : NUMBER
WITHDRAW : NUMBER
TRANSFERFROM : TEXT
TRANSFERTO : TEXT
AMT : NUMBER
BALANCE : NUMBER
CASA : NUMBER
CUSTOMER_CUSTOMERID : NVARCHAR2
«PK» CUURENT_PK1 : CUURENTID
«UK» CUURENT_UK1 : CUURENTID
CUURENT_FK
1
*
«table»
MYPACKAGE1
CUSTOMER
CUSTOMERID : NVARCHAR2(1)
NAME : NUMBER
SURNAME : NUMBER
ADDRESS : NUMBER
TELEPHONE : NUMBER
CUURENTACNO : NUMBER
SAVINGSACNO : NUMBER
«PK» CUSTOMER_PK1 : CUSTOMER
«table»
MYPACKAGE1
USERS
USERS : NUMBER
PASSWORD : NUMBER
«PK» USERS_PK1 : USERS

3.4 Αρχιτεκτονική του συστήματος(προέκταση)

Η αρχιτεκτονική ,η οποία θα διαλέξουμε για επικοινωνία των υποκαταστημάτων μιας τράπεζας είναι η web-based.Η αρχιτεκτονική αυτή ανήκει στην κατηγορία των Client-Server. Δίνει την δυνατότητα στους χρήστες να χρησιμοποιούν ταυτόχρονα την ίδια εφαρμογή. Η αρχιτεκτονική αυτή, είναι πλέον ιδιαίτερα διαδεδομένη και οικεία, καθώς το internet έχει μπει πια για τα καλά στην ζωή μας, είναι ένα από τα πιο διαδεδομένα μέσα επικοινωνίας.

Τα πλεονεκτήματα που μας προσφέρει αυτή η εφαρμογή είναι:

- Χαμηλό κόστος ,καθώς δεν είμαστε υποχρεωμένοι να κάνουμε εγκατάσταση της εφαρμογής σε κάθε τερματικό ξεχωριστά.
- Τα τερματικά που θα χρησιμοποιήσουμε δεν χρειάζεται να έχουν κάποια ιδιαίτερα χαρακτηριστικά ,αρκεί να είναι εξοπλισμένα με ένα web-browser.
- Ασφάλεια από εξωτερικούς χρήστες ,καθώς η σύνδεση των client με τους server θα γίνεται ενός μέσω ενός ιδιωτικού δικτύου τύπου WAN (Wide Area Network).

Web – based:

- Στη αρχιτεκτονική web-based θα έχουμε ένα client ,ο οποίος θα διαθέτει ένα browser , παραδείγματός χάρη τον (internet explorer).

- Ο browser θα μας συνδέει στο σύστημα μας.

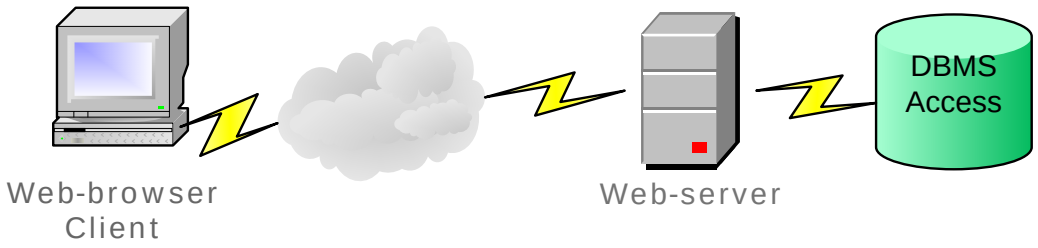
Το σύστημά μας αποτελείται από :

- ✓ Το web-server στον οποίο θα έχουμε εγκαταστημένη την εφαρμογή μας

- ✓ Το Database server στο οποίο θα είναι εγκαταστημένη η βάση δεδομένων. Οι server βάσεων δεδομένων τρέχουν DBMS(Database Management system) λογισμικό .Τα DBMS προσφέρουν εξειδικευμένες υπηρεσίες :την δυνατότητα να ανακτά πληροφορίες και να διαχειρίζεται πληροφορίες .

- ➢ Η σύνδεση των client με τους server θα γίνεται μέσω ενός ιδιωτικού δικτύου ευρείας περιοχής το WAN (Wide Area Network).

Σχήμα: web-based



Κεφάλαιο 4 : Εργαλεία του συστήματος

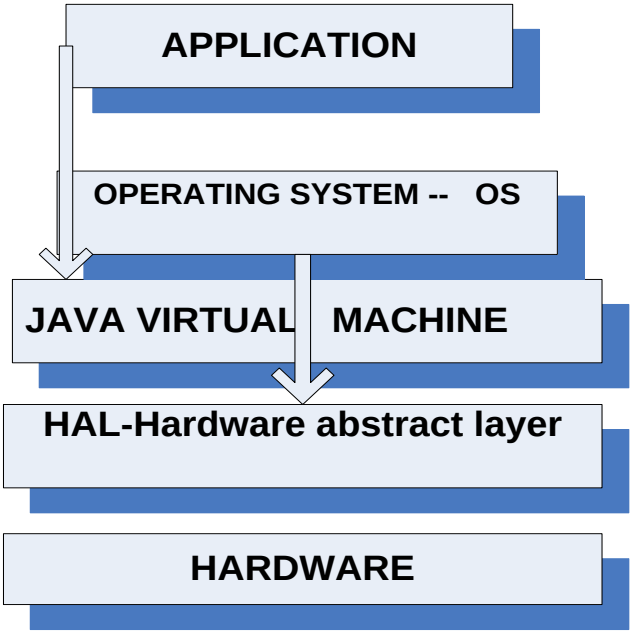
Τι εργαλεία χρησιμοποιούμε για την υλοποίηση του τραπεζικού μας συστήματος:

4.1 Γλώσσα προγραμματισμού java

Η γλώσσα που επιλέγουμε για την ανάπτυξη του συστήματος είναι η JAVA**.**Η Java πρωτοπαρουσιάστηκε από την Sun Microsystems το 1995. Πρόκειται για μια γλώσσα προγραμματισμού, ειδικά σχεδιασμένη για χρήση σε κατανεμημένα περιβάλλοντα, όπως το Internet. Δημιουργήθηκε με την προοπτική να είναι απλή και εύκολη στη χρήση, επιβάλλοντας μια ολοκληρωτικά αντικειμενοστραφή αντιμετώπιση. Συμπαγής και αξιόπιστα τα προγράμματα της Java μπορούν να «μεταφερθούν» σε οποιαδήποτε συσκευή είτε ασύρματα, είτε μέσω του Διαδικτύου.

Η Java έχει την ικανότητα ανεξαρτησίας μηχανής. Ο μεταγλωττιστής της Java δεν παράγει εκτελέσιμο κώδικα αλλά παράγει μια μορφή αρχείων γνωστή ως Java Bytecodes που δεν είναι απευθείας εκτελέσιμη από μια μηχανή αλλά εκτελείται από μια εικονική μηχανή Java(Java Virtual Machine—JVM).

Αυτό δίνει στη Java το πλεονέκτημα της ανεξαρτησίας από την συγκεκριμένη μηχανή στην οποία εκτελείται .Αρκεί κανείς να έχει εγκαταστήσει την κατάλληλη έκδοση της Java και το πρόγραμμα θα εκτελεστεί από την εικονική μηχανή .Αυτή η επινόηση έμεινε γνωστή σαν <<Γράψε μια φορά για να τρέχει παντού >>(Write Once-RunEverywhere).



Λόγοι για να χρησιμοποιήσουμε τη Java

- **Η Java είναι σχεδιασμένη για το Internet**. Προγράμματα της Java μπορούν να τρέχουν σε προγράμματα πλοήγησης εφοδιασμένα με την κατάλληλη JVM. Αφού το πρόγραμμα κατέβει από το Internet εκτελείται στη μηχανή του πελάτη και όχι στο Server. Αυτά τα προγράμματα της Java είναι γνωστά σαν Java Applets.

- **Η Java είναι σχεδιασμένη με στόχο την εξασφάλιση ασφάλειας.** Ένα Java Applet εκτελείται με απόλυτη ασφάλεια στη μηχανή του χρήστη και δεν μπορεί να κάνει τίποτα που να μην επιτρέπεται από την πολιτική ασφάλειας της JVM της μηχανής του πελάτη.
- **Η Java προσφέρει εκτέλεση σε όλους τους υπολογιστές.** Με το επίπεδο αφαίρεσης υλικού της JVM ένα πρόγραμμα Java εκτελείται σε όλους τους υπολογιστές ανεξαρτήτως Λειτουργικού Συστήματος. Αρκεί αυτές να είναι εφοδιασμένες με την κατάλληλη JVM (Java Virtual Machine)

- **Η Java είναι μια πλούσια γλώσσα προγραμματισμού.** Η Java είναι πολύ ισχυρή αντικειμενοστραφής γλώσσα προγραμματισμού για το Internet (και όχι μόνο) και όχι μία απλή script γλώσσα για την σύνθεση συστατικών (π.χ. VBScript με ActiveX controls). Αυτό μας επιτρέπει να γράφουμε προγράμματα τα οποία είναι εύκολο να συντηρηθούν και να εξελιχτούν στο χρόνο και όχι απλά σενάρια (scripts) τα οποία παίζουν το ρόλο της κόλλας για components που έχουν γίνει με άλλες γλώσσες αλλά γενικά δεν μπορούμε να τα αλλάξουμε ή να ελέγξουμε την εξέλιξή τους.
- **Η Java προσφέρει την ευελιξία της δυναμικής σύνδεσης.** Ένα πρόγραμμα Java εκτελείται από μία εικονική μηχανή και οι τάξεις (τα αρχεία .class που περιέχουν τα bytecodes) φορτώνονται από την JVM όταν εκτελείται το πρόγραμμα. Αυτό σημαίνει ότι μπορεί κανείς να αντικαταστήσει μία παλιά τάξη με μία νέα ακόμα και την στιγμή που εκτελείται το πρόγραμμα. Αυτό είναι ένα βασικό χαρακτηριστικό των components. Έτσι οι τάξεις Java πάνε πέρα από τον αντικειμενοστραφή προγραμματισμό, στον προγραμματισμό βασισμένο σε συστατικά (component-based programming). Είναι επίσης σημαντικό ότι η Java είναι σχεδιασμένη γι' αυτό και δεν απαιτείται η προσθήκη κάποιου πολύπλοκου μοντέλου (όπως το COM) της Microsoft, πάνω από μία γλώσσα προγραμματισμού που δεν είναι σχεδιασμένη με αυτό τον τρόπο (π.χ. Visual C++).

Πιθανά μειονεκτήματα της Java

- **Τα προγράμματα πλοήγησης ενδέχεται να μην μπορούν να εκτελέσουν το Applet μας**. Αυτό διότι τα προγράμματα πλοήγησης ενδέχεται να μην είναι εφοδιασμένα με την τρέχουσα έκδοση της Java.
- **Ο κώδικας που είναι μεταγλωττισμένος για μια συγκεκριμένη μηχανή είναι ταχύτερος από τον κώδικα που εκτελείται από ένα μεταφραστή (interpreter**). Πάντως αξίζει να σημειωθεί ότι μία τεχνική που ονομάζεται Just-In Time μεταγλώττιση (JIT compilation) έχει βελτιώσει σημαντικά τους χρόνους εκτέλεσης των Java προγραμμάτων. Αν και ακόμα τα προγράμματα

της Java δεν έχουν γίνει ταχύτερα από τον μεταγλωττισμένο κώδικα της C++, πολλοί είναι αυτοί που υποστηρίζουν ότι είναι αρκετά γρήγορα για να μην γίνει η ταχύτητα η αιτία να μην πετύχει η Java.

4.2 Βοηθητικά Εργαλεία

Το εργαλείο που θα χρησιμοποιήσουμε για την ανάπτυξη της εφαρμογής είναι το jdbc και πως χρησιμοποιείται.

Ο προγραμματισμός εφαρμογών για πρόσβαση σε βάσεις δεδομένων πάντα αποτελούσε για τον developer μία δύσκολη υπόθεση. Υπάρχουν εκατοντάδες προϊόντα (programming interfaces) που επιτρέπουν την πρόσβαση των εφαρμογών στις βάσεις δεδομένων. Το μειονέκτημά τους όμως είναι ότι το καθένα απ' αυτά μιλάει στην εφαρμογή του προγραμματιστή τη δική του γλώσσα. Επομένως κάθε φορά που ο προγραμματιστής επιχειρεί τη σύνδεση της εφαρμογής του με μια βάση διαφορετικού τύπου (π.χ. Sybase, Ingres) είναι αναγκασμένος να γράψει από την αρχή την εφαρμογή του έτσι ώστε αυτή να «μιλάει» ,το καινούριο interface προς την βάση. Κάτι τέτοιο όμως είναι αρκετά κοπιαστικό και αποσπά τον developer από το κυρίως έργο του. Την απάντηση σ' αυτό το πρόβλημα έρχεται να δώσει η Java και μάλιστα με τρόπο επαναστατικό. Πρόκειται για το JDBC API που η Javasoft (της Sun Microsystems) περιέλαβε στα APIs της πλατφόρμας προγραμματισμού Java. Το JDBC API επαληθεύει τον ισχυρισμό της Java : "write once, compile once, run everywhere".

Το JDBC API (Java DataBase Connectivity) είναι ένα από τα ισχυρότερα και πιο ολοκληρωμένα APIs της Java. Παρέχει στον προγραμματιστή τη δυνατότητα να συνδέσει την εφαρμογή του με βάσεις δεδομένων διαφόρων τύπων χωρίς να χρειαστεί να τροποποιήσει το πρόγραμμά του κάθε φορά που συνδέει την εφαρμογή του σε μια διαφορετική βάση. Παρεμβάλλει και ενεργεί ως middleware ανάμεσα στις Java εφαρμογές και τις σχεσιακές (relational) βάσεις. Το JDBC είναι ένα interface για πρόσβαση σε βάσεις δεδομένων (database access interface) που χρησιμοποιεί standard SQL ερωτήσεις. Προσφέρει στον

προγραμματιστή την άνεση να γράφει ένα πρόγραμμα που να στέλνει μια SQL ερώτηση σε μια σχεσιακή βάση δεδομένων είτε πρόκειται για Oracle είτε για Sybase είτε για Informix είτε για Access είτε για οποιαδήποτε άλλη πλατφόρμα. Η μόνη φροντίδα του προγραμματιστή είναι να επιλέγει κάθε φορά που συνδέεται σε μια βάση τον κατάλληλο driver για να καταλαβαίνει το JDBC τι είδους βάση πρόκειται να προσπελάσει ώστε να φορτώνει τις απαραίτητες ρουτίνες.

Επιπλέον το JDBC API περιέχει το JDBC–ODBC bridge (πρόκειται για κοινό προϊόν της Javasoft με την Intersolv). Αφορά ένα σύνολο προγραμμάτων που περιέχει το JDBC API τα οποία επιτρέπουν τη σύνδεση και την επικοινωνία των εφαρμογών που γράφει ο προγραμματιστής με οποιαδήποτε βάση δεδομένων που μπορεί να προσπελαστεί με τον οδηγό ODBC (Open DataBase Connectivity) της Microsoft. Η γέφυρα αυτή προσδίδει στο JDBC τη δυνατότητα να χρησιμοποιήσει ένα μέρος από τις λειτουργίες του ODBC προκειμένου η εφαρμογή να επικοινωνήσει με την βάση που χρησιμοποιεί τον οδηγό ODBC.

4.3 Πως γίνεται η σύνδεση της βάσης με το jdbc;

Το JDBC έχει την ικανότητα να συνδέεται με διάφορες βάσεις ανάλογα με τις ανάγκες του συστήματος .Η διαδικασία σύνδεσης της βάσης με το πρόγραμμα jdbc είναι απλή. Ανάλογα με την βάση που θέλουμε να εγκαταστήσουμε, μπορούμε να επιλέξουμε και τους κατάλληλους drivers .Στην συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε την access.

Η σύνδεση γίνεται ως εξής:

Control panel->administrative tools->Data sources->Add->.Microsoft Access Driver

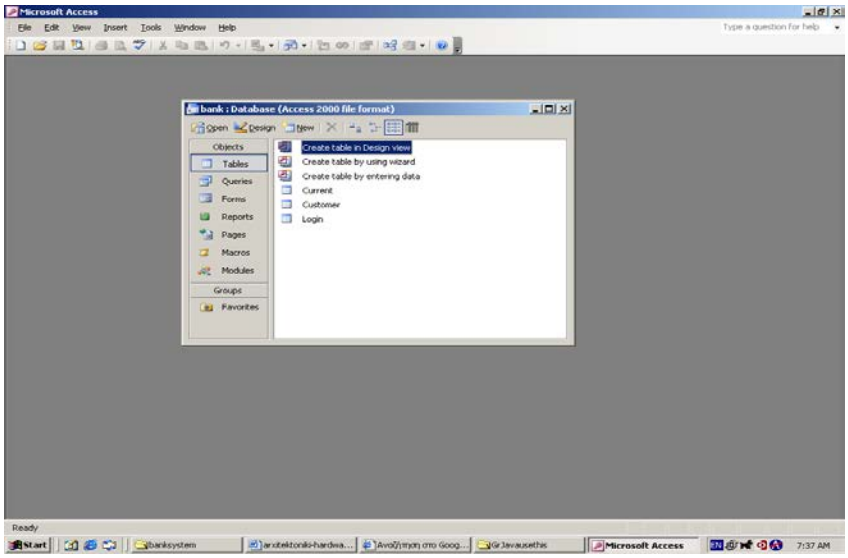
>DataSourceName->bank->select ->My Document.

Το My Document πρέπει να είναι υποθηκευμένο στο δίσκο c:/

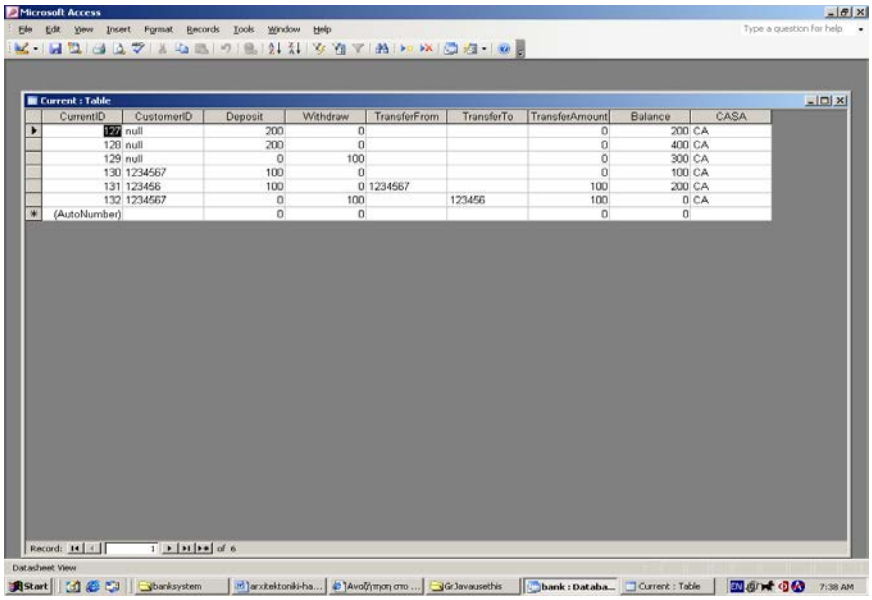
4.4 Access

Μια βάση δεδομένων είναι μια συλλογή πληροφοριών. Αφού δημιουργήσετε μια βάση δεδομένων με πολλές εγγραφές, τότε μπορείτε σε αυτήν να ψάξετε και να βρείτε εύκολα και γρήγορα το/τα στοιχείο/α που επιθυμείτε. Εδώ η βάση δεδομένων που θα μας βοηθήσει να αντλήσουμε τις πληροφορίες για τον τραπεζικό μας σύστημα είναι η access.Ένα από τα πλεονεκτήματα τις ACCESS είναι η δυνατότητα της να τρέχει σε πολλά λειτουργικά όπως Windows 95,Windows 98 ,Windows 2000 και τέλος Windows XP. Με την βοήθεια της access θα οργανώσουμε όλα τα απαραίτητα στοιχεία μιας τράπεζας με σκοπό την καλύτερη οργάνωση και εξυπηρέτηση πελατών μας. Η βάση θα αποτελείται από διάφορους πίνακες, οι οποίοι θα περιέχουν όλα αυτά τα στοιχεία που χρειαζόμαστε ώστε να γίνεται η καλύτερη ταξινόμηση και παρακολούθηση των συναλλαγών των πελατών .

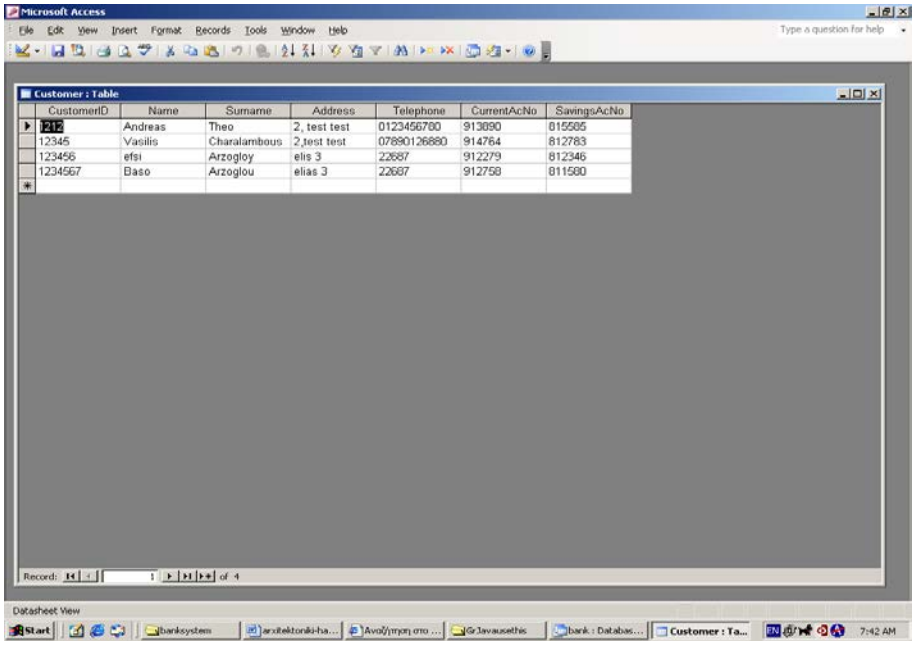
Σχήμα :Πίνακες



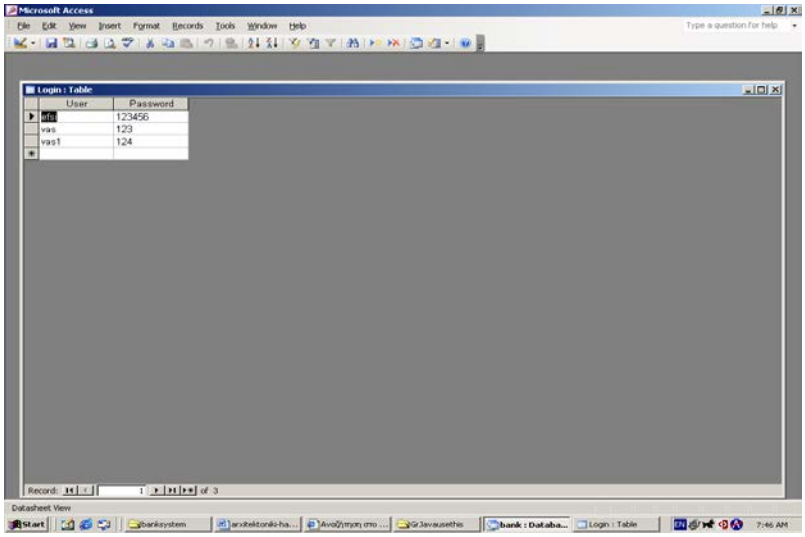
Σχήμα: Πίνακας Συναλλαγών(Current)



Σχήμα :Πίνακας Πελατών



Σχήμα: Πίνακας Σύνδεσης

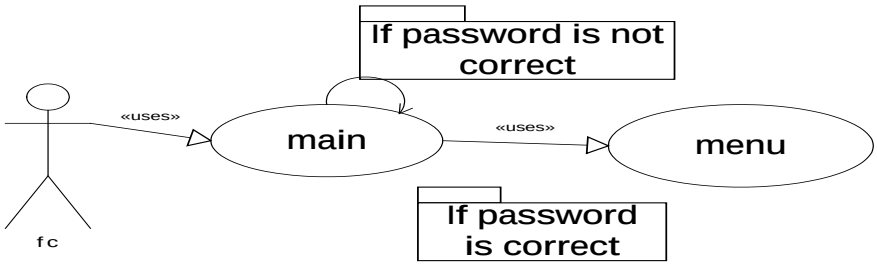


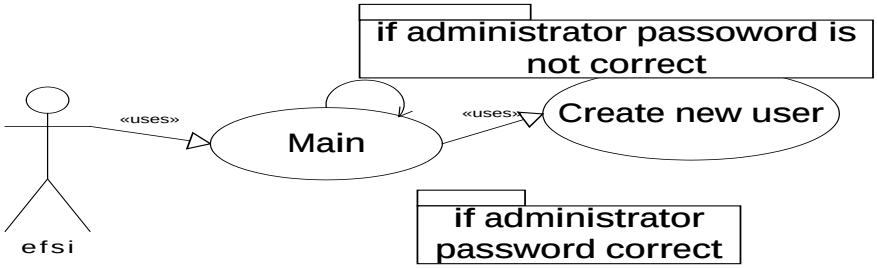
Κεφάλαιο 5 : Έλεγχος και Μετρήσεις .

5.1 **Μοντελοποίηση με περιπτώσεις χρήσης(USE AND CASE)**

Μοντέλο με περιπτώσεις χρήσεις ,είναι μια άποψη του συστήματος που αποδίδει έμφαση στην λειτουργικότητα ενός συστήματος ,όπως αυτή είναι ορατή από τους χρήστες του συστήματος . Με τις περιπτώσεις χρήσης μπορούμε να κάνουμε έλεγχο, μετά την υλοποίηση του λογισμικού ,αν πληρεί όλες τις αρχικές μας προϋποθέσεις

Use and Case 1:



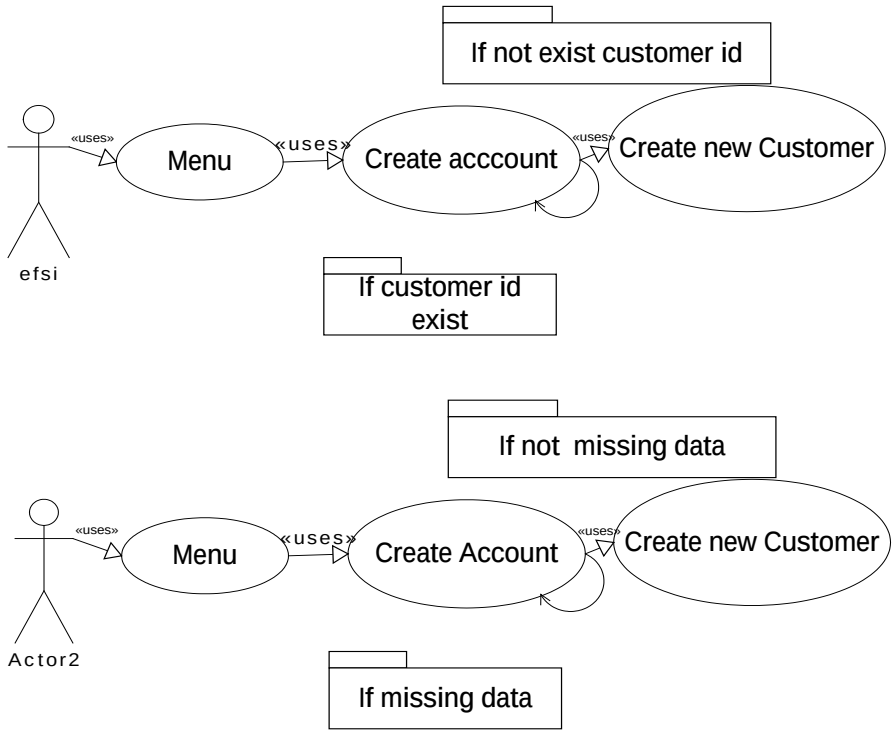


Περιπτώσεις ,Κωδικού(Password)

Ρόλοι :Χρήστης, Βάση δεδομένων χρήστη, σύστημα εισαγωγής στο κεντρικό παράθυρο.
Προ-συνθήκες: Ο χρήστης πληκτρολογεί τον κωδικό.
Βασικό Σενάριο: Το σύστημα ελέγχει αν ο κωδικός είναι σωστός ,
Και εισάγει τον χρήστη στο σύστημα.
Εναλλακτικά σενάρια :Αν ο κωδικός δεν είναι σωστός τότε ,θα
πρέπει να εμφανίζεται ένα μήνυμα προειδοποίησης ότι ο κωδικός είναι λάθος.

Ρόλοι :Χρήστης, Βάση δεδομένων χρήστη, δημιουργία νέου χρήστη
Προ-συνθήκες: Ο διαχειριστής πληκτρολογεί τον κωδικό
Βασικό Σενάριο: Το σύστημα ελέγχει αν ο κωδικός είναι σωστός
Και δημιουργεί νέο χρήστη .
Εναλλακτικά σενάρια :Αν ο κωδικός δεν είναι σωστός θα πρέπει
να εμφανίζεται μήνυμα προειδοποίησης ότι ο κωδικός είναι λάθος.

Use and Case 2



<u>Περιπτώσεις ,Δημιουργία νέου πελάτη(Create Customer)</u>

Ρόλοι :Χρήστης ,Βάση δεδομένων πελάτη, δημιουργία νέου πελάτη.

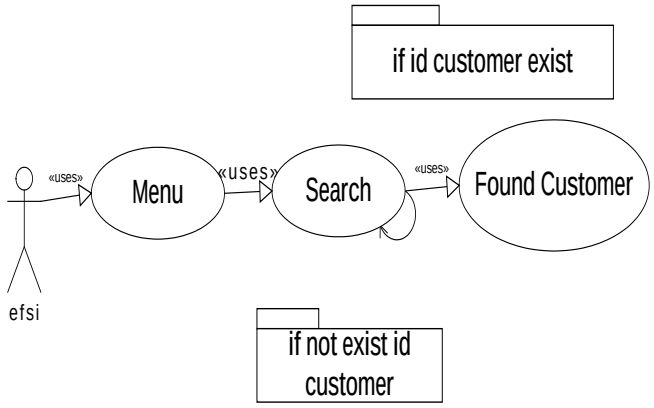
Προ-συνθήκες: Ο χρήστης συμπληρώνει τα στοιχεία του νέου πελάτη .

Βασικό Σενάριο: Το σύστημα ελέγχει αν η ταυτότητα του πελάτη υπάρχει. Αν δεν υπάρχει καταχωρεί τα στοιχεία του πελάτη .

Εναλλακτικά σενάρια :Αν υπάρχει ,θα πρέπει να εμφανίζει μήνυμα προειδοποίησης ότι ο πελάτης έχει ξανακαταχωρηθεί.

Ρόλοι :Χρήστης ,Βάση δεδομένων πελάτη, δημιουργία νέου πελάτη.
Προ-συνθήκες: Ο χρήστης συμπληρώνει τα στοιχεία του νέου πελάτη .
Βασικό Σενάριο: Αν ο χρήστης συμπληρώσει όλα τα στοιχεία τότε γίνεται η καταχώρηση του νέου πελάτη
Εναλλακτικά σενάρια :Αν δεν συμπληρωθούν όλα τα στοιχεία , τότε θα πρέπει να εμφανίζεται μήνυμα προειδοποίησης ότι λείπουν στοιχεία.

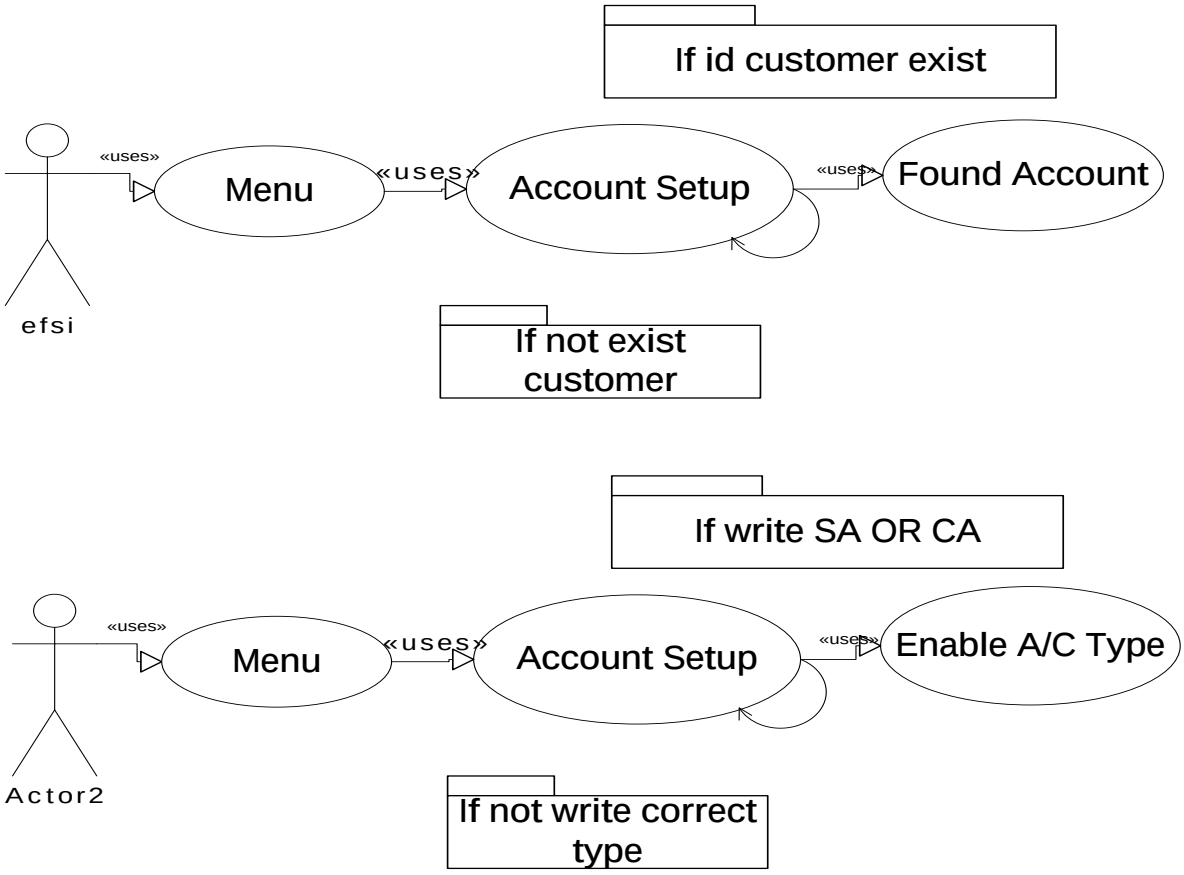
Use and Case 3



<u>Περιπτώσεις ,Αναζήτησης πελάτη(Search)</u>

Ρόλοι :Χρήστης ,Βάση δεδομένων πελάτη ,αναζήτηση πελάτη
Προ-συνθήκες: Ο χρήστης δίνει τον αριθμό ταυτότητας του πελάτη .
Βασικό Σενάριο: Το σύστημα ελέγχει αν η ταυτότητα του πελάτη υπάρχει.
Αν υπάρχει ο αριθμός ταυτότητας ,βρίσκει τα στοιχεία του πελάτη
Εναλλακτικά σενάρια :Αν δεν υπάρχει ο αριθμός ταυτότητας ,θα πρέπει να εμφανίζει μήνυμα προειδοποίησης ότι ο πελάτης αυτός δεν έχει καταχωρηθεί.

Use and Case 4



Περιπτώσεις ,Συναλλαγών(Account Setup)

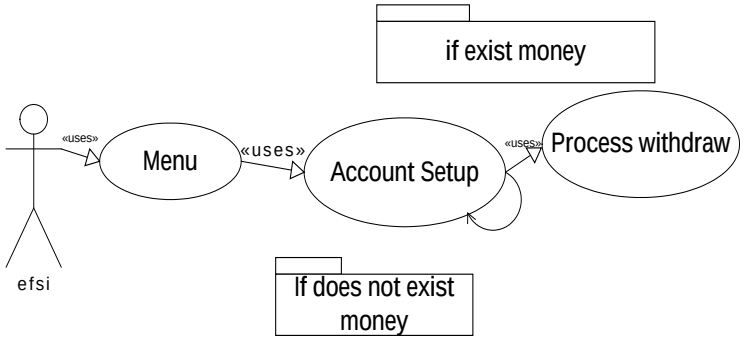
Ρόλοι :Χρήστης Βάση δεδομένων συναλλαγών ,συναλλαγές

Προ-συνθήκες: Ο χρήστης δίνει τον αριθμό ταυτότητας του πελάτη .

Βασικό Σενάριο: Το σύστημα ελέγχει αν η ταυτότητα του πελάτη υπάρχει. Αν υπάρχει ο αριθμός ταυτότητας ,βρίσκει τους αριθμούς ταυτότητας των λογαριασμών

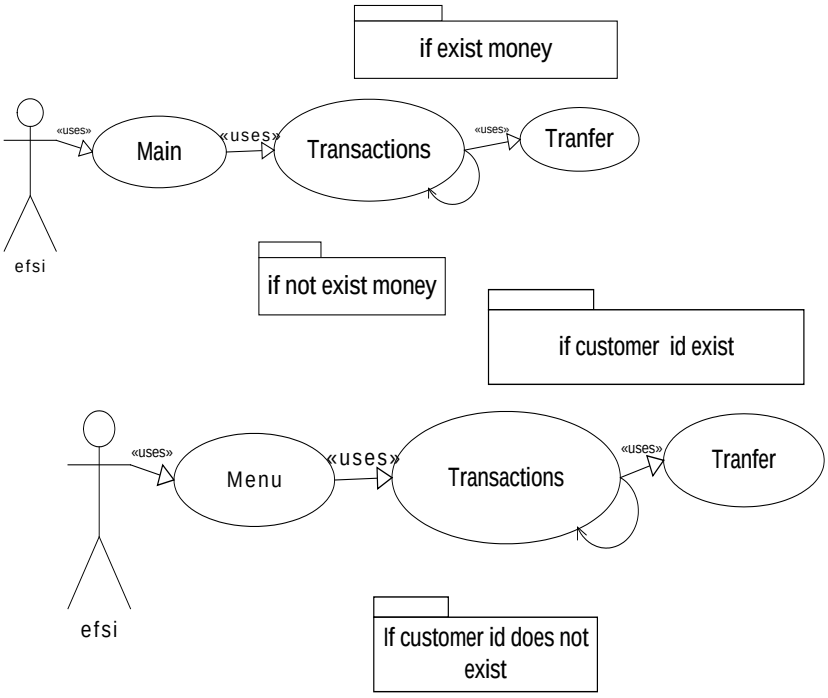
Εναλλακτικά σενάρια: Αν δεν υπάρχει ο αριθμός ταυτότητας ,θα πρέπει να εμφανίζει μήνυμα προειδοποίησης ότι ο πελάτης αυτός δεν έχει καταχωρηθεί.

Ρόλοι :Χρήστης ,Βάση δεδομένων πελάτη ,αναζήτηση πελάτη
Προ-συνθήκες: Ο χρήστης πληκτρολογεί το τύπο του λογαριασμού.
Βασικό Σενάριο: Αν πληκτρολογήσει το σωστό τύπο λογαριασμού ,μπορεί να ολοκληρώσει τις συναλλαγές του.
Εναλλακτικά σενάρια: Αν δεν πληκτρολογήσουμε τον σωστό τύπο, τότε εμφανίζεται μήνυμα προειδοποίησης ότι πρέπει να πληκτρολογήσουμε τον σωστό τύπο.



Ρόλοι :Χρήστης, Βάση δεδομένων πελάτη ,αναζήτηση πελάτη
Προ-συνθήκες: Ο χρήστης θέλει να κάνει ανάληψη.
Βασικό Σενάριο :Αν υπάρχει το απαιτούμενο ποσό γίνεται ανάληψη .
Εναλλακτικά σενάρια :Αν δεν υπάρχει εμφανίζει μήνυμα ότι το απαιτούμενο ποσό δεν υπάρχει.

Use and Case 5:



Περιπτώσεις ,Μεταφοράς (Transfer)

Ρόλοι :Χρήστης ,Βάση δεδομένων συναλλαγών ,συναλλαγές

Προ-συνθήκες: Ο χρήστης δίνει τον αριθμό ταυτότητας του πελάτη .

Βασικό Σενάριο: Αν οι αριθμοί ταυτότητας των πελατών είναι σωστοί γίνεται οι μεταφορά των χρημάτων.

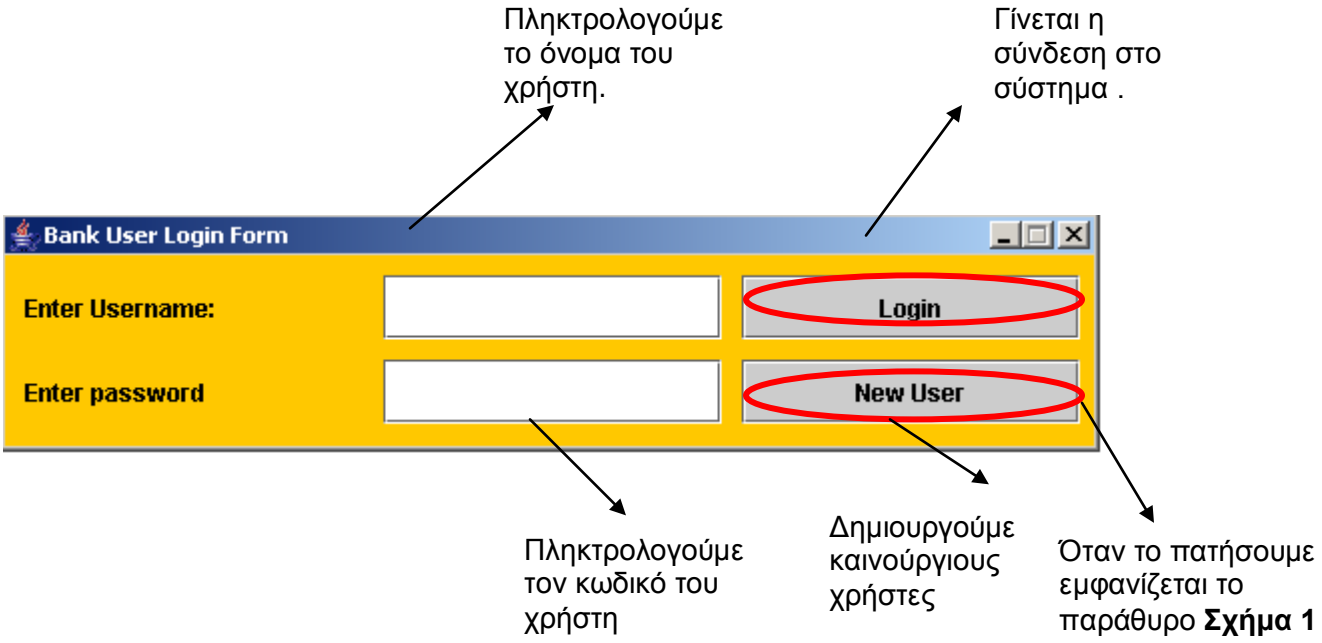
.**Εναλλακτικά σενάρια: :** Αν οι αριθμοί ταυτότητας των πελατών δεν είναι σωστοί θα πρέπει να εμφανίζεται μήνυμα προειδοποίησης ,ότι ο πελάτης αυτός δεν υπάρχει.

Ρόλοι :Χρήστης, Βάση δεδομένων συναλλαγών ,συναλλαγές
Προ-συνθήκες: Ο χρήστης θέλει να μεταφέρει χρήματα από ένα λογαριασμό σε άλλο.
Βασικό Σενάριο: Αν υπάρχουν τα απαιτούμενα χρήματα γίνεται η μεταφορά .
Εναλλακτικά σενάρια: Αν δεν υπάρχουν τα απαιτούμενα χρήματα θα πρέπει να εμφανίζεται μήνυμα προειδοποίησης ότι δεν υπάρχουν τα απαιτούμενα χρήματα.

Κεφάλαιο 6 : Εγχειρίδιο

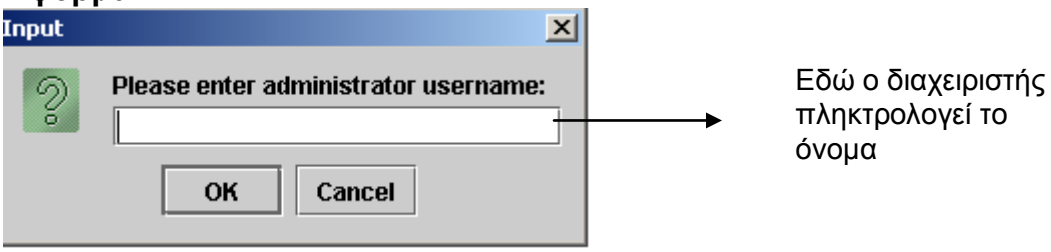
6.1 Εγχειρίδιο χρήσης

<u>Παράθυρο 1</u>*: Είναι το αρχικό μας παράθυρο και μας συνδέει με το υπόλοιπο σύστημα* .



Σχήμα 1

1 φόρμα



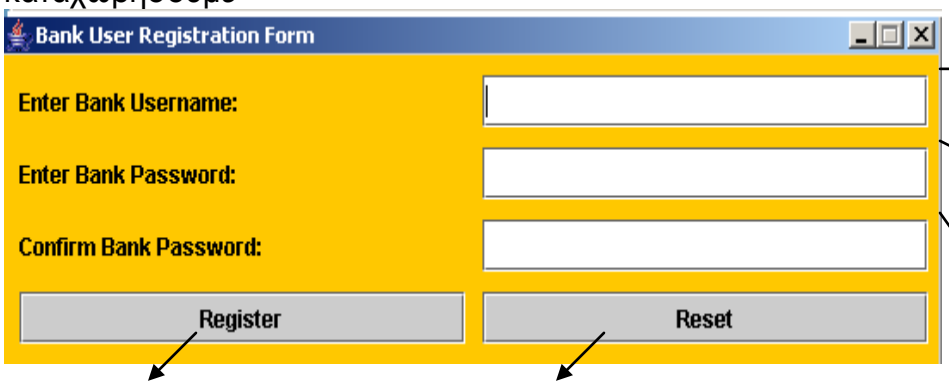
2 φόρμα

Εδώ ο διαχειριστής πληκτρολογεί το κωδικό

3 φόρμα

Στη φόρμα αυτή δημιουργούμε καινούργιους χρήστες .

καταχωρήσουμε



Γραφούμε Το όνομα του χρήστη που θέλουμε να καταχωρήσουμε

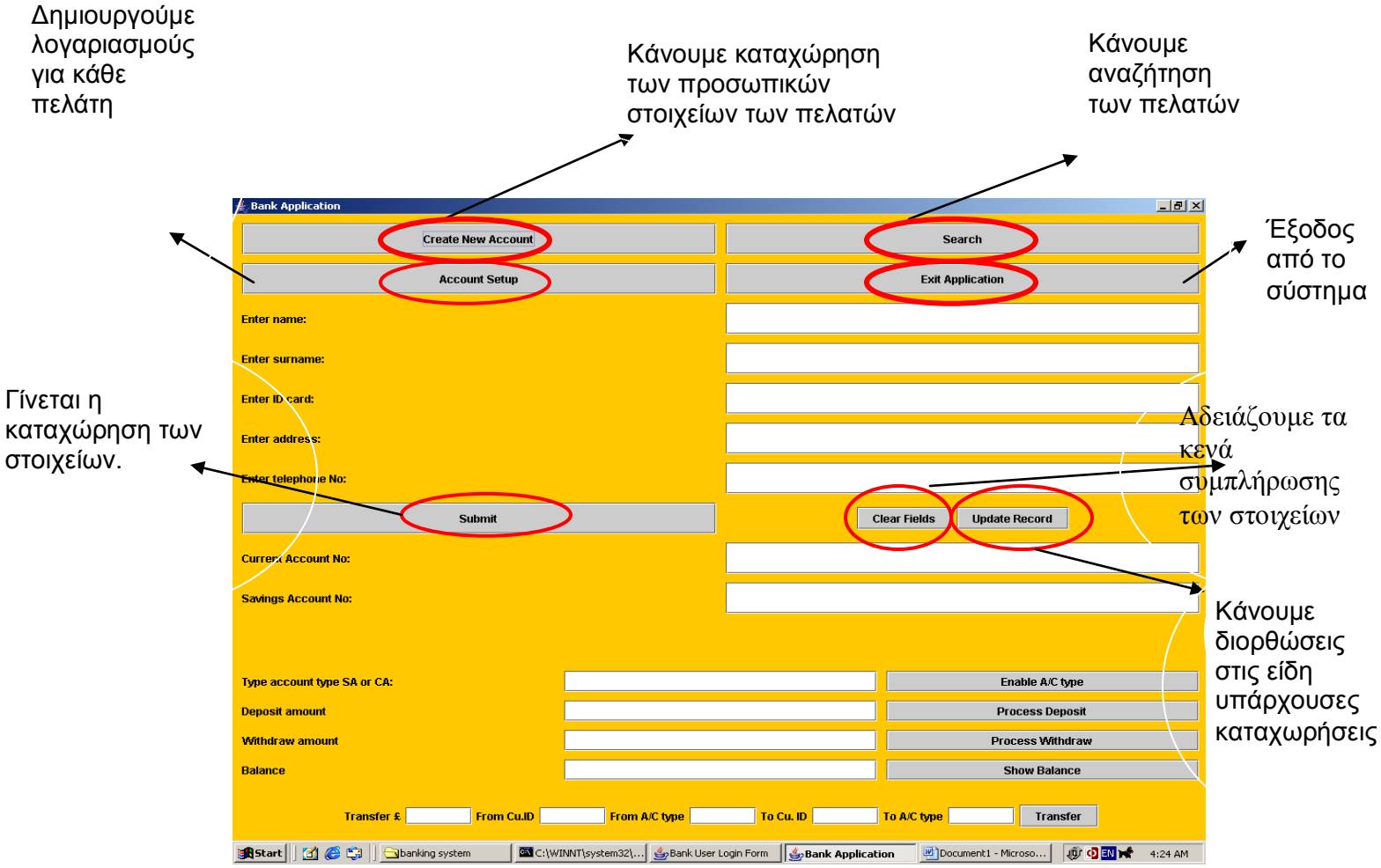
Γράφουμε το κωδικό του χρήστη που θέλουμε να καταχωρήσουμε

Επιβεβαιώνουμε τον κωδικό

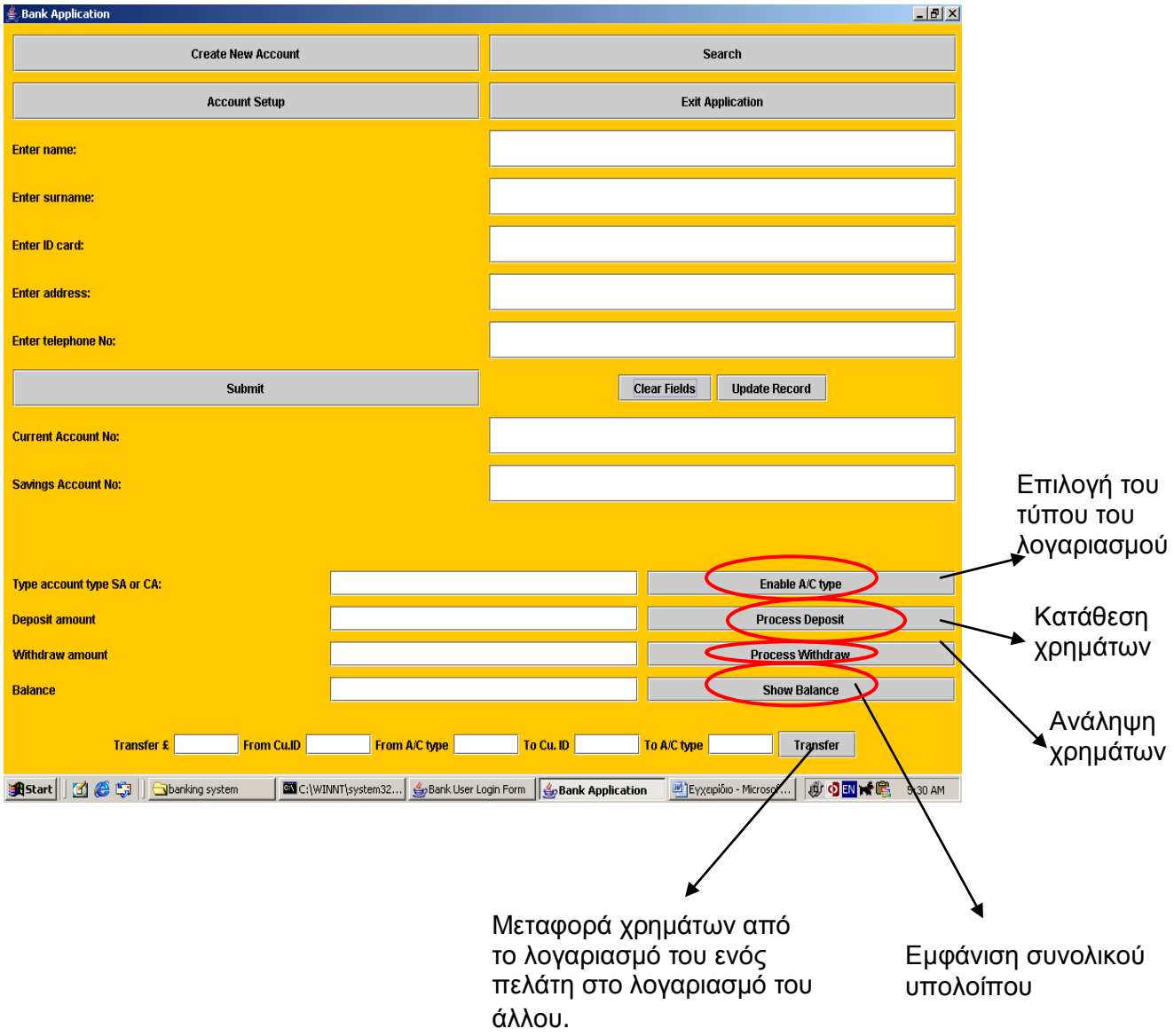
Γίνεται η καταχώρηση του νέου χρήστη

Αδειάζουμε τα κενά και μπορούμε να κάνουμε νέα καταχώρηση

Παράθυρο 2: Στο παράθυρο αυτό υλοποιούνται οι βασικές λειτουργίες του προγράμματος μας .



Παράθυρο 3:


Bank Application
Create New Account
Search
Account Setup
Exit Application
Enter name:
Enter surname:
Enter ID card:
Enter address:
Enter telephone No:
Submit
Clear Fields
Update Record
Current Account No:
Savings Account No:
Type account type SA or CA:
Enable A/C type
Deposit amount
Process Deposit
Withdraw amount
Process Withdraw
Balance
Show Balance
Transfer £
From Cu.ID
From A/C type
To Cu. ID
To A/C type
Transfer
Start
banking system
Bank User Login Form
Bank Application
Επιλογή του τύπου του λογαριασμού
Κατάθεση χρημάτων
Ανάληψη χρημάτων
Μεταφορά χρημάτων από το λογαριασμό του ενός πελάτη στο λογαριασμό του άλλου.
Εμφάνιση συνολικού υπολοίπου

Κεφάλαιο 7 : Μελλοντικές Επεκτάσεις

7.1 E-Banking :

Ένα από τα βασικά χαρακτηριστικά του λογισμικού μας θα είναι η ευελιξία .Η δυνατότητα του να ανταποκρίνεται στις νέες προκλήσεις της τεχνολογίας χωρίς κόπο και κόστος.Το E-Banking αποτελεί μια μελλοντική επέκταση του λογισμικού μας προγράμματος .

Το e-BANKING, η τραπεζική συναλλαγή μέσω της οθόνης του υπολογιστή με τη βοήθεια του INTERNET, αποτελεί μια καινοτομία στην κοινωνία που ζούμε, μια ιδέα επαναστατική που συμβάλλει στην αναβάθμιση της καθημερινής μας ζωής.

Το E-BANKING μεταφέρει την τράπεζα στην οθόνη του υπολογιστή σας.
Το παρεχόμενο φάσμα των υπηρεσιών θα περιλαμβάνει σταδιακά το σύνολο των τραπεζικών εργασιών.

Μέσω του INTERNET ο χρήστης μπορεί να συνδεθεί με την τράπεζα στην οποία διατηρεί λογαριασμό, να ρωτήσει για το υπόλοιπο του λογαριασμού του, να πληρώσει την πιστωτική του κάρτα, ή τις υποχρεώσεις του προς τρίτους, να υπολογίζει τους τόκους των καταθέσεων του ή να προχωρήσει ακόμη περισσότερο στην διαχείριση του ΧΑΡΤΟΦΥΛΑΚΙΟΥ του, στην εκτέλεση εντολών για το Χρηματιστήριο, στη λήψη και την αποπληρωμή δανείων του.

7.2 Τα πλεονεκτήματα του E-Banking:

- *Το INTERNET BANKING* ***χρησιμοποιεί λογισμικό που είναι εγκατεστημένο στον υπολογιστή της τράπεζας*****,** δίνοντας τη δυνατότητα στον πελάτη να έχει πρόσβαση στους λογαριασμούς του από οποιοδήποτε υπολογιστή που είναι συνδεδεμένος με το διαδίκτυο.

- *Οι τραπεζικές συναλλαγές μέσω INTERNET* ***βοηθούν τον χρήστη – πελάτη να αποφύγει την ταλαιπωρία των μετακινήσεων από τη μια τράπεζα στην άλλη,*** την αναμονή σε συχνά ατέλειωτες ουρές στα ταμεία τους, προσφέροντας του ταυτόχρονα άμεση πληροφόρηση και πληθώρα εναλλακτικών λύσεων σε όσα θέματα τον απασχολούν.

- ***Αρκεί ένα απλό κλικ στην αντίστοιχη επιλογή και όλες οι τράπεζες είναι στα πόδια σας!!!***

- *Η ηλεκτρονική τραπεζική είναι το μέσον που θα οδηγήσει* ***στην μείωση των λειτουργικών εξόδων*** *των τραπεζών.*
  Μια τραπεζική συναλλαγή μέσω διαδικτύου κοστίζει ελάχιστα έναντι μιας τραπεζικής συναλλαγής σε κατάστημα. Επιπλέον με το E-Banking και το E-Commerce είναι απαραίτητη η χρήση των πιστωτικών καρτών και κατ' επέκταση αυξάνονται τα τραπεζικά κέρδη. Όλες οι μεγάλες τράπεζες στην Ελλάδα έχουν προχωρήσει σε κάποια μορφή E-Banking.

7.3 Αρχιτεκτονική Web-based.

Στις μελλοντικές επεκτάσεις ,επίσης συμπεριλαμβάνεται και αρχιτεκτονική Web-based που περιγράψαμε παραπάνω. Επιλέγουμε την αρχιτεκτονική αυτή για την επικοινωνία των υποκαταστημάτων της τράπεζας .

Κεφάλαιο 8 : Ελαττώματα του λογισμικού μας

8.1 Ελαττώματα

Το λογισμικό μας λόγω πίεσης χρόνου παρουσιάζει κάποιες αδυναμίες οι οποίες είναι:

- ✓ Τα ποσά που μπορούμε να καταχωρήσουμε, δεν πρέπει να ξεπερνούν τα τρία ψηφία .Σε περίπτωση που καταχωρήσουμε μεγαλύτερο αριθμό , δεν γίνεται σωστή ολοκλήρωση των συναλλαγών .

- ✓ Πρέπει να συμπληρώνουμε ακριβώς τον τύπο του λογαριασμού που θέλουμε να κάνουμε κατάθεση ,ανάληψη ,η μεταφορά χρημάτων. Δηλαδή αν είναι ταμιευτηρίου πρέπει να πληκτρολογούμε Saving Account (SA) ,ενώ αν είναι τρεχούμενο Current Account (CA).Σε περίπτωση λάθους δεν ολοκληρώνονται σωστά η συναλλαγές .

Βιβλιογραφία

Σελίδες Internet

java.sun.com

www.myphone.gr

www.dmst.aueb.gr

www.infogr.org/content/computers/program/**java**/

thalis.cs.unipi.gr/~sgouros/Courses/OOP/**java**1.htm

www.cs.teilar.gr/gkakaron/**Java**/Index.html

publib.boulder.ibm.com/wcmid/ mp/v502/helpsystem/el/startNavBar.html

Βιβλία

Schild (2001). Οδηγός της Java 2. Εκδόσεις: Μ.Γκιούρδας.

K. N. King (2000). Java Programming: From the Beginning. W.W. Norton & Company.

Θεσσαλονίκη 2004. URL : http://seg.ee.upatras.gr/JavaBook

Java 2 Η Βίβλος **Εκδότης:** ΓΚΙΟΥΡΔΑΣ Μ.

• **Ivor Horton**, "Beginning Java 2 - Jdk 1.3 Edition"

Ένθετο Συμπλήρωμα

- **Κώδικας προγράμματος**
- **Έρευνα ερωτηματολογίων**

Παρουσίαση του Κώδικα

//Εδώ αναπτύσσουμε το interface Account //

```
public interface Account
{

    // get customerID
    public String getCustomerID(String id);

    // Set customerID
    public void setCustomerID (String id);

    //check the database if the ID already exists
    public boolean checkID(String id);

    // get the name of this account
    public String getName(String id);

    // Set the name on the account
    public void setName (String n);

    // get the surname of this account
    public String getSurname(String id);

     // Set the surname on the account
    public void setSurname (String s);


    // get the address of this account
    public String getAddress(String id);

     // Set the address on the account
    public void setAddress (String a);


    // get the telephone of this account
    public String getTelephone(String id);

     // Set the telephone on the account
    public void setTelephone (String t);

    //for update
```

```
    public void setID (String id);

    //get the value of the customer currentAcNo
     public boolean checkCurrentAcNo(String id);

    //set current account No
    public void setCurrentAcNo(String acnumber,String id);

    public String getCurrentAcNo(String id);

    // Get current balance
    public int getBalance(String id,String actype);


    // Take some money away
   public boolean withdraw(int amt,String id, String
actype);

    // Put some money in
    public void deposit(int amt,String actype,String id);

    //move money from another account into this one
   public void transfer (int amt,String fromid, String
fromactype,
                                      String toid,String
toactype  );

   //savings A/c methods
   public String getSavingsAcNo(String id);

   public void setSavingsAcNo(String sanumber,String id);

   public boolean checkSavingsAcNo(String id);

}
```

//Εδώ έχουμε την υλοποίηση του Account //

```
import java.sql.*;

public class AccountImpl implements Account
{

    // account number

    String number="";
    // balance
    private int balance=0;
    // Name on account
    private String name = "";
    private String surname = "";
    private String address = "";
    private String telephone = "";
     private String customerID = "";
    private static Statement st;
    ResultSet rs;

    // Create a new account with the given number
    public AccountImpl()
    {
    }

    // Create a new account with the given number
    public AccountImpl(Statement sta)


    {
     st = sta;
    }

    public void setID (String id)
    {
     customerID=id;
    }
```

```
    //check the database if the ID already exists
    public boolean checkID(String id)
    {
     String data="";
     try
     {
          rs = st.executeQuery("Select CustomerID from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               data = rs.getString("CustomerID");

               if(data.equals(id))
               {
                    //data already exists
                    return true;
                    }
          }
     }
     catch (Exception e)
        {
           e.printStackTrace();
        }
          return false;
    }

     // get customerID
    public String getCustomerID(String id)
    {
     String result="";
     try
     {
          rs = st.executeQuery("Select CustomerID from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("CustomerID");
          }
     }
     catch (Exception e)
        {
           e.printStackTrace();
```

```
        }
     return result;
    }

    // Set customerID
    public void setCustomerID (String id)
    {
        try
        {
             customerID  = id;
             st.executeUpdate("Insert into Customer
(CustomerID) values('"+customerID+"')");
     }
      catch (Exception e)
        {
            e.printStackTrace();
        }
    }


    // return the name of this account
    public String getName(String id)
    {
        String result="";
     try
     {
          rs = st.executeQuery("Select Name from Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("Name");
          }

     }

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

     return result;
    }

    // Set the name of the account
    public void setName (String n)
    {
```

```
        try
        {
        name = n;
        st.executeUpdate("Update Customer set Name
='"+name+"' where"+
        " CustomerID = '"+customerID+"'");

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

    // get the surname of this account
    public String getSurname(String id)
    {
     String result="";
     try
     {
          rs = st.executeQuery("Select Surname from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("Surname");
          }

     }

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

     return result;
    }

     // Set the surname on the account
    public void setSurname (String s)
    {
     try
        {
        surname=s;
        st.executeUpdate("Update Customer set Surname
='"+surname+"' where"+
```

```
          " CustomerID = '"+customerID+"'");

      }

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

     }

     // get the address of this account
     public String getAddress(String id)
     {
      String result="";
      try
      {
           rs = st.executeQuery("Select Address from
Customer"+
           " where CustomerID='"+id+"'");
           while(rs.next())
           {
                result = rs.getString("Address");
           }
      }
      catch (Exception e)
         {
             e.printStackTrace();
         }
      return result;
     }

      // Set the address on the account
     public void setAddress (String a)
     {
      try
         {
         address=a;
         st.executeUpdate("Update Customer set Address
='"+address+"' where"+
         " CustomerID = '"+customerID+"'");

      }

       catch (Exception e)
         {
             e.printStackTrace();
```

```
        }
    }

    // get the telephone of this account
    public String getTelephone(String id)
    {
     String result="";
     try
     {
          rs = st.executeQuery("Select Telephone from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("Telephone");
          }

     }

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

     return result;
    }

     // Set the telephone on the account
    public void setTelephone (String t)
    {
     try
        {
        telephone=t;
        st.executeUpdate("Update Customer set Telephone
='"+telephone+"' where"+
         " CustomerID = '"+customerID+"'");

     }

     catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

```
//*********************************************************
***********8
     public String getCurrentAcNo(String id)
     {
          String result="";

     try
        {
         customerID=id;
           rs = st.executeQuery("Select CurrentAcNo from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("CurrentAcNo");
               return result;
          }
     }

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

        return result;
     }

    public boolean checkCurrentAcNo(String id)
    {

     String result;

     try
        {
         customerID=id;

        rs = st.executeQuery("Select CurrentAcNo from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("CurrentAcNo");
               if(result==null)
               {
```

```
                    return true;
               }

               else if (result!=null)
               {
                    return false;
               }
          }
     }

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

     return true;
    }

    public void setCurrentAcNo(String acnumber,String id)
    {
     try
        {

          customerID=id;
         st.executeUpdate("Update Customer set CurrentAcNo
='"+acnumber+"' where"+
         " CustomerID = '"+customerID+"'");

     }

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

//*****************************************************
**********8
    public boolean checkSavingsAcNo(String id)
    {

     String result;

     try
        {
         customerID=id;
```

```
        rs = st.executeQuery("Select SavingsAcNo from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("SavingsAcNo");
               if(result==null)
               {
                    return true;
               }

               else if (result!=null)
               {
                    return false;
               }
          }
     }

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

     return true;
    }

    public void setSavingsAcNo(String sanumber,String id)
    {
     try
        {

           customerID=id;
          st.executeUpdate("Update Customer set SavingsAcNo
='"+sanumber+"' where"+
          " CustomerID = '"+customerID+"'");

     }

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

     public String getSavingsAcNo(String id)
     {
```

```
          String result="";

     try
        {
         customerID=id;
           rs = st.executeQuery("Select SavingsAcNo from
Customer"+
          " where CustomerID='"+id+"'");

          while(rs.next())
          {
               result = rs.getString("SavingsAcNo");
               return result;
          }
     }

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

        return result;
     }

//*****************************************************
********
    // Take some money away
    public boolean withdraw(int amt,String id, String
actype)
    {

        boolean that=true;
        try
        {
             if(getBalance(id,actype) >= amt)
             {
                 balance = getBalance(id,actype) - amt;

             st.executeUpdate("Insert into
Current(CustomerID,Withdraw,Balance,CASA) "+
             "values('"+
id+"',"+amt+","+balance+",'"+actype+"')");

                 that=true;
                 //return true;
             }
             else
```

```
            {
              that=false;
                //return false;
            }

     }

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

    }

    // Put some money in
    public void deposit (int amt,String actype,String id)
    {
        try
        {
        balance = getBalance(id,actype) + amt;

        st.executeUpdate("Insert into
Current(CustomerID,Deposit,Balance,CASA)"+
        "
values('"+id+"',"+amt+","+balance+",'"+actype+"')");

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

     }

     // Get current balance
    public int getBalance(String id,String actype)
    {
     try
     {
          rs = st.executeQuery("Select Balance from
Current"+
          " where CustomerID='"+id+"' and CASA =
'"+actype+"'");

          while(rs.next())
          {
```

```
                balance = rs.getInt("Balance");
            }

            return balance;

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

       //move money from another account into this one
       public void transfer (int amt,String fromid, String
fromactype,
                                        String toid,String
toactype  )

       {
           try
           {
            //this.withdraw(amt,fromactype,fromid);

             //AccountImpl from
             //from.deposit(amt,toactype,toid);

            //TransferFrom
            //TransferTo
            //TransferAmount
            //CASA
            int temp2balance = getBalance(toid,toactype);

            //System.out.println(""+temp2balance);

            temp2balance =temp2balance + amt;
            //System.out.println(""+temp2balance);


            //insert to the account where the user is
receiving the amount from
            st.executeUpdate("Insert into Current
(CustomerID,Deposit,TransferFrom,TransferAmount,Balance,CAS
A)"+

"values('"+toid+"',"+amt+",'"+fromid+"',"+amt+","+temp2bala
nce+",'"+toactype+"')");
```

```
         int temp1balance = getBalance(fromid,fromactype);
         //System.out.println(""+temp1balance);
         temp1balance =temp1balance - amt;
         //System.out.println(""+temp1balance);

         //insert to the account where the user is sending
the amount to
         st.executeUpdate("Insert into Current
(CustomerID,Withdraw,TransferTo,TransferAmount,Balance,CASA
)"+

"values('"+fromid+"',"+amt+",'"+toid+"',"+amt+","+temp1bala
nce+",'"+fromactype+"')");

     }

      catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

//Εδώ αναπτύσσουμε το interface του Login //

```
public interface Login
{

     //get Username
     public String getUsername();

     // Set the Username
    public void setUsername (String u);


     //get Password
     public String getPassword();

      // Set the Password
    public void setPassword (String p);

    //this method checks if the user exists
    public boolean checkLoginData(String user);
```

```
    //this method checks if the username and passowrd exist
in the database
    public boolean checkUP(String user,String password);

}
```

//Εδώ έχουμε την υλοποίηση της τάξης LoginForm //

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
//
public class LoginForm extends JFrame
{

     private JLabel user,pass,pass2;
    private JPanel p1;
    static ResultSet rs;
    private JButton login,newUser;
     private JTextField t0;
     private JPasswordField passfield;
     private static Statement statsent;
     LoginImpl log;

     LoginForm()
     {
          super("Bank User Login Form");
          setSize(550,120);
          setLocation(250,150);
          setResizable(false);
          Transaction handler = new Transaction();
          Container c = getContentPane();

          c.setLayout(new BorderLayout());

          //************************************
          p1 = new JPanel();
          p1.setLayout(new GridLayout(2,3,10,10));
          p1.setBackground(Color.orange);
          p1.setBorder(new EmptyBorder(10,10,10,5));
```

```
        user = new JLabel("Enter Username:");
        p1.add(user);
        t0 = new JTextField(20);
        p1.add(t0);

        login = new JButton("Login");
        login.addActionListener(handler);
        p1.add(login);


        pass = new JLabel("Enter password");
        p1.add(pass);

        passfield = new JPasswordField(20);
        p1.add(passfield);

        newUser = new JButton("New User");
        newUser.addActionListener(handler);
        p1.add(newUser);

        c.add(p1,BorderLayout.CENTER);
            //show();

     }


     public static void main (String [] args)
    {

          LoginForm lf = new LoginForm();

     //lf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          lf.show();
        try
        {

            //database driver for Access
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

            //connect to Database driver name bank
            Connection c =
DriverManager.getConnection("jdbc:odbc:bank","","");

            //create statement for maintaining the Database
            Statement st = c.createStatement();
```

```
                statsent = st;

                System.out.println("database connection
created!!");

        }

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


    class Transaction implements ActionListener
     {
          public void actionPerformed(ActionEvent e)
          {
                log = new LoginImpl(statsent);

                //create new account through the internal
frame
                if ( e.getActionCommand().equals("Login"))
                {

                     if(t0.getText().equals("") ||
passfield.getText().equals(""))
                     {
                          //if error  for recompile with
depreciation for details
                          //is shown on the compiler is due
to jdk version.


     JOptionPane.showMessageDialog(null,"You have not
entered any data",
                                   "Bank
Application",JOptionPane.ERROR_MESSAGE);
                     }

                     else
                     {


//System.out.println(t0.getText()+passfield.getText());
```

```
                    if(log.checkUP(t0.getText(),
passfield.getText()) == true)
                    {
                        //new window
                        MainBankForm bk = new
MainBankForm(statsent);
                        bk.show();

     //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                    }

                    else

JOptionPane.showMessageDialog(null,"This user is not
registered",
                    "Bank
Application",JOptionPane.WARNING_MESSAGE);
                   }
               }

               else if ( e.getActionCommand().equals("New
User"))
               {

                    try
                    {
                        String username =
JOptionPane.showInputDialog("Please enter administrator
username:");

                        String password =
JOptionPane.showInputDialog("Please enter password:");

                        if
(username.equals("administrator") &&
password.equals("administrator"))
                        {
                             LoginRegForm reg = new
LoginRegForm(log);
                             reg.show();
                        }

     //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                        else
                        {
```

```
JOptionPane.showMessageDialog(null,"Please enter the
correct administrator name and password",
                              "Bank
Application",JOptionPane.WARNING_MESSAGE);
                              }
                         }

                         catch(Exception ex)
                         {
                              System.exit(0);
                         }
                    }
               }
          }


}
```

//Εδώ έχουμε την υλοποιήση της τάξης LoginImpl //

```
import java.sql.*;

public class LoginImpl implements Login
{

     private String username;
     private String password;
     private static Statement st;
     ResultSet rs;

     //1st constructor
     public LoginImpl()
     {

     }

     //2nd constructor
     public LoginImpl(Statement sta)
     {
          st=sta;
     }


     //get Username
```

```
     public String getUsername()
     {
          return username;
     }

     // Set the Username
    public void setUsername (String u)
    {
     try
        {

        username = u;
        st.executeUpdate("Insert into Login (User)
values('"+username+"')");


     }

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


     //get Password
     public String getPassword()
     {
          return password;
     }

      // Set the Password
    public void setPassword (String p)
    {
     try
        {

        password = p;
       st.executeUpdate("Update Login set Password
='"+password+"' where"+
        " User = '"+username+"'");

     }

      catch (Exception e)
        {
            e.printStackTrace();
```

```
        }
    }

      public boolean checkLoginData(String user)
      {

          String data="";
          try
        {

                rs = st.executeQuery("Select User from
Login");

          while(rs.next())
          {
               data = rs.getString("User");
          }

          if (data.equals(user))
          {

               return false;
          }

          return true;

     }


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

      public boolean checkUP(String user,String password)
      {

          String data1="";
          String data2="";
          try
        {

                rs = st.executeQuery("Select User,Password
from Login");
```

```
          while(rs.next())
          {
               data1 = rs.getString("User");
               data2 = rs.getString("Password");


               //System.out.println("FromDB:"+data1+data2);

               //check if the user and password are the
same to procees
               //to the main bank application
               if (data1.equals(user) &&
data2.equals(password))
               {
                    return true;
               }
          }


     }


     catch (Exception e)
        {
            e.printStackTrace();
        }
        return false;
      }
}
```

//Εδώ έχουμε την υλοποιήση της τάξης LoginRegForm //

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;




public class LoginRegForm extends JFrame
{

     private JLabel
bankusername,bankpassword,confirmbankpassword;
    private JPanel p1;
```

```
    private JButton register,reset;
     private JTextField formuser;
     private JPasswordField
formpassfield,confirmformpassfield;
     LoginImpl log;

     LoginRegForm(LoginImpl logfrom)
     {
               super("Bank User Registration Form");
          setSize(550,180);
          setLocation(250,280);
          setResizable(false);
          log=logfrom;//this object will set the newuser
and password

          TransRegForm handler = new TransRegForm();
          Container c = getContentPane();

          c.setLayout(new BorderLayout());

          //************************************
          p1 = new JPanel();
          p1.setLayout(new GridLayout(4,2,10,10));
          p1.setBackground(Color.orange);
          p1.setBorder(new EmptyBorder(10,10,10,5));

               bankusername= new JLabel("Enter Bank
Username:");
          p1.add(bankusername);
          formuser = new JTextField(20);
          formuser.setEditable(true);
          p1.add(formuser);


          bankpassword= new JLabel("Enter Bank Password:");
          p1.add(bankpassword);

          formpassfield = new JPasswordField(20);
          formpassfield.setEditable(true);
          p1.add(formpassfield);

          confirmbankpassword = new JLabel("Confirm Bank
Password:");
          p1.add(confirmbankpassword);

          confirmformpassfield = new JPasswordField(20);
          confirmformpassfield.setEditable(true);
```

```
          p1.add(confirmformpassfield);


          //Button
          register =  new JButton("Register");
          register.addActionListener(handler);
          //register.setEditable(false);
          p1.add(register);

          reset =  new JButton("Reset");
          reset.addActionListener(handler);
          //reset.setEditable(false);
          p1.add(reset);

          c.add(p1,BorderLayout.NORTH);
              //show();
     }

     class TransRegForm implements ActionListener
     {
          public void actionPerformed(ActionEvent e)
          {

               String u1="",p1="",p2="";
               u1=formuser.getText();
               p1=formpassfield.getText();
               p2=confirmformpassfield.getText();

               if (
e.getActionCommand().equals("Register"))
               {
                    //check if the passowrd is the same
                    if(!p1.equals(p2))
                    {


     JOptionPane.showMessageDialog(null,"Password is not
the same",
                              "Bank
Application",JOptionPane.ERROR_MESSAGE);

                    }

                    else
if(u1.equals("")||p1.equals("")||p2.equals(""))
                    {
```

```
     JOptionPane.showMessageDialog(null,"Missing
data.Please fill the fields",
                         "Bank
Application",JOptionPane.ERROR_MESSAGE);

                    }

                    else
                    {
                     //first check if this user is already
register
                     if(log.checkLoginData(u1) ==true)
                     {
                         log.setUsername (u1);
                         log.setPassword (p1);


     JOptionPane.showMessageDialog(null,"User created
successful."+
                         "\nPlease close the window and
login.",
                              "Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                     }
                     else


     JOptionPane.showMessageDialog(null,"This user is
already registered",
                         "Bank
Application",JOptionPane.WARNING_MESSAGE);
                    }
               }
               else if (
e.getActionCommand().equals("Reset"))
               {
                    formuser.setText("");
                    formpassfield.setText("");
                    confirmformpassfield.setText("");
               }
          }
     }

}
```

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
```

//Εδώ έχουμε την υλοποιήση της τάξης MainBankForm //

```
public class MainBankForm extends JFrame
{
     JButton create, search, setup, transactions,submit,
clear,transferBut,

     depositBut,withdrawBut,balanceBut,update,typebutton;

     JPanel p1,p2,p3,p4;

     JLabel name,
surname,idcard,address,tel,cuaccountno,saaccountno,type,

deposit,withdraw,transferfrom,transferto,balance,transferam
ount,
             transtype1,transtype2;

     JTextField
t2,t3,t4,t5,t6,t7,t8,depositfld,withdrawfld,transferfromfld
,
     transfertofld,balancefld,transferamountfld,typefld,tra
nstype1fld,
     transtype2fld;

     private static Statement statsent;

     AccountImpl ac;

     //constructor
     MainBankForm(Statement st)
     {
               super("Bank Application");
          setSize(855,730);
          setLocation(100,5);
```

```
setResizable(true);

TransBank handler = new TransBank();
statsent=st;
Container c = getContentPane();
c.setLayout(new BorderLayout());

p1 = new JPanel();
p1.setLayout(new GridLayout(11,2,10,10));
p1.setBackground(Color.orange);
p1.setBorder(new EmptyBorder(10,10,10,5));

//Button GUI
create =  new JButton("Create New Account");
create.addActionListener(handler);
p1.add(create);

search =  new JButton("Search");
search.addActionListener(handler);
p1.add(search);

setup =  new JButton("Account Setup");
setup.addActionListener(handler);
p1.add(setup);

transactions=  new JButton("Exit Application");
transactions.addActionListener(handler);
p1.add(transactions);

//*****************************************
//New account Details
//***
name= new JLabel("Enter name:");
p1.add(name);

t2 = new JTextField(20);
p1.add(t2);

//***
surname= new JLabel("Enter surname:");
p1.add(surname);

t3 = new JTextField(20);
p1.add(t3);

//***
idcard= new JLabel("Enter ID card:");
```

```
p1.add(idcard);

t4 = new JTextField(20);
p1.add(t4);

//***
address= new JLabel("Enter address:");
p1.add(address);
t5 = new JTextField(20);
p1.add(t5);

//***
tel= new JLabel("Enter telephone No:");
p1.add(tel);
t6 = new JTextField(20);
p1.add(t6);

submit =  new JButton("Submit");
submit.addActionListener(handler);
p1.add(submit);


p2 = new JPanel();
p2.setLayout(new FlowLayout());
p2.setBackground(Color.orange);

clear =  new JButton("Clear Fields");
clear.addActionListener(handler);
p2.add(clear);

update =  new JButton("Update Record");
update.addActionListener(handler);
p2.add(update);

p1.add(p2);

cuaccountno = new JLabel("Current Account No:");
p1.add(cuaccountno);
t7 = new JTextField(20);
p1.add(t7);

saaccountno = new JLabel("Savings Account No:");
p1.add(saaccountno);
t8 = new JTextField(20);
p1.add(t8);

//*****************************************
```

```
            //transactions
            p3 = new JPanel();
            p3.setLayout(new GridLayout(4,3,10,10));
            p3.setBackground(Color.orange);
            p3.setBorder(new EmptyBorder(10,10,10,5));


                 //1
                 type= new JLabel("Type account type SA or
CA:");
                 p3.add(type);
                 typefld =  new JTextField(10);
                 p3.add(typefld);

                 typebutton = new JButton("Enable A/C type");
                 typebutton.addActionListener(handler);
            p3.add(typebutton);
                 //2
            deposit = new JLabel("Deposit amount");
            p3.add(deposit);

            depositfld = new JTextField(10);
            p3.add(depositfld);

            depositBut = new JButton("Process Deposit");
            depositBut.addActionListener(handler);
            p3.add(depositBut);

            //3
            withdraw = new JLabel("Withdraw amount");
            p3.add(withdraw);

            withdrawfld = new JTextField(10);
            p3.add(withdrawfld);

            withdrawBut= new JButton("Process Withdraw");
            withdrawBut.addActionListener(handler);
            p3.add(withdrawBut);

            //4
            balance = new JLabel("Balance");
            p3.add(balance);

            balancefld = new JTextField(10);
            p3.add(balancefld);

            balanceBut= new JButton("Show Balance");
```

```
balanceBut.addActionListener(handler);
p3.add(balanceBut);


//*****************************************
//gridlayout 1,7
p4 = new JPanel();
p4.setLayout(new FlowLayout());
p4.setBackground(Color.orange);
p4.setBorder(new EmptyBorder(10,10,10,5));
//**
transferamount = new JLabel("Transfer £");
p4.add(transferamount);
transferamountfld = new JTextField(6);
p4.add(transferamountfld);

//**
transferfrom = new JLabel("From Cu.ID");
p4.add(transferfrom);
transferfromfld = new JTextField(6);
p4.add(transferfromfld);


transtype1 = new JLabel("From A/C type");
p4.add(transtype1);
transtype1fld = new JTextField(6);
p4.add(transtype1fld);


//**
transferto = new JLabel("To Cu. ID");
p4.add(transferto);
transfertofld = new JTextField(6);
p4.add(transfertofld);

transtype2 = new JLabel("To A/C type");
p4.add(transtype2);

transtype2fld = new JTextField(6);
p4.add(transtype2fld);


transferBut= new JButton("Transfer");
transferBut.addActionListener(handler);
p4.add(transferBut);
```

```
          //*****************************************
               c.add(p1,BorderLayout.NORTH);
              c.add(p3,BorderLayout.CENTER);
              c.add(p4,BorderLayout.SOUTH);


     }


     class TransBank implements ActionListener
     {
          String input=""; //customer id
          String actype=""; //account type

          public void actionPerformed(ActionEvent e)
          {

               ac = new AccountImpl(statsent);


               if ( e.getActionCommand().equals("Create New
Account"))
               {

     JOptionPane.showMessageDialog(null,"Please fill the
form",
                    "Bank
Application",JOptionPane.INFORMATION_MESSAGE);

               }

               //search customer details
               else if (
e.getActionCommand().equals("Search"))
               {

                    String number="",number2="";
                    t2.setText("");
                    t3.setText("");
                    t4.setText("");
                    t5.setText("");
                    t6.setText("");
                    t7.setText("");
                    t8.setText("");
                    typefld.setText("");
                    transferamountfld.setText("");
                   transferfromfld.setText("");
```

```
                    transfertofld.setText("");
                    withdrawfld.setText("");
                    depositfld.setText("");
                    balancefld.setText("");
                    transtype1fld.setText("");
                     transtype2fld.setText("");

                     input =
JOptionPane.showInputDialog("Search Customer ID:");

                     //check the input with the database
                     if(ac.checkID(input)==true)
                     {
                      //name
                      t2.setText(ac.getName(input));
                      //surname
                      t3.setText(ac.getSurname(input));
                      //customerID
                      t4.setText(ac.getCustomerID(input));
                      //address
                      t5.setText(ac.getAddress(input));
                      //telephone
                      t6.setText(ac.getTelephone(input));

                      boolean result1 =
ac.checkCurrentAcNo(input);

                       if( result1 == true)
                       {

                        //random gerenation of account No
                          int value = (int) (Math.random() *
10000);
                          number =
"91"+Integer.toString(value);

                         int res =
JOptionPane.showConfirmDialog((Component)null,
                         "The Current Account No customer ID
"+input+" is not set.\n"+
                         "Current Account No generated
"+number+" .\nPress Yes to create current ac",
                         "Choose yes or
no",JOptionPane.YES_NO_OPTION);

                          if (res == JOptionPane.YES_OPTION)
                          {
```

```
                        ac.setCurrentAcNo(number,input);

JOptionPane.showMessageDialog(null,"Current Account No
created "
                         +number+".","Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                        t7.setText(number);
                        }
                    }

                    else if (result1==false)
                    {


t7.setText(ac.getCurrentAcNo(input));

                    }
                    //now check for the savings ac
                    boolean result2 =
ac.checkSavingsAcNo(input);

                    if( result2 == true)
                    {

                     //random gerenation of account No
                       int value2 = (int) (Math.random()
* 10000);
                       number2 =
"81"+Integer.toString(value2);

                      int res2 =
JOptionPane.showConfirmDialog((Component)null,
                      "The Savings Account No customer ID
"+input+" is not set.\n"+
                      "Current Account No generated
"+number2+" .\nPress Yes to create current ac",
                      "Choose yes or
no",JOptionPane.YES_NO_OPTION);

                       if (res2 ==
JOptionPane.YES_OPTION)
                       {
```

```
ac.setSavingsAcNo(number2,input);


JOptionPane.showMessageDialog(null,"Savings Account No
created "
                          +number2+".","Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                        t8.setText(number2);
                        }
                    }

                    else if (result2==false)
                    {


t8.setText(ac.getSavingsAcNo(input));

                    }

                  }

                  else

     JOptionPane.showMessageDialog(null,"This customer does
not"+
                               " exists.\nPlease insert a
new customer ID",
                               "Bank
Application",JOptionPane.ERROR_MESSAGE);


              }

              //set type of account and enable form fields

              else if (
e.getActionCommand().equals("Enable A/C type"))
              {
                   if(typefld.getText().equals(""))
                   {
                        //show error message

     JOptionPane.showMessageDialog(null,"Account type not
set.",
```

```
                              "Bank
Application",JOptionPane.ERROR_MESSAGE);
                    }


                    else if(typefld.getText().equals("SA"))
                    {
                         actype ="SA";
                         withdrawfld.setText("");
                    depositfld.setText("");
                    balancefld.setText("");

     JOptionPane.showMessageDialog(null,"The bank
application "+
                         "is enable for the Savings
account.",
                         "Bank
Application",JOptionPane.INFORMATION_MESSAGE);

                    }

                    else if(typefld.getText().equals("CA"))
                    {
                         actype ="CA";
                         withdrawfld.setText("");
                    depositfld.setText("");
                    balancefld.setText("");

     JOptionPane.showMessageDialog(null,"The bank
application "+
                         "is enable for the Current
account.",
                         "Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                    }
                    else

     //(!typefld.getText().equals("SA")||!typefld.getText()
.equals("CA"))
                    //show error message

     JOptionPane.showMessageDialog(null,"Account type not
set."+
                         "Please type 'SA' for savings or
'CA' for the current account.",
                         "Bank
Application",JOptionPane.ERROR_MESSAGE);
```

```
            }


            //2 types of account setup
            else if (
e.getActionCommand().equals("Account Setup"))
            {
              //create random number
              String number="",number2="";

              input = JOptionPane.showInputDialog("Enter
Customer ID card No:");
              //check the input with the database

                 if(ac.checkID(input)==true)
                 {

                   boolean result1 =
ac.checkCurrentAcNo(input);

                   if( result1 == true)
                   {

                    //random gerenation of account No
                      int value = (int) (Math.random() *
10000);
                      number =
"91"+Integer.toString(value);

                     int res =
JOptionPane.showConfirmDialog((Component)null,
                     "The Current Account No customer ID
"+input+" is not set.\n"+
                     "Current Account No generated
"+number+" .\nPress Yes to create current ac",
                     "Choose yes or
no",JOptionPane.YES_NO_OPTION);

                      if (res == JOptionPane.YES_OPTION)
                      {

                        ac.setCurrentAcNo(number,input);
```

```
JOptionPane.showMessageDialog(null,"Current Account No
created "
                          +number+".","Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                          t7.setText(number);
                          }
                       }

                       else if (result1==false)
                       {


t7.setText(ac.getCurrentAcNo(input));

                       }
                       //now check for the savings ac
                       boolean result2 =
ac.checkSavingsAcNo(input);

                       if( result2 == true)
                       {

                        //random gerenation of account No
                          int value2 = (int) (Math.random()
* 10000);
                          number2 =
"81"+Integer.toString(value2);

                         int res2 =
JOptionPane.showConfirmDialog((Component)null,
                         "The Savings Account No customer ID
"+input+" is not set.\n"+
                         "Current Account No generated
"+number2+" .\nPress Yes to create current ac",
                         "Choose yes or
no",JOptionPane.YES_NO_OPTION);

                          if (res2 ==
JOptionPane.YES_OPTION)
                          {


ac.setSavingsAcNo(number2,input);
```

```
JOptionPane.showMessageDialog(null,"Savings Account No
created "
                          +number2+".","Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                          t8.setText(number2);
                          }
                       }

                       else if (result2==false)
                       {


t8.setText(ac.getSavingsAcNo(input));

                       }
                    }

                    else

     JOptionPane.showMessageDialog(null,"This customer does
not"+
                                " exists.\nPlease insert a
new customer ID",
                                "Bank
Application",JOptionPane.ERROR_MESSAGE);

                }

                //enable bank transactions
                else if ( e.getActionCommand().equals("Exit
Application"))
                {


     JOptionPane.showMessageDialog(null,"Terminating bank
application.",
                           "Bank
Application",JOptionPane.WARNING_MESSAGE);
                    System.exit(0);


                }

                //submit user details
```

```
               else if (
e.getActionCommand().equals("Submit"))
               {


     if(t2.getText().equals("")||t3.getText().equals("")||

     t4.getText().equals("")||t5.getText().equals("")||
                         t6.getText().equals(""))
                    {

     JOptionPane.showMessageDialog(null,"Missind data",
                         "Bank
Application",JOptionPane.ERROR_MESSAGE);
                         return;
                    }

                    else
                    {
                         //insert into the database
                         //first check if this user alreay
exists
                         if(ac.checkID(t4.getText())==true)
                         {

     JOptionPane.showMessageDialog(null,"This customer
already"+
                              " exists.\nPlease insert a
new customer ID",
                              "Bank
Application",JOptionPane.ERROR_MESSAGE);
                              return;

                         }

                         else
                         {

                         ac.setCustomerID(t4.getText());
                          ac.setName(t2.getText());
                          ac.setSurname(t3.getText());
                          ac.setAddress(t5.getText());
                          ac.setTelephone(t6.getText());


JOptionPane.showMessageDialog(null,"Customer succesfull
created.",
```

```
                              "Bank
Application",JOptionPane.INFORMATION_MESSAGE);

                         }
                     }
               }

               else if ( e.getActionCommand().equals("Clear
Fields"))
               {
                    t2.setText("");
                    t3.setText("");
                    t4.setText("");
                    t5.setText("");
                    t6.setText("");
                    t7.setText("");
                    t8.setText("");
                    typefld.setText("");
                    transferamountfld.setText("");
                   transferfromfld.setText("");
                   transfertofld.setText("");
                   withdrawfld.setText("");
                   depositfld.setText("");
                   balancefld.setText("");
                   transtype1fld.setText("");
                    transtype2fld.setText("");
               }


               //update customer details
               else if (
e.getActionCommand().equals("Update Record"))
               {

                if(!t4.getText().equals(""))
                {

                     int result =
JOptionPane.showConfirmDialog((Component)null,
                       "Are you sure you want to update
this record?","Choose yes or no",
                          JOptionPane.YES_NO_OPTION);

                          if (result ==
JOptionPane.YES_OPTION)
                          {
```

```
                        ac.setID(t4.getText());
                        ac.setName(t2.getText());
                        ac.setSurname(t3.getText());
                        ac.setAddress(t5.getText());
                        ac.setTelephone(t6.getText());


JOptionPane.showMessageDialog(null,"Record updated
successful.",
                        "Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                       }
                       else


     JOptionPane.showMessageDialog(null,"Data not
updated.",
                        "Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                 }
                     else


     JOptionPane.showMessageDialog(null,"Customer ID not
entered.",
                        "Bank
Application",JOptionPane.ERROR_MESSAGE);

               }

               else if (
e.getActionCommand().equals("Process Deposit"))
               {

                 //first check of the account type has been
set
                 if(actype.equals("SA")||
actype.equals("CA"))
                 {

                      if(depositfld.getText().equals(""))
                      {

     JOptionPane.showMessageDialog(null,"Amount not
entered.",
```

```
                                "Bank
Application",JOptionPane.ERROR_MESSAGE);
                        }

                        else
                        {
                         //System.out.println("Input" +
input);


ac.deposit(Integer.parseInt(depositfld.getText()),actype,in
put);


JOptionPane.showMessageDialog(null,"Amount deposited: "+
                          depositfld.getText(),
                          "Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                       }
                  }

                  else
                  JOptionPane.showMessageDialog(null,"You
have not entered an account"+
                  "type. Transaction failed ",
                                "Bank
Application",JOptionPane.ERROR_MESSAGE);
                 }

                else if (
e.getActionCommand().equals("Process Withdraw"))
                {
                     //first check of the account type has
been set
                  if(actype.equals("SA")||
actype.equals("CA"))
                  {

                     if(withdrawfld.getText().equals(""))
                     {

     JOptionPane.showMessageDialog(null,"Amount not
entered.",
                            "Bank
Application",JOptionPane.ERROR_MESSAGE);
                     }
```

```
                    else
                    {

                      //withdraw money function

if(ac.withdraw(Integer.parseInt(withdrawfld.getText()),inpu
t,actype)==true)


JOptionPane.showMessageDialog(null,"Amount withdrawn:"
                     +withdrawfld.getText(),"Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                         else
                         {

     JOptionPane.showMessageDialog(null,"Not enough money
to withdraw."+
                         "\nPlease check your balance.",
                         "Bank
Application",JOptionPane.ERROR_MESSAGE);
                         }
                    }
                 }

                 else
                 JOptionPane.showMessageDialog(null,"You
have not entered an account"+
                 "type. Transaction failed ",
                              "Bank
Application",JOptionPane.ERROR_MESSAGE);

               }

               else if (
e.getActionCommand().equals("Transfer"))
               {

                     String from, to, fromtype, totype,
amount;



                     amount = transferamountfld.getText();
                     from = transferfromfld.getText();
                     to = transfertofld.getText();
                     fromtype =transtype1fld.getText();
                     totype =transtype2fld.getText();
```

```
                    int amt=Integer.parseInt(amount);

                    int
tempx=(int)ac.getBalance(from,fromtype);


                     if
(amount.equals("")||from.equals("") || to.equals("")
                          || fromtype.equals("")||
totype.equals(""))
                     {

JOptionPane.showMessageDialog(null,"Missind data.",
                               "Bank
Application",JOptionPane.ERROR_MESSAGE);
                          return;
                     }


                    else if(ac.checkID(from)==false ||
ac.checkID(to)==false)
                   {

     JOptionPane.showMessageDialog(null,"This customer does
not"+
                                      " exists.\nPlease insert
a new customer ID",
                                      "Bank
Application",JOptionPane.ERROR_MESSAGE);
                          return;
                     }


               /*   else
if(ac.withdraw(Integer.parseInt(amount),from,fromtype)==fal
se

     ||ac.withdraw(Integer.parseInt(amount),from,totype)==f
alse

     ||ac.withdraw(Integer.parseInt(amount),to,fromtype)==f
alse

     ||ac.withdraw(Integer.parseInt(amount),to,totype)==fal
se )
                   {
```

```
     JOptionPane.showMessageDialog(null,"Not enough money
to withdraw."+
                          "\nPlease check your balance.",
                          "Bank
Application",JOptionPane.ERROR_MESSAGE);

                    }

                */

                    else if(fromtype.equals("SA")||
fromtype.equals("CA")||
                                totype.equals("SA")  ||
totype.equals("CA"))
                    {

                        boolean check=false;
                        if(tempx>=amt)


//ac.withdraw(Integer.parseInt(amount),from,fromtype)==true
                          //||
ac.withdraw(Integer.parseInt(amount),from,totype)==true )

                          {


                               if ((tempx==0) && (amt==0))
                               {


     JOptionPane.showMessageDialog(null,"Transfer
unsuccessful.",
                               "Bank
Application",JOptionPane.INFORMATION_MESSAGE);
                              }
                              else{ check=true; }


                            //System.out.println("account
"+fromtype+" to transfer"+tempx+"  existing bal "+amt);

                             if( check==true ){
```

```
     ac.transfer(Integer.parseInt(amount),from,fromtype,
to,totype);


     JOptionPane.showMessageDialog(null,"Transfer
successful.",
                                "Bank
Application",JOptionPane.INFORMATION_MESSAGE);


                               }
                          }
                          else
                          {

     JOptionPane.showMessageDialog(null,"Not enough money
to withdraw."+
                          "\nPlease check your balance.",
                          "Bank
Application",JOptionPane.ERROR_MESSAGE);
                          }


                    }

                          else

     JOptionPane.showMessageDialog(null,"Invalid account
type.\n"+
                                "Please type SA for savings
or CA for current account type.",
                                     "Bank
Application",JOptionPane.ERROR_MESSAGE);

               }

               else if ( e.getActionCommand().equals("Show
Balance"))
               {
                 //first check of the account type has been
set
                 if(actype.equals("SA")||
actype.equals("CA"))
                 {
                          int temp =
ac.getBalance(input,actype);
```

```
                        balancefld.setText(""+temp);
                }

                else
                JOptionPane.showMessageDialog(null,"You
have not entered an account"+
                "type. Transaction failed ",
                "Bank
Application",JOptionPane.ERROR_MESSAGE);

            }
        }
    }
}
```

Ερωτηματολόγια που απευθύνονται στους χρήστες και προϊσταμένους .
Συμπεράσματα από τρία δείγματα .(Εθνική ,Κύπρου ,Λαϊκή)

Ερωτηματολόγια που αφορούν τους χρήστες :

1. Πιο είναι το μορφωτικό σας επίπεδο :

Δευτεροβάθμια	25%
Τριτοβάθμια	45%
Μεταπτυχιακοί	30%

2. Ποια είναι η εξοικείωση σας με τους υπολογιστές :

Καθόλου	44%
Καλή	36%
Πολύ καλή	20%

3. Είστε ευχαριστημένη από το υπάρχον λογισμικό :

Ναι	60%
Όχι	40%

4. Τι θα ζητούσατε από ένα καινούργιο λογισμικό :

Αξιοπιστία	55%
Ταχύτητα	25%
Ευελιξία	20%

5. Πιστεύετε ότι ένα καινούργιο σύστημα θα σας κάνει πιο γρήγορους κα ευέλικτους στην δουλεία σας :

Ναι	60%
Όχι	40%

6. Πιστεύετε ότι ένα καινούργιο λογισμικό θα αντικαταστήσει τη δουλεία σας :

Ναι	52%
Όχι	48%

7. Το internet Banking το θεωρείτε απαραίτητο:

Ναι	45%
Όχι	55%

Ερωτηματολόγια που αφορούν τους προϊσταμένους :

1. Πιο είναι το μορφωτικό επίπεδο των υπαλλήλων σας :

Δευτεροβάθμια	20%
Τριτοβάθμια	50%
Μεταπτυχιακοί	30%

2. Η τράπεζα σας ακολουθεί τις εξελίξεις της τεχνολογίας :

Ναι	25%
Όχι	75%

3. Πιστεύετε ότι οι υπάλληλοι σας είναι ευχαριστημένοι από το υπάρχον σύστημα :

Ναι	60%
Όχι	40%

4. Θεωρείτε πιο εύκολο να αναβαθμίσετε το υπάρχον σύστημα ή να το αντικαταστήσετε με ένα καινούργιο :

Αναβάθμιση	20%
Αντικατάσταση	80%

5. Οι υπάλληλοι θα αντιμετώπιζαν θετικά την αντικατάσταση του υπάρχοντος λογισμικού :

Ναι	35%
Όχι	65%

6. Το κόστος αποτελεί σημαντικό ανασταλτικό παράγοντα :

Ναι	80%
Όχι	20%

Ανάλυση των ερωτηματολογίων χρηστών :

Πριν ξεκινήσουμε την ανάπτυξη του λογισμικού θα πρέπει να κάνουμε έρευνα αγοράς για να δούμε αν η αγορά είναι έτοιμή να δεχθεί ένα καινούργιο λογισμικό .

Θα ετοιμάσουμε δύο κατηγορίες ερωτηματολογίων αυτά που θα απευθύνονται στους υπαλλήλους και τους προϊσταμένους της τράπεζας .

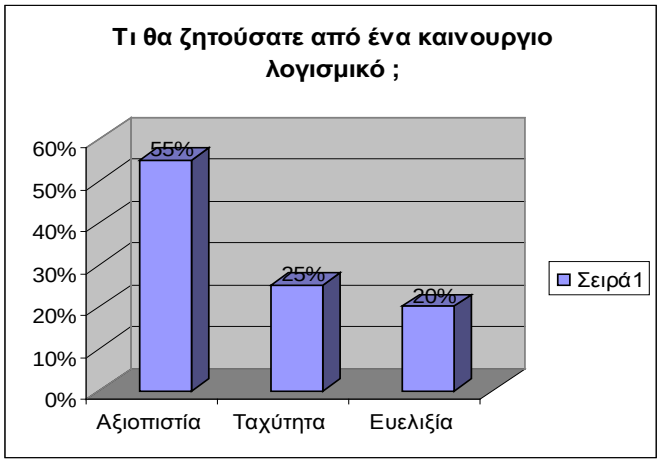
Αρχικά θα προσπαθήσουμε να εντοπίσουμε το μορφωτικό επίπεδο των υπαλλήλων *Σχήμα 1*.Διαπιστώνουμε ότι οι περισσότεροι υπάλληλοι της τράπεζας είναι απόφοιτοι τριτοβάθμιας εκπαίδευσης, αυτό είναι ιδιαίτερα θετικό .Έχουν κάποια εξοικείωση με τους υπολογιστές αυτό το δεδομένο μας ενθαρρύνει ,καθώς θα μπορέσουν εύκολα να εξοικειωθούν με μια ενδεχόμενη αλλαγή *Σχήμα 2.*Διαπιστώνουμε ότι έχουν μια αρνητική αντιμετώπιση στην αντικατάσταση του λογισμικού *Σχήμα 5*. Είναι φυσικό ,γιατί ένα καινούργιο λογισμικό αποτελεί κίνδυνο να αντικαταστήσει τη δουλειά τους .Υπάρχουν όμως και κάποιοι υπάλληλοι που αντιμετωπίζουν πιο θετικά .Ένα σύγχρονο λογισμικό θα τους κάνει πιο αποτελεσματικούς και και ευέλικτους στην δουλεία τους *Σχήμα 4.* Ένα καινούργιο λογισμικό δεν πρέπει να είναι στατικό θα πρέπει να ακολουθεί τις αλλαγές και προσαρμόζεται εύκολα και γρήγορά σε αυτές .Μια τέτοια εξέλιξη θα είναι και το Internet Banking .Διαπιστώνουμε ότι ένα μεγάλο ποσοστό θεωρεί σημαντική μια τέτοια εξέλιξη *Σχήμα 6.*

Τα συμπεράσματα που βγάζουμε από τα ερωτηματολόγια των χρηστών είναι ,ότι οι υπάλληλοι φοβούνται μια πιθανή αντικατάσταση του λογισμικού. Όμως αναγνωρίζουν και τα πλεονεκτήματα που μπορούν να προκύψουν .

Σχήμα 1

Σχήμα 2

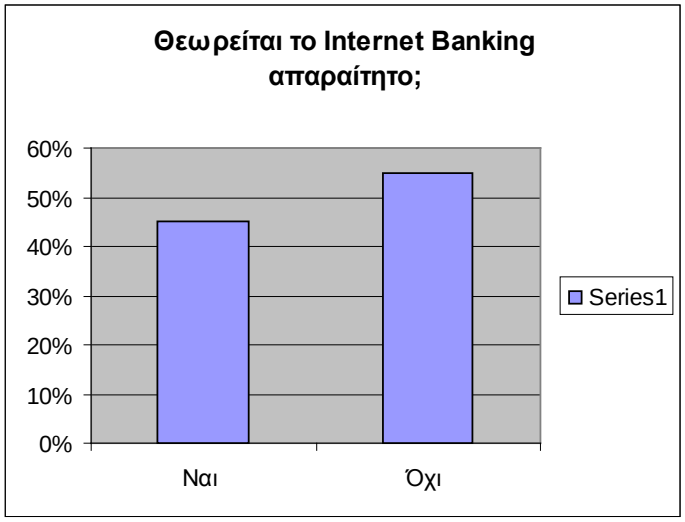
Σχήμα 3



Σχήμα 4

Σχήμα 5

Σχήμα 6

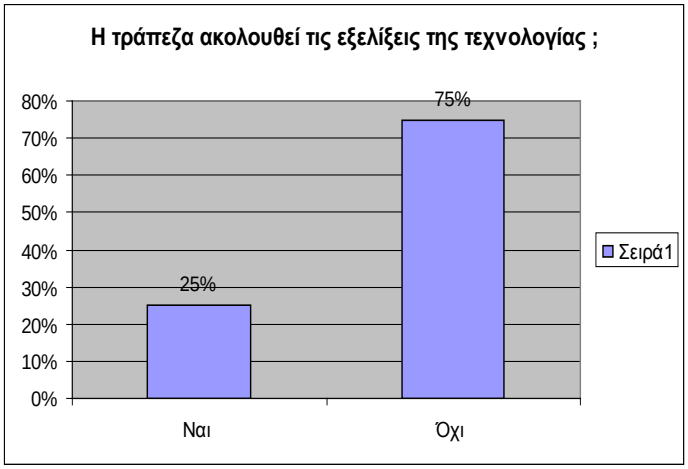


Ανάλυση ερωτηματολογίων για τους προϊσταμένους:

Εκτός από τους χρήστες είναι σημαντικό να δούμε πως αντιμετωπίζουν και οι προϊστάμενοι μια ενδεχόμενη αντικατάσταση του υπάρχοντος λογισμικού.

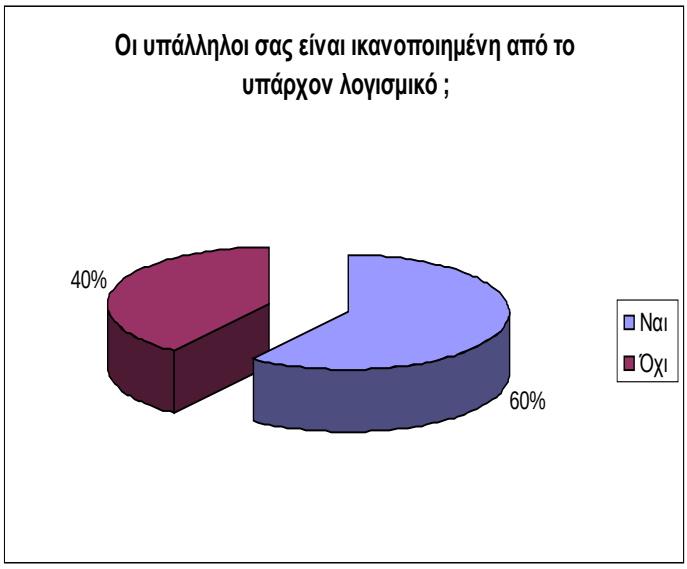
Αρχικά θέλουμε να διαπιστώσουμε σε τι μορφωτικό επίπεδο επιλέγουν να είναι οι υπάλληλοι των τραπεζών.Σημαντικό επίσης είναι *να* διαπιστώσουμε αν οι τράπεζες προσπαθούν να ακολουθούν τις εξελίξεις της τεχνολογίας *Σχήμα 1.* Η αναβάθμιση των υπαρκτών λογισμικών κοστίζει πολύ πιο ακριβά από την αντικατάσταση *Σχήμα 2.*Αυτό οφείλεται στο γεγονός ότι τα λογισμικά που υπάρχουν αυτή την στιγμή είναι κυρίως πολύ παλιά και είναι πολύ δύσκολο να ακολουθήσουν την σημερινή τεχνολογία .Το κόστος αποτελεί σημαντικό ανασταλτικό παράγοντα *Σχήμα 5.*

Σχήμα 1

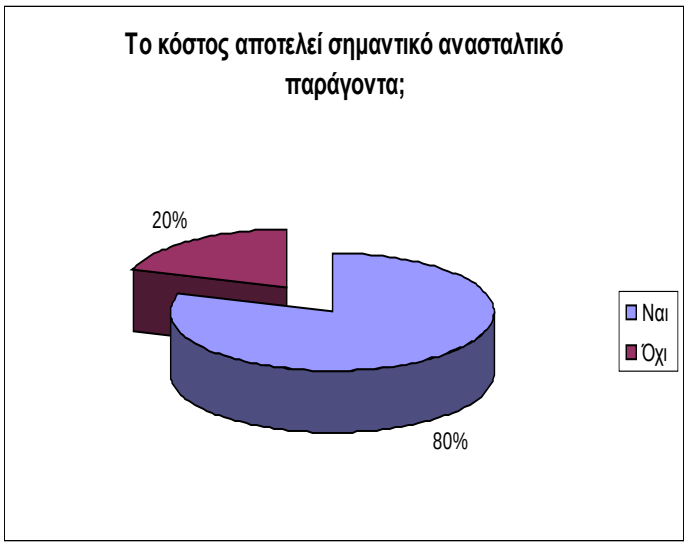


Σχήμα 2

Σχήμα 3



Σχήμα 4


Το κόστος αποτελεί σημαντικό ανασταλτικό παράγοντα;
20%
80%
Ναι
Όχι