

ΕΡΓΑΣΙΑ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ 2017-2018

Ο/Ε-ΜΗΤΡΩΟ: ΖΕΡΚΕΛΙΔΗΣ ΔΗΜΗΤΡΗΣ Π14046

ΘΕΜΑ ΕΡΓΑΣΙΑΣ:

Υλοποίηση ενός ATM με RMI, το οποίο θα έχει τις παρακάτω λειτουργίες:

- Ο χρήστης θα εισάγει τον αριθμό της κάρτας του και μετά θα εισάγει και το PIN.
- Το ATM θα καλεί μεθόδους από το σέρβερ της τράπεζας.
- Θα υπάρχει η λειτουργία της κατάθεσης,
- Της ανάληψης,
- εμφάνισης BALANCE,
- Πληροφορίες συναλλαγών που έχει κάνει ο χρήστης, οι οποίες θα είναι είτε όλες , είτε σε ορισμένο χρονικό διάστημα που θα το δηλώνει,
- τέλος , θα μπορεί να αλλάζει το PIN του.

ΠΡΟΛΟΓΟΣ:

RMI σημαίνει Remote Method Invocation , δηλαδή απομακρυσμένη επίκληση μεθόδου.

Είναι μηχανισμός που επιτρέπει σε ένα αντικείμενο που βρίσκεται σε ένα JVM σύστημα να έχει πρόσβαση σε ένα αντικείμενο που τρέχει σε άλλο JVM σύστημα.

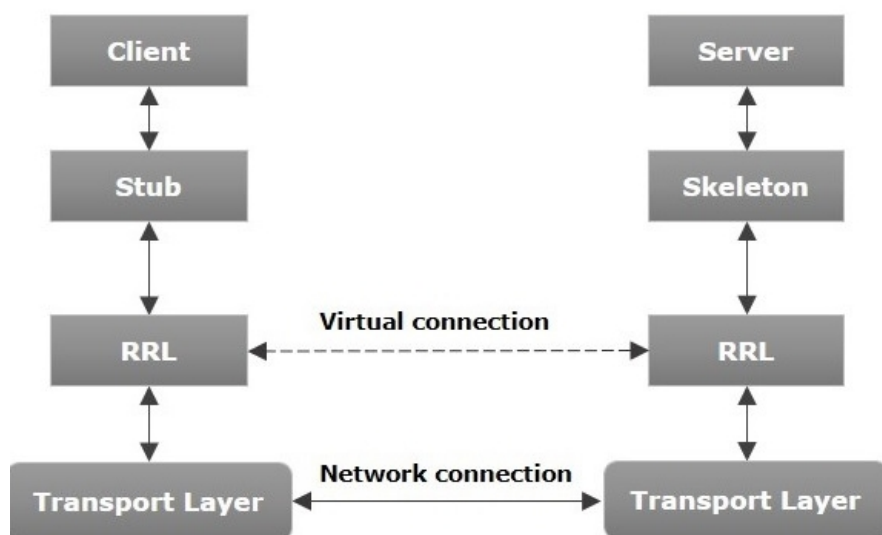
Το RMI χρησιμοποιείται για να δημιουργήσουμε κατανεμημένες εφαρμογές : παρέχει απομακρυσμένη επικοινωνία μεταξύ java προγραμμάτων.(παρέχεται απο το package java.rmi)

ΑΡΧΙΤΕΚΤΟΝΙΚΗ RMI APPLICATION:

Γράφουμε 2 προγράμματα ένα για τον εξυπηρετητή και ένα για τον πελάτη (server-client).

Μέσα στο server program ,ένα απομακρυσμένο αντικείμενο δημιουργείται και η αναφορά του αντικειμένου αυτού γίνεται διαθέσιμη στον πελάτη μέσω του registry.

Ο client ζητά αυτό το αντικείμενο από τον server απομακρυσμένα , και προσπαθεί να καλέσει τις μεθόδους του.



TRANSPORT LAYER: Χάρη σ' αυτό το επίπεδο συνδεέται ο πελάτης με τον εξυπηρετητή. Διαχειρίζεται την υπάρχουσα σύνδεση και όποια άλλη προκύψει.

STUB: Κορμός ή αλλιώς stub είναι η αντιπροσώπευση του απομακρυσμένου αντικειμένου στον πελάτη. Βρίσκεται στον πελάτη και λειτουργεί ως gateway του.

SKELETON: Βρίσκεται στον εξυπηρετητή και επικοινωνεί με τον κορμό (stub) για να περάσει το αίτημα για το απομακρυσμένο αντικείμενο.

Remote Reference Layer(RRL):

Είναι το επίπεδο που διαχειρίζεται τις αναφορές που κάνει ο πελάτης στο απομακρυσμένο αντικείμενο.

Marshalling-Unmarshalling:

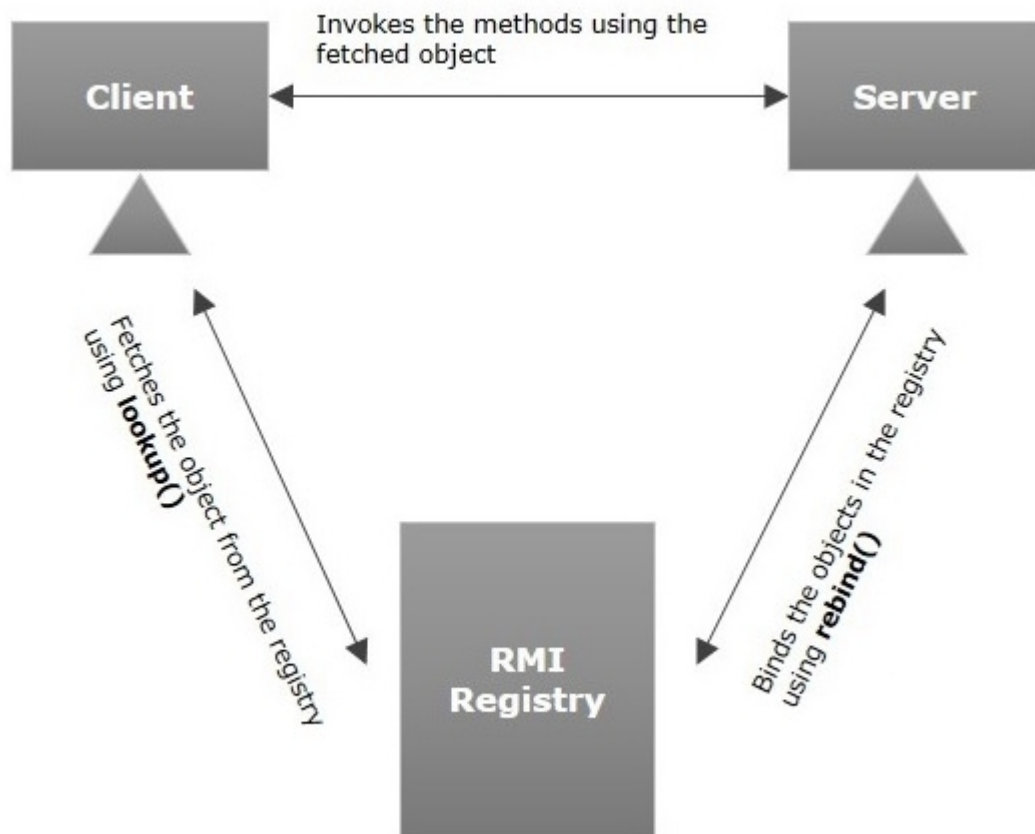
Όποτε ο πελάτης καλεί μια μέθοδο που δέχεται τις παραμέτρους σε ένα απομακρυσμένο αντικείμενο, οι παράμετροι αυτοί συσσωρεύονται σε ένα μήνυμα πριν σταλούν μέσω δικτύου. Αν οι παράμετροι είναι κανονικοί και όχι αντικείμενα -object μπαίνουν μαζί με ένα header. Αλλιώς αν είναι objects τότε γίνονται serialized. Αυτό σημαίνει marshalling.

Ο server από την άλλη παίρνει το μήνυμα και το διαμορφώνει όπως πρέπει και μετά καλεί τη μέθοδο. Αυτό λέγεται unmarshalling.

RMI-REGISTRY:

RMI-REGISTRY είναι ένα namespace που όλα τα αντικείμενα του σέρβερ τοποθετούνται. Αυτή η λειτουργία γίνεται με τις bind(),rebind() μεθόδους. Αυτά γίνονται register με τη χρήση κάποιου μοναδικού ονόματος(bind name).

Για να καλέσει ο πελάτης αυτό το αντικείμενο θέλει την αναφορά του, έτσι ο πελάτης προσκομίζει το αντικείμενο με τη χρήση του bind name και της lookup() μεθόδου.



ΛΕΙΤΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Για ένα RMI APPLICATION πρέπει να γίνουν τα εξής:

- Ορισμός του remote interface
- Ανάπτυξη του implementation class (remote object)
- Ανάπτυξη του server program
- Ανάπτυξη του client program

ΟΡΙΣΜΟΣ INTERFACE:

```
import java.rmi.*;

public interface ATM_Interface extends Remote
{
    public void deposit(int x,String y) throws RemoteException, SQLException;
    public void withdraw(int x,String y) throws RemoteException, SQLException;
    public int display_balance(String y) throws RemoteException, SQLException;
    public String transaction_details1( String y) throws RemoteException, SQLException;
    public String transaction_details2(String y, Date start_date, Date end_date) throws RemoteException, SQLException;
    public boolean login(String acc,int pin) throws RemoteException,SQLException;
    public void change_pin(String acc,int pin) throws RemoteException,SQLException;
}
```

Κάνουμε extend το Remote που είναι ορισμένο στο πακέτο και ορίζω όλες τις μεθόδους που θα υλοποιεί ο server, όπως λεγαμε και στην αρχή της παρουσίασης κατάθεση,ανάληψη,πληροφορίες συναλλαγών απο πάντα και σε ορισμένο χρονικό διάστημα , login και αλλαγή κωδικού.

Πετάμε RemoteException γιατί λόγω δικτύου ίσως υπάρχει κάποιο πρόβλημα και SQLException λόγω της χρήσης βάσης δεδομένων.

Ανάπτυξη του implementation class:

βιβλιοθήκες:

```
import java.rmi.*;

import java.rmi.server.UnicastRemoteObject;
import java.sql.*;
import java.time.LocalDate;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
```

σύνδεση με βάση:

```
public class ATM_Implementation extends UnicastRemoteObject implements ATM_Interface
{
    private static final long serialVersionUID = 1L;
    // JDBC driver name and database URL
    String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    String DB_URL = "jdbc:mysql://localhost:3306/atm_db";
    // Database credentials
    String USER = "root";
    String PASS = "jim123456";

    Connection conn = null;
    Statement stmt = null;

    public Connection getConnection() throws SQLException, ClassNotFoundException {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        return conn;
    }
}
```

implementation μεθόδων:

```
public void deposit(int x,String y) throws RemoteException, SQLException
{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    PreparedStatement prepstmt =conn.prepareStatement("UPDATE account_info SET balance= balance + ? WHERE account_no =? ");
    prepstmt.setInt(1,x);
    prepstmt.setString(2,y);
    prepstmt.executeUpdate();

    LocalDate localDate = LocalDate.now();
    java.util.Date date = Date.from(localDate.atStartOfDay(ZoneId.systemDefault()).toInstant());
    java.sql.Date sqlDate = new java.sql.Date(date.getTime());
    String query = " insert into transactions (account_no,type,amount,date)"
        + " values (?, 'deposit',?,?)";
    PreparedStatement prepstmt2 =conn.prepareStatement(query);
    prepstmt2.setString(1, y);
    prepstmt2.setInt(2, x);
    prepstmt2.setDate(3, sqlDate);
    prepstmt2.executeUpdate();
}
```

```

public void withdraw(int x,String y) throws RemoteException, SQLException
{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    PreparedStatement prepstmt =conn.prepareStatement("UPDATE account_info SET balance= balance - ? WHERE account_no=? ");
    prepstmt.setInt(1,x);
    prepstmt.setString(2,y);
    prepstmt.executeUpdate();

    LocalDate localDate = LocalDate.now();
    java.util.Date date = Date.from(LocalDate.atStartOfDay(ZoneId.systemDefault()).toInstant());
    java.sql.Date sqlDate = new java.sql.Date(date.getTime());

    String query = " insert into transactions (account_no,type,amount,date)"
        + " values (?, 'withdraw',?,?)";
    PreparedStatement prepstmt2 =conn.prepareStatement(query);
    prepstmt2.setString(1, y);
    prepstmt2.setInt(2, x);
    prepstmt2.setDate(3, sqlDate);
    prepstmt2.executeUpdate();
}

```

```

public int display_balance(String y) throws RemoteException, SQLException
{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    int balance=0;
    PreparedStatement prepstmt =conn.prepareStatement("SELECT balance FROM account_info WHERE account_no=? ");
    prepstmt.setString(1,y);
    ResultSet rs = prepstmt.executeQuery();
    while(rs.next()) {
        // Retrieve by column name
        balance = rs.getInt("balance");
    }
    rs.close();
    return balance;
}

```

```

public String transaction_details1( String y) throws RemoteException, SQLException
{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    String result="";
    String type=null;
    int amount=0;
    Date dt=null;
    PreparedStatement prepstmt =conn.prepareStatement("SELECT * FROM transactions WHERE account_no=? ");
    prepstmt.setString(1,y);
    ResultSet rs = prepstmt.executeQuery();
    while(rs.next()) {
        // Retrieve by column name
        type = rs.getString("type");
        amount = rs.getInt("amount");
        dt = rs.getDate("date");
        result += type + "\t" + amount + "\t" + dt.toString() + "\n" ;
    }
    rs.close();

    return result;
}

```

```

@Override
public String transaction_details2(String y, Date start_date, Date end_date)
    throws RemoteException, SQLException{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    String result="";
    String type=null;
    int amount=0;
    Date dt=null;
    PreparedStatement prepstmt =conn.prepareStatement("SELECT * FROM transactions WHERE account_no=? AND date>=? AND date<=? ");
    prepstmt.setString(1,y);
    prepstmt.setDate(2,(Date) start_date);
    prepstmt.setDate(3,(Date) end_date);
    ResultSet rs = prepstmt.executeQuery();
    while(rs.next()) {
        // Retrieve by column name
        type = rs.getString("type");
        amount = rs.getInt("amount");
        dt = rs.getDate("date");
        result += type + "\t" + amount + "\t" + dt.toString() + "\n" ;
    }
    rs.close();

    return result;
}

```



```

public boolean login(String acc,int pin) throws RemoteException,SQLException{
    int stored_pin= 000000000000000000000000;
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    PreparedStatement prepstmt =conn.prepareStatement("SELECT * FROM users WHERE account_no= ?");
    prepstmt.setString(1,acc);
    ResultSet rs = prepstmt.executeQuery();
    while(rs.next()) {
        // Retrieve by column name
        stored_pin = rs.getInt("pin");

    }
    rs.close();
    if(stored_pin != 000000000000000000000000) {
        if (stored_pin == pin) {
            return true;
        }
    }
    return false;
}

public void change_pin(String acc,int pin) throws RemoteException,SQLException{
    try {
        conn = getConnection();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    PreparedStatement prepstmt =conn.prepareStatement("UPDATE users SET pin= ? WHERE account_no=? ");
    prepstmt.setInt(1,pin);
    prepstmt.setString(2,acc);
    prepstmt.executeUpdate();
}
}

```

Ανάπτυξη server program:

Δημιουργούμε το απομακρυσμένο αντικείμενο με τη χρήση της implementation class και κάνουμε bind το αντικείμενο με την bind() στο registry.

```

import java.rmi.*;
public class ATM_Server
{
    public static void main(String args[])
    {
        try
        {
            //ATM_Implementation ATM=new ATM_Implementation();
            Registry reg = LocateRegistry.createRegistry(5000);
            reg.bind("atm", new ATM_Implementation());
        }
        catch(Exception ex)
        {
            System.out.println ("ATM failed with error: " + ex);
        }
    }
}

```

Το remote object δημιουργείται με την new ATM_Implementation() και είναι anonymous object.

Ανάπτυξη client program:

Επειδή έχει αρκετό κώδικα θα δείξουμε μόνο το πως καλούμε το remote object.

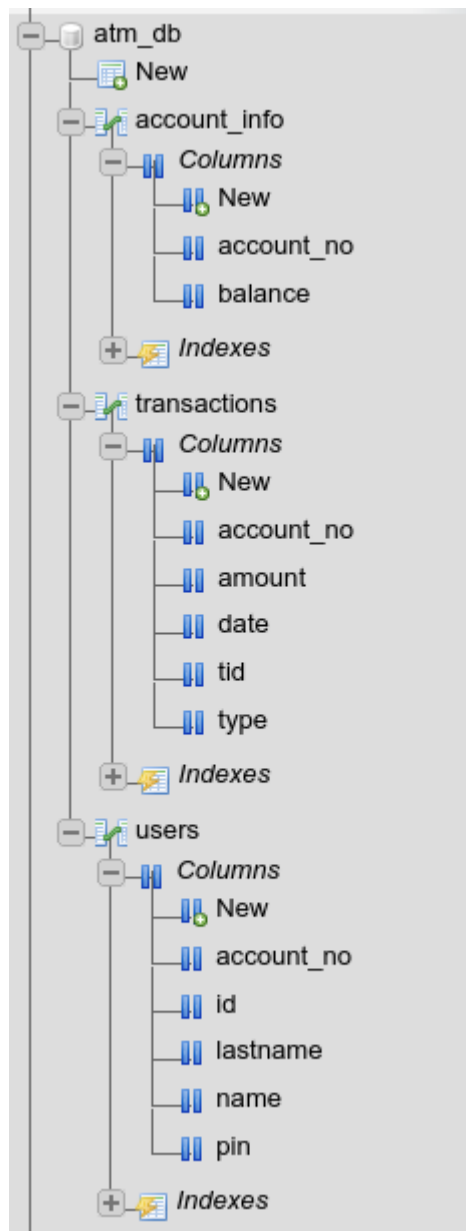
```

Scanner s=new Scanner(System.in);
String url="rmi://localhost:5000/atm";
ATM_Interface intf=(ATM_Interface)Naming.lookup(url);

```

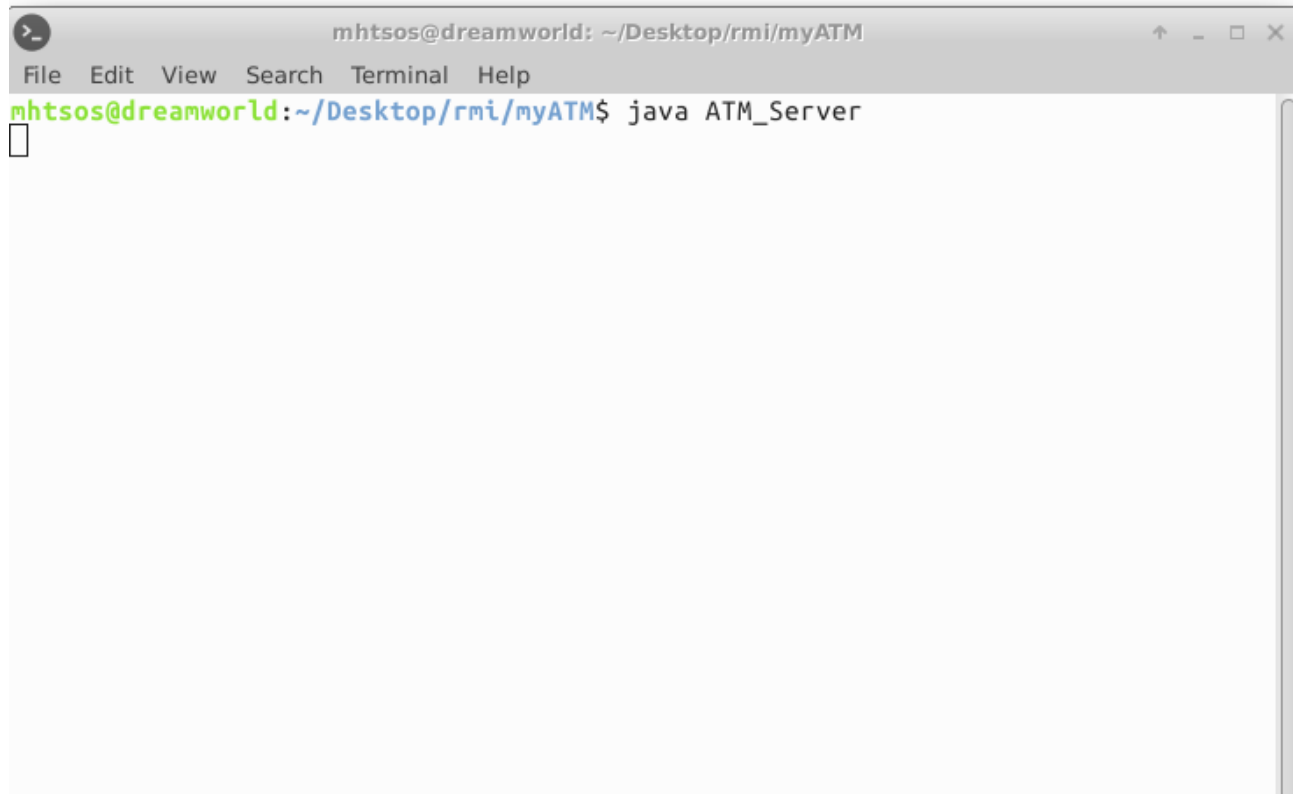
Το αντικείμενο intf καλεί τις απομακρυσμένες μεθόδους.

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ-MYSQL:



ΤΡΕΞΙΜΟ ΚΩΔΙΚΑ:

τρέχουμε τον σέρβερ.



```
mhtsos@dreamworld: ~/Desktop/rmi/myATM
File Edit View Search Terminal Help
mhtsos@dreamworld:~/Desktop/rmi/myATM$ java ATM_Server

```

Τρέχουμε τον client



```
mhtsos@dreamworld:~/Desktop/rmi/myATM$ java ATM_Client
Enter Account Number:

```

βάζουμε αριθμό λογαριασμού και pin
πατάμε από το 1-6 για τις λειτουργίες που θέλουμε.

```
Enter Account Number:
1234567890
Enter PIN:
4321
Logged in!
1. : Deposit Money
2. : Withdraw Money
3. : Check Balance
4. : Check Transaction History
5. : To Quit
6. : To change pin
□
```

Παράδειγμα εκτέλεσης των λειτουργιών , μετά απο κάθε λειτουργία ξανα εμφανίζει το μενού επιλογών ώσπου ο χρήστης πατήσει το 5 για έξοδο.

```
1
Enter Amount to be deposited :
100
```

```
2
Enter Amount to be withdrawn :
100
Withdrawn completed
```

```
3
Balance left in account number 1234567890 is : 30900
```

```
4
Enter 1 to view all transactions and 2 to view transactions within a range :
□
```

```

1
deposit 600      2017-12-12
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
deposit 100      2017-12-26
withdraw        100      2017-12-26
deposit 100      2018-01-20
withdraw        100      2018-01-20

```

- 1. : Deposit Money
- 2. : Withdraw Money
- 3. : Check Balance
- 4. : Check Transaction History
- 5. : To Quit
- 6. : To change pin

□

Enter 1 to view all transactions and 2 to view transactions within a range :

2

Enter Start Date (Eg. 2015-12-26):

2015-12-26

Enter End Date:

2018-12-12

```

deposit 600      2017-12-12
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
withdraw        100      2017-12-26
deposit 100      2017-12-26
withdraw        100      2017-12-26
deposit 100      2018-01-20
withdraw        100      2018-01-20

```

- 1. : Deposit Money
 - 2. : Withdraw Money
 - 3. : Check Balance
 - 4. : Check Transaction History
 - 5. : To Quit
-

```
6
Enter New pin:
4321
Re-enter New pin:
4321
Pin changed!
1. : Deposit Money
2. : Withdraw Money
3. : Check Balance
4. : Check Transaction History
5. : To Quit
6. : To change pin
```

```
—
```