**Name:** Bhavik Ransubhe

**Class :** TE (B) COMP

**ROLL NO:** 39055

--------------------------------------------------------------------------------------------------------------------------------------

**Title:**

Enhance the earlier system(BANK MANAGEMENT) with the help of socket programming use client server architecture.

**Objective:**

Understand the implementation of Client Server Architecture Using Socket Programming Concept

**Problem Statement:**

Write a Java Program with the help of Socket Programming using Cilent Server Architecture on earlier system (assignment1) .

**Outcomes :**

After completion of this assignment Students able to design and implement Client Server Architecture with help of Socket Programming.

**Software Requirements:**

Intellij IDEA community version .

**Hardware Requirements:**

PC/Laptop with min 4GB RAM , Intel core i3 processor ,500GB HDD ,128 GB SSD.

**Theory Concepts in brief:**

Java Socket Programming:

Java Socket programming is used for communication between the applications running on different JRE.

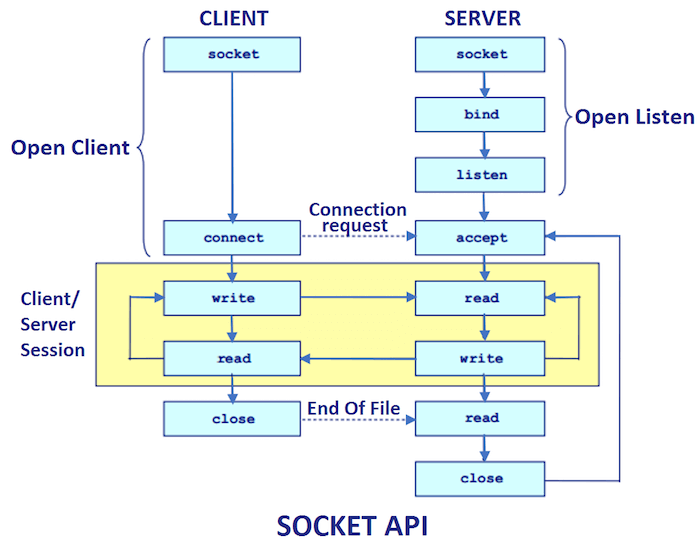
Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.



Socket class:

|  |  |
| --- | --- |
| Method | Description |
| 1) public InputStream getInputStream() | returns the InputStream attached with this socket. |
| 2) public OutputStream getOutputStream() | returns the OutputStream attached with this socket. |
| 3) public synchronized void close() | closes this socket |

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods:

ServerSocket class:

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods:

Sample Code for Server

• ServerSocket ss=new ServerSocket(6666);

• Socket s=ss.accept();//establishes connection

• DataInputStream dis=new DataInputStream(s.getInputStream());

• String str=(String)dis.readUTF();

• System.out.println("message= "+str); ss.close();

Sample Code for Client

• Socket s=new Socket("localhost",6666);

• DataOutputStream dout=new DataOutputStream(s.getOutputStream());

• dout.writeUTF("Hello Server");

• dout.flush();

• dout.close();

• s.close();

|  |  |  |
| --- | --- | --- |
| Method | | Description |
| 1) public Socket accept() | returns the socket and establish a connection between server and client. | |
| 2) public synchronized void close() | closes the server socket. | |

**Java DatagramSocket class**

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

Commonly used Constructors of DatagramSocket class

• DatagramSocket() throws SocketEeption: it creates a datagram socket and binds it with the available Port Number on the localhost machine.

• DatagramSocket(int port) throws SocketEeption: it creates a datagram socket and binds it with the given Port Number.

• DatagramSocket(int port, InetAddress address) throws SocketEeption: it creates a datagram socket and binds it with the specified port number and host address.

**Java DatagramPacket class**

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Commonly used Constructors of DatagramPacket class

• DatagramPacket(byte[] barr, int length): it creates a datagram packet. This constructor is used to receive the packets.

• DatagramPacket(byte[] barr, int length, InetAddress address, int port): it creates a datagram packet. This constructor is used to send the packets. Java InetAddress class represents an IP address.

**java.net.InetAddress** class provides methods to get the IP of any host name

• InetAddress ip=InetAddress.getByName("www.snjb.org”);

• System.out.println("Host Name: "+ip.getHostName());

• System.out.println("IP Address: "+ip.getHostAddress());

The **Java** **HttpURLConnection** class is http specific URLConnection. It works for HTTP protocol only.

By the help of HttpURLConnection class, you can information of any HTTP URL such as header information, status code, response code etc.

The java.net.HttpURLConnection is subclass of URLConnection class.

• URL url=new URL("http://www.snjb.org");

• HttpURLConnection huc=(HttpURLConnection)url.openConnection();

The **Java URLConnection** class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.

• URL url=new URL("http://www.snjb.org");

• URLConnection urlcon=url.openConnection();

• InputStream stream=urlcon.getInputStream();

**SerialVersionUID:**

The Serialization runtime associates a version number with each Serializable class called a SerialVersionUID, which is used during Deserialization to verify that sender and reciever of a serialized object have loaded classes for that object which are compatible with respect to serialization. If the reciever has loaded a class for the object that has different UID than that of corresponding sender’s class, the Deserialization will result in an InvalidClassException.

A Serializable class can declare its own UID explicitly by declaring a field name. It must be static, final and of type long. i.e- ANY-ACCESS-MODIFIER static final long serialVersionUID=42L;

If a serializable class doesn’t explicitly declare a serialVersionUID, then the serialization runtime will calculate a default one for that class based on various aspects of class, as described in Java Object Serialization Specification.However it is strongly recommended that all serializable classes explicitly declare serialVersionUID value, since its computation is highly sensitive to class details that may vary depending on compiler implementations, any change in class or using different id may affect the serialized data.

**Features of tool/framework/language used:**

The prime reason behind creation of Java was to bring portability and security features into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

**Platform Independent**

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into a platform specific machine, rather into platform-independent bytecode. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

**Simple**

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

**Secure**

With Java's secure features it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

**Architecture-neutral**

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of a Java runtime system.

**Portable**

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

**Robust**

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

**Multithreaded**

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

**Interpreted**

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

**High Performance**

With the use of Just-In-Time compilers, Java enables high performance.

**Distributed**

Java is designed for the distributed environment of the internet.

**Dynamic**

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

**Conclusion:**

In this assignment student are able to understand and implement Client Server Architecture using Socket Program concept.

**Program code (with proper comments, titles):**

**FILES:-**

**Client side:**

* **Client.java:**
* package com.company;  
  import java.io.\*;  
  import java.net.Socket;  
  import java.text.SimpleDateFormat;  
  import java.util.Scanner;  
  public class Client { //Class Client  
   Socket socket; //socket  
   ObjectOutputStream objectOutputStream; //objectOutputStream  
   OutputStream outputStream;  
   InputStream inputStream;  
   ObjectInputStream objectInputStream;  
   Scanner scanner=new Scanner(System.in);  
   DataInputStream dataInputStream;  
   DataOutputStream dataOutputStream;  
   Client(){ //Constructor  
   try {  
   socket=new Socket("localhost",1401); //connecting to localhost  
   dataInputStream=new DataInputStream(socket.getInputStream());  
   dataOutputStream=new DataOutputStream(socket.getOutputStream());  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   menu(); //Call Menu  
   }  
   public void menu(){  
   int ch;  
   do{  
   System.out.print("\n| WELCOME TO BR14x BANK SYSTEM |\nEnter Following Choice\n1)Register The User \n2)Login User \n3)Exit : "); //Portal  
   ch=scanner.nextInt();  
   switch (ch){  
   case 1: registerUser();  
   break;  
   case 2:loginUser();  
   break;  
   }  
   }while (ch<3);  
   }  
   public void registerUser(){  
   AccountInformation account= new AccountInformation(); //create object  
   account.getData(); //getData  
   try {  
   dataOutputStream.write(1);  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   try { //Send Object To Server  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(account);  
   String msg=dataInputStream.readUTF(); //Print Message From Server  
   System.out.print(msg);  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   }  
   public void loginUser(){ //login User  
   String accountNumber;  
   String accountPassword;  
   scanner.nextLine();  
   System.out.print("Enter Account Number :- ");  
   accountNumber=scanner.nextLine();  
   System.out.print("Enter Account Password :- ");  
   accountPassword=scanner.nextLine();  
   try { //Transfer Details  
   dataOutputStream.write(2);  
   dataOutputStream.writeUTF(accountNumber);  
   dataOutputStream.writeUTF(accountPassword);  
   int type=dataInputStream.read();  
   if(type==1){ //if Account Exist in System  
   inputStream = socket.getInputStream(); //Receive Account  
   objectInputStream = new ObjectInputStream(inputStream);  
   AccountInformation rAccount= (AccountInformation)  
   objectInputStream.readObject();  
   System.out.print("\nWelcome User !\n");  
   int ch;  
   do{ //Menu  
   System.out.print("\n\n1)Show Account Details \n2)Withdraw Amount\n3)Deposit Amount\n4)Update Account Details\n5)Transfer Amount\n6)Show Account Statement\n7)LogOut : ");  
   ch = scanner.nextInt();  
   switch (ch){  
   case 1:rAccount.showData(); //Show Data  
   break;  
   case 2:withdraw(rAccount); //Withdraw Amount  
   break;  
   case 3:deposit(rAccount); //Deposit Amount  
   break;  
   case 4:updateData(rAccount); //Update Data  
   break;  
   case 5:transferAmountF(rAccount); //Transfer Amount  
   break;  
   case 6:showStatement(rAccount); //Show Statement  
   break;  
   case 7:rAccount=null; //Logout  
   break;  
   }  
   }while(ch<7);  
   }  
   else{  
   System.out.print("Invalid Credentials"); //If account doesnt Exist  
   }  
   } catch (IOException | ClassNotFoundException e) {  
   e.printStackTrace();  
   }  
   }  
   public void transferAmountF(AccountInformation information) { //TransferAmount  
   Scanner scanner=new Scanner(System.in);  
   int accNo;  
   double amount;  
   String msg="@";  
   System.out.print("\nEnter Account Number Of User Whom You Want To Transfer Amount :"); //Ask Acc Number  
   accNo = scanner.nextInt();  
   System.out.print("Enter Amount :"); //Ask amount  
   amount = scanner.nextDouble();  
   TransferAmount amount1=new  
   TransferAmount(information.accountNumber,accNo,amount); //Send TransferRequest  
   try { //Send Object  
   dataOutputStream.write(4);  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(amount1);  
   msg=dataInputStream.readUTF(); //Get Status From Server  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   System.out.print(msg);  
   }  
   public void withdraw(AccountInformation information) { //Ask Amount To Withdraw  
   Scanner scanner=new Scanner(System.in);  
   double amount = information.accountBalance;  
   double withdrawAmount;  
   System.out.print("Enter The Amount You Withdraw : "); //Ask Amount  
   withdrawAmount = scanner.nextDouble();  
   if (withdrawAmount > amount+3000) { //Check Limit  
   System.out.println("Account Balance is low to withdraw amount " +  
   withdrawAmount);  
   } else {  
   amount -= withdrawAmount;  
   System.out.println("Amount Withdrawn \nAccount Balance : " +  
   amount);  
   information.setAccountBalance(amount); //Update Balance  
   SimpleDateFormat formattedDate = new SimpleDateFormat("dd:MM:yyyy hh:mm a");  
   String date = formattedDate.format(System.currentTimeMillis());  
   Transactions Trans = new Transactions(date, withdrawAmount,  
   "Withdraw", amount, "Withdrawn Self"); //Create Statement  
   information.transactionStatement.add(Trans);  
   try { //Send Object  
   dataOutputStream.write(3);  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(information);  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   }  
   }  
   public void showStatement(AccountInformation information) { //Show Account Statement  
   for (int i = 0; i < 110; i++)  
   System.out.print('\_');  
   System.out.println();  
   System.out.format("%1$-21s%2$-10s%3$-12s%4$-10s%5$-21s", "Date", "Amount", "Type", "Balance", "Description");  
   System.out.println();  
   for (int i = 0; i < 110; i++)  
   System.out.print('\_');  
   System.out.println();  
   for (int i = 0; i < information.transactionStatement.size(); i++) {  
   if(i!=0) {  
   System.out.println();  
   }  
   Transactions trans = information.transactionStatement.get(i);  
   System.out.format("%1$-21s%2$-10s%3$-12s%4$-10s%5$-21s",  
   trans.getDate(),  
   trans.getAmount(),  
   trans.getType(),  
   trans.getTotalBalance(),  
   trans.getDescription());  
   }  
   System.out.println();  
   for (int i = 0; i < 110; i++)  
   System.out.print('\_');  
   }  
   public void deposit(AccountInformation information) { //Deposit Amount  
   Scanner scanner=new Scanner(System.in);  
   double amount = information.accountBalance;  
   double withdrawAmount;  
   System.out.print("Enter The Amount You Deposit : "); //Ask amount  
   withdrawAmount = scanner.nextDouble();  
   amount += withdrawAmount;  
   System.out.println("Amount Deposited \nAccount Balance : " +  
   amount); //Ask Balance  
   information.setAccountBalance(amount);  
   SimpleDateFormat formattedDate = new SimpleDateFormat("dd:MM:yyyy hh:mm a");  
   String date = formattedDate.format(System.currentTimeMillis());  
   Transactions Trans = new Transactions(date, withdrawAmount,  
   "Deposit", amount, "Deposited Self"); //Create Statement  
   information.transactionStatement.add(Trans);  
   try { //Send Object  
   dataOutputStream.write(3);  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(information);  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   }  
   public void updateData(AccountInformation information) { //Update Data  
   Scanner scanner = new Scanner(System.in);  
   int ch;  
   String field2;  
   int ch2;  
   boolean process=true;  
   do { //Ask Field To Update  
   System.out.print("\nEnter Field To Change\n1)Name\n2)DOB\n3)PAN\n4)Address\n5)Nominee\n6)PhoneNumber\n7)Aadhar No\n8)EXIT : ");  
   ch = scanner.nextInt();  
   switch (ch) {  
   case 1: //Change Name  
   scanner.nextLine();  
   System.out.println("Current Name :" +  
   information.getUserName());  
   System.out.print("Enter Change Name To :");  
   field2 = scanner.nextLine();  
   System.out.print("Change Name [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
   information.setUserName(field2);  
   } else {  
   System.out.print("Name Unchanged");  
   }  
   break;  
   case 2: //Change DOB  
   scanner.nextLine();  
   System.out.println("Current DOB :" +  
   information.getUserDOB());  
   System.out.print("Enter Change DOB To :");  
   field2 = scanner.nextLine();  
   System.out.print("Change DOB [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
   information.setUserDOB(field2);  
   } else {  
   System.out.print("DOB Unchanged");  
   }  
   break;  
   case 3: //Change PAN  
   scanner.nextLine();  
   System.out.println("Current PAN :" +  
   information.getUserPAN());  
   System.out.print("Enter Change PAN To :");  
   field2 = scanner.nextLine();  
   System.out.print("Change PAN [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
   information.setUserPAN(field2);  
   } else {  
   System.out.print("PAN Unchanged");  
   }  
   break;  
   case 4: //Change Address  
   scanner.nextLine();  
   System.out.println("Current Address :" +  
   information.getUserAddress());  
   System.out.print("Enter Change Address :");  
   field2 = scanner.nextLine();  
   System.out.print("Change Address [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
   information.setUserAddress(field2);  
   } else {  
   System.out.print("Address Unchanged");  
   }  
   break;  
   case 5: //Change Nominee  
   scanner.nextLine();  
   System.out.println("Current Nominee :" +  
   information.getAccountNominee());  
   System.out.print("Enter Change Nominee :");  
   field2 = scanner.nextLine();  
   System.out.print("Change Nominee [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
   information.setAccountNominee(field2);  
   } else {  
   System.out.print("Nominee Unchanged");  
   }  
   break;  
   case 6: //Change Phone Number  
   scanner.nextLine();  
   System.out.println("Current Phone Number :" +  
   information.getUserPhoneNumber());  
   boolean stat = false;  
   do {  
   System.out.print("Enter Change Phone Number To : ");  
   String phoneNum = scanner.nextLine();  
   if (phoneNum.length() != 10) {  
   System.out.print("Phone Number Is Not Valid");  
   System.out.println();  
   stat = true;  
   } else {  
   for (int i = 0; i < 10; i++) { //take proper phonenumber  
   assert phoneNum != null;  
   if (phoneNum.charAt(i) >= '0' &&  
   phoneNum.charAt(i) <= '9') {  
   System.out.print("Change Phone [1/0]? :");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
    
   information.setUserPhoneNumber(phoneNum);  
   } else {  
   System.out.print("Phone Number Unchanged");  
   }  
   stat = false;  
   } else {  
   System.out.print("Phone Number Should Not Contain Any Letters");  
   System.out.println();  
   phoneNum = null;  
   stat = true;  
   }  
   }  
   }  
   } while (stat);  
   break;  
   case 7: //Change Aadhar Number  
   scanner.nextLine();  
   System.out.println("Current Aadhar :" +  
   information.getUserAadharNumber());  
   stat = false;  
   do {  
   System.out.print("Enter Aadhar : ");  
   String userAadhar = scanner.nextLine();  
   if (userAadhar.length() != 12) {  
   System.out.print("Aadhar Number Is Not Valid.");  
   System.out.println();  
   stat = true;  
   } else {  
   for (int i = 0; i < 12; i++) { //take proper aadhar number  
   assert userAadhar != null;  
   if (userAadhar.charAt(i) >= '0' &&  
   userAadhar.charAt(i) <= '9') {  
   System.out.print("Change Aadhar [1/0]?:");  
   ch2 = scanner.nextInt();  
   if (ch2 == 1) {  
    
   information.setUserAadharNumber(userAadhar);  
   } else {  
   System.out.print("Aadhar Number Unchanged");  
   }  
   stat = false;  
   } else {  
   System.out.print("Aadhar Should Not Contain Any Letters");  
   System.out.println();  
   userAadhar = null;  
   stat = true;  
   }  
   }  
   }  
   } while (stat);  
   break;  
   case 8:  
   try { //Update in Server  
   dataOutputStream.write(3);  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new  
   ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(information);  
   } catch (IOException e) {  
   e.printStackTrace();  
   }  
   System.out.print("Updation Process Completed !\n");  
   information.showData();  
   process=false;  
   }  
   } while (process);  
   }  
   public static void main(String[] args){  
   new Client(); //Initialize Object  
   }  
  }
* **AccountInfromation.java:**
* package com.company;  
  import java.io.Serializable;  
  import java.util.ArrayList;  
  import java.util.Scanner;  
  import java.util.regex.Matcher;  
  import java.util.regex.Pattern;  
  public class AccountInformation implements Serializable { //Account Information Class  
   private static final long serialVersionUID = 6128016096756071380L;  
   int accountNumber; //Class Fields  
   String userName;  
   String userDOB;  
   String userPAN;  
   String userAddress;  
   String accountNominee;  
   String userPhoneNumber;  
   String userAadharNumber;  
   double accountBalance;  
   String accountPassword;  
   public ArrayList<Transactions> transactionStatement;  
   public AccountInformation() {  
   }  
   public void showData(){ //Show Data Function  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   System.out.println(); //Format  
   System.out.format("%1$-13s%2$-22s%3$-15s%4$-15s%5$-12s%6$-12s%7$-30s%8$-20s%9$-17s", "Account No", "Name", "Phone Number", "Aadhar No", "PAN No", "DOB", "Address", "Nominee", "Account Balance");  
   System.out.println();  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   System.out.println();  
   System.out.format("%1$-13s%2$-22s%3$-15s%4$-15s%5$-12s%6$-12s%7$-30s%8$-20s%9$-17s", accountNumber, userName, userPhoneNumber, userAadharNumber, userPAN, userDOB, userAddress, accountNominee, accountBalance);  
   System.out.println();  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   }  
   public void getData(){ //Get Data Function  
   Scanner scanner=new Scanner(System.in);  
   boolean stat = false; //Ask data to User  
   System.out.println("Please Fill Out Following Details :\n");  
   System.out.print("Enter Full Name : ");  
   userName=scanner.nextLine();  
   System.out.print("Enter Date Of Birth : ");  
   userDOB= scanner.nextLine();  
   System.out.print("Enter Address : ");  
   userAddress=scanner.nextLine();  
   System.out.print("Enter Nominee : ");  
   accountNominee = scanner.nextLine();  
   System.out.print("Enter PAN :");  
   userPAN = scanner.nextLine();  
   do { //Take Appropriate Phone Number  
   System.out.print("Enter Phone Number : ");  
   userPhoneNumber = scanner.nextLine();  
   if(userPhoneNumber.length()!=10){  
   System.out.print("Phone Number Is Not Valid");  
   System.out.println();  
   stat=true;  
   userPhoneNumber=null;  
   }  
   else{  
   for(int i=0;i<10;i++){  
   assert userPhoneNumber != null;  
   if(userPhoneNumber.charAt(i) >='0' &&  
   userPhoneNumber.charAt(i)<='9'){  
   stat=false;  
   }  
   else{  
   System.out.print("Phone Number Should Not Contain Any Letters");  
   System.out.println();  
   userPhoneNumber=null;  
   stat=true;  
   }  
   }  
   }  
   }while(stat);  
   do { //Take Appropriate Aadhar Number  
   System.out.print("Enter Aadhar : ");  
   userAadharNumber=scanner.nextLine();  
   if(userAadharNumber.length()!=12){  
   System.out.print("Aadhar Number Is Not Valid.");  
   System.out.println();  
   stat=true;  
   userAadharNumber=null;  
   }  
   else{  
   for(int i=0;i<12;i++){  
   assert userAadharNumber != null;  
   if(userAadharNumber.charAt(i) >='0' &&  
   userAadharNumber.charAt(i)<='9'){  
   stat=false;  
   }  
   else{  
   System.out.print("Aadhar Should Not Contain Any Letters");  
   System.out.println();  
   userAadharNumber=null;  
   stat=true;  
   }  
   }  
   }  
   }while(stat);  
   do { //Take Balance Above 3000  
   System.out.print("Enter Initial Balance (Must Be Above Rs.10,000) : ");  
   accountBalance = scanner.nextDouble();  
   }while(accountBalance<10000);  
   System.out.print("Creating An Account.....");  
   scanner.nextLine();  
   boolean stat1;  
   String pass1;  
   do {  
   System.out.print("\nPlease Enter New And Unique Password : ");  
   pass1 = scanner.nextLine();  
   if (pass1.length() >= 8) {  
   Pattern letter = Pattern.compile("[a-zA-z]");  
   Pattern digit = Pattern.compile("[0-9]");  
   Pattern special = Pattern.compile("[!@#$%&\*()\_+=|<>?{}\\[\\]~-]");  
   Matcher hasLetter = letter.matcher(pass1);  
   Matcher hasDigit = digit.matcher(pass1);  
   Matcher hasSpecial = special.matcher(pass1);  
   if (hasDigit.find() && hasLetter.find() && hasSpecial.find())  
   {  
   stat1=false;  
   } else {  
   System.out.println("Should be of minimum size 8 with Letter, Digit And Special Characters"); //Take Password In Format  
   pass1 = null;  
   stat1=true;  
   }  
   } else {  
   System.out.println("Invalid Pass. Should be of minimum size 8 ");  
   stat1=true;  
   }  
   }while (stat1);  
   String pass2;  
   do {  
   System.out.print("Please Enter Password Again To Confirm :");  
   pass2 = scanner.nextLine();  
   if (pass2.isEmpty()) {  
   System.out.println("Password cant be empty");  
   } else {  
   if (pass1.equals(pass2)) {  
   accountPassword = pass1;  
   } else {  
   System.out.println("Password Do Not Match");  
   }  
   }  
   }while (!pass1.equals(pass2));  
   transactionStatement=new ArrayList<>();  
   }  
   //getter Setter Function  
   public void setUserName(String userName) {  
   this.userName = userName;  
   }  
   public void setUserDOB(String userDOB) {  
   this.userDOB = userDOB;  
   }  
   public void setUserPAN(String userPAN) {  
   this.userPAN = userPAN;  
   }  
   public void setUserAddress(String userAddress) {  
   this.userAddress = userAddress;  
   }  
   public void setAccountNominee(String accountNominee) {  
   this.accountNominee = accountNominee;  
   }  
   public void setUserPhoneNumber(String userPhoneNumber) {  
   this.userPhoneNumber = userPhoneNumber;  
   }  
   public void setUserAadharNumber(String userAadharNumber) {  
   this.userAadharNumber = userAadharNumber;  
   }  
   public void setAccountBalance(double accountBalance) {  
   this.accountBalance = accountBalance;  
   }  
   public String getUserName() {  
   return userName;  
   }  
   public String getUserDOB() {  
   return userDOB;  
   }  
   public String getUserPAN() {  
   return userPAN;  
   }  
   public String getUserAddress() {  
   return userAddress;  
   }  
   public String getAccountNominee() {  
   return accountNominee;  
   }  
   public String getUserPhoneNumber() {  
   return userPhoneNumber;  
   }  
   public String getUserAadharNumber() {  
   return userAadharNumber;  
   }  
  }
* **Transactions.java:**
* package com.company;  
  import java.io.Serializable;  
  public class Transactions implements Serializable { //Class For Transaction  
   private static final long serialVersionUID = 6128016096756071380L;  
   private String date; //Statement Fields  
   private double amount;  
   private String type;  
   private double totalBalance;  
   private String description;  
   public Transactions(String date, double amount, String type, double  
   totalBalance, String description) { // Transaction Constructor  
   this.date = date;  
   this.amount = amount;  
   this.type = type;  
   this.totalBalance = totalBalance;  
   this.description = description;  
   }  
   //getter Setter Functions  
   public String getDate() {  
   return date;  
   }  
   public double getAmount() {  
   return amount;  
   }  
   public String getType() {  
   return type;  
   }  
   public double getTotalBalance() {  
   return totalBalance;  
   }  
   public String getDescription() {  
   return description;  
   }  
  }
* **TransferAmount.java**
* package com.company;  
  import java.io.Serializable;  
  public class TransferAmount implements Serializable { //Class to Transfer Amount  
   private static final long serialVersionUID = 6128016096756071380L;  
   int accountNumberSender,accountNumberReceiver;  
   double amount;  
   public TransferAmount(int accountNumberSender, int accountNumberReceiver,  
   double senderAmount) { //Constructor  
   this.accountNumberSender = accountNumberSender;  
   this.accountNumberReceiver = accountNumberReceiver;  
   this.amount = senderAmount;  
   }  
  }

**Server side:**

* **Server.java:**
* package com.company;  
  import java.io.\*;  
  import java.net.ServerSocket;  
  import java.net.Socket;  
  import java.net.SocketException;  
  import java.text.SimpleDateFormat;  
  import java.util.ArrayList;  
  import static java.lang.System.exit;  
  public class Server {  
   private static int latestAccNo = 14010;  
   private ArrayList<AccountInformation> accountData;  
   ServerSocket serverSocket;  
   Socket socket;  
   InputStream inputStream;  
   OutputStream outputStream;  
   ObjectInputStream objectInputStream;  
   ObjectOutputStream objectOutputStream;  
   DataInputStream dataInputStream;  
   DataOutputStream dataOutputStream;  
   Server(){ //Constructor  
   accountData=new ArrayList<>();  
   System.out.print("");  
   try {  
   serverSocket = new ServerSocket(1401); //Create Socket  
   socket = serverSocket.accept();  
   if(socket.isConnected()){ //Check If Client is Connected  
   System.out.print("\nClient is SuccessfullyConnected");  
   }  
   //initialize input and output streams  
   dataInputStream=new DataInputStream(socket.getInputStream());  
   dataOutputStream=new DataOutputStream(socket.getOutputStream());  
   } catch (SocketException s){ //catch exceptions  
   System.out.print("Client Disconnected !");  
   showData(1);  
   }  
   catch (IOException e) {  
   e.printStackTrace();  
   }  
   try {  
   menu(); //call Menu  
   } catch (SocketException s){ //Handle Exception  
   System.out.println();  
   System.out.print("Accounts :\n");  
   showData(1);  
   } catch (IOException | ClassNotFoundException e) {  
   e.printStackTrace();  
   }  
   }  
   public void showData(int type){ //Show Data  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   System.out.println();  
   System.out.format("%1$-13s%2$-22s%3$-15s%4$-15s%5$-12s%6$-12s%7$-30s%8$-20s%9$-17s", //format  
   "Account No", "Name", "Phone Number", "Aadhar No", "PAN No",  
   "DOB", "Address", "Nominee", "Account Balance");  
   System.out.println();  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   System.out.println();  
   for (AccountInformation accountDatum : accountData) {  
   System.out.format("%1$-13s%2$-22s%3$-15s%4$-15s%5$-12s%6$-12s%7$-30s%8$-20s%9$-17s",  
   accountDatum.accountNumber,  
   accountDatum.userName,  
   accountDatum.userPhoneNumber,  
   accountDatum.userAadharNumber,  
   accountDatum.userPAN,  
   accountDatum.userDOB,  
   accountDatum.userAddress,  
   accountDatum.accountNominee,  
   accountDatum.accountBalance);  
   System.out.println();  
   }  
   System.out.println();  
   System.out.println();  
   for (int i = 0; i < 155; i++)  
   System.out.print('\_');  
   if(type==1){  
   exitProgram();  
   }  
   }  
   void exitProgram(){ //Exit Program  
   exit(0);  
   }  
   void menu() throws IOException, ClassNotFoundException { //Menu  
   int ch;  
   do{  
   ch = dataInputStream.read();  
   switch (ch) {  
   case 1: createAccount();  
   break;  
   case 2:authenticateAccount();  
   break;  
   case 3: updateAccount();  
   break;  
   case 4: transferAmount();  
   break;  
   }  
   }while (true);  
   }  
   public void createAccount() throws IOException, ClassNotFoundException {  
  //Create Account  
   inputStream = socket.getInputStream();  
   objectInputStream = new ObjectInputStream(inputStream);  
   AccountInformation account=(AccountInformation)  
   objectInputStream.readObject(); //Get Object from Client  
   int accNo = generateAccNo(); //assign new Account Number  
   account.accountNumber = accNo;  
   accountData.add(account);  
   System.out.print("\nAccount Created Successfully");  
   dataOutputStream.writeUTF("Account Created !\n Account Number : "+"\033[0;31m"+accNo+"\033[0m"+" And Password : "+"\033[0;31m"+account.accountPassword+"\033[0m");  
   //convey User Login Details  
   }  
   public void authenticateAccount() throws IOException { //AuthenticateClient  
   String accNum=dataInputStream.readUTF(); //Receive AccountNumber  
   String pass=dataInputStream.readUTF(); // Receive Password  
   int index ;  
   int accNo=Integer.parseInt(accNum); //Check if account Exist  
   index=getAccountIndex(accNo);  
   if(index!=-1){ //of account Exist  
   dataOutputStream.write(1);  
   AccountInformation information;  
   if(pass.equals(accountData.get(index).accountPassword)){ //isPassword Matches  
   information=accountData.get(index);  
   outputStream=socket.getOutputStream();  
   objectOutputStream=new ObjectOutputStream(outputStream);  
   objectOutputStream.writeObject(information); //Send The User'sAccount Info  
   }  
   else{  
   dataOutputStream.write(2);//if Password Dont match , sendError  
   }  
   }else{  
   dataOutputStream.write(2); //if Account Does Not Exist , sendError  
   }  
   }  
   public void updateAccount() throws IOException, ClassNotFoundException {  
  //Update Information  
   inputStream = socket.getInputStream();  
   objectInputStream = new ObjectInputStream(inputStream);  
   AccountInformation account=(AccountInformation)  
   objectInputStream.readObject(); //Receive updated object  
   int accNo=account.accountNumber;  
   int index=getAccountIndex(accNo); //get Account Index  
   accountData.set(index,account); //update account on Server side  
   }  
   public void transferAmount() throws IOException, ClassNotFoundException {  
  //transfer Account  
   inputStream = socket.getInputStream();  
   objectInputStream = new ObjectInputStream(inputStream);  
   TransferAmount amountT=(TransferAmount)objectInputStream.readObject();  
  //get Transaction Information  
   int indexS=getAccountIndex(amountT.accountNumberSender);  
   int indexR=getAccountIndex(amountT.accountNumberReceiver);  
   double amountTransfer=amountT.amount;  
   double Amt1=accountData.get(indexS).accountBalance;  
   double Amt2=accountData.get(indexR).accountBalance;  
   if (Amt1 > amountTransfer + 3000.0) { //check minimum balance limit  
   Amt1 -= amountTransfer;  
   Amt2 += amountTransfer;  
   accountData.get(indexS).setAccountBalance(Amt1);//update amount inaccount1  
   accountData.get(indexR).setAccountBalance(Amt2);//update amount inaccount2  
   SimpleDateFormat formattedDate = new SimpleDateFormat("dd:MM:yyyy hh:mm a");  
   String date = formattedDate.format(System.currentTimeMillis());  
   Transactions Trans = new Transactions(date, amountTransfer,  
   "Withdraw", Amt1, "Transferred To " + accountData.get(indexR).getUserName());  
   accountData.get(indexS).transactionStatement.add(Trans); //updateTransaction Statement  
   Transactions Trans1 = new Transactions(date, amountTransfer,  
   "Deposit", Amt2, "Transferred From " + accountData.get(indexS).getUserName());  
   accountData.get(indexR).transactionStatement.add(Trans1); //updateTransaction Statement  
   dataOutputStream.writeUTF("Transferred The Money"); //Convey UserThat Money is Transferred  
   } else {  
   dataOutputStream.writeUTF("You Dont Have Enough Balance To Transfer The Money"); //Low Balance Error  
   }  
   }  
   public int getAccountIndex(int accNo){ // function to search Account inArray List  
   int index=-1;  
   boolean stat=false;  
   for (int i=0;i<accountData.size();i++) {  
   if (accountData.get(i).accountNumber == accNo) { //if accountExist  
   index=i;  
   stat = true;  
   break;  
   }  
   }  
   if(stat){  
   return index; //return index  
   }  
   return -1; //else return error  
   }  
   public static int generateAccNo() { //generate Account Number  
   return ++latestAccNo;  
   }  
   public static void main(String[] args){  
   new Server(); //Initialize Object  
   }  
  }
* **AccountInfromation.java:**
* package com.company;  
  import java.io.Serializable;  
  import java.util.ArrayList;  
  public class AccountInformation implements Serializable { //AccountInformation Class  
   private static final long serialVersionUID = 6128016096756071380L;  
   //Data Fields  
   int accountNumber;  
   String  
   userName,userDOB,userPAN,userAddress,accountNominee,userPhoneNumber,userAadharNumber,accountPassword;  
   double accountBalance;  
   ArrayList<Transactions> transactionStatement;  
   //getter steer Functions  
   public void setAccountBalance(double accountBalance) {  
   this.accountBalance = accountBalance;  
   }  
   public String getUserName() {  
   return userName;  
   }  
  }
* **Transactions.java:**
* package com.company;  
  import java.io.Serializable;  
  public class Transactions implements Serializable { //Transaction Class  
   private static final long serialVersionUID = 6128016096756071380L;  
   // Data fields  
   private String date;  
   private double amount;  
   private String type;  
   private double totalBalance;  
   private String description;  
   public Transactions(String date, double amount, String type, double  
   totalBalance, String description) {  
   this.date = date;  
   this.amount = amount;  
   this.type = type;  
   this.totalBalance = totalBalance;  
   this.description = description;  
   }  
  }
* **TransferAmount.java**
* package com.company;  
  import java.io.Serializable;  
  public class TransferAmount implements Serializable { //Transfer Amount Class  
   private static final long serialVersionUID = 6128016096756071380L;  
   int accountNumberSender,accountNumberReceiver; // fields  
   double amount;  
   public TransferAmount(int accountNumberSender, int accountNumberReceiver, double amount) {  
   this.accountNumberSender = accountNumberSender;  
   this.accountNumberReceiver = accountNumberReceiver;  
   this.amount = amount;  
   }  
  }

**Output:**

* **Client side:**

| WELCOME TO BR14x BANK SYSTEM |

Enter Following Choice

1)Register The User

2)Login User

3)Exit : 1

Please Fill Out Following Details :

Enter Full Name : Bhavik

Enter Date Of Birth : 14-01-2000

Enter Address : Pune,India

Enter Nominee : Girish

Enter PAN :EMP1401BR

Enter Phone Number : 3723081038

Enter Aadhar : 13131730013131414

Aadhar Number Is Not Valid.

Enter Aadhar : 123456654321

Enter Initial Balance (Must Be Above Rs.10,000) : 15000

Creating An Account.....

Please Enter New And Unique Password : Bh@vik14

Please Enter Password Again To Confirm :Bh@vik14

Account Created !

Account Number : 14011 And Password : Bh@vik14

| WELCOME TO BR14x BANK SYSTEM |

Enter Following Choice

1)Register The User

2)Login User

3)Exit : 1

Please Fill Out Following Details :

Enter Full Name : Srivatsa Ransubhe

Enter Date Of Birth : 21-12-1997

Enter Address : London,Uk

Enter Nominee : Rasana

Enter PAN :EMP2112Sr

Enter Phone Number : 8080808080

Enter Aadhar : 987654456789

Enter Initial Balance (Must Be Above Rs.10,000) : 30000

Creating An Account.....

Please Enter New And Unique Password : Sr!@2112

Please Enter Password Again To Confirm :Sr!@2112

Account Created !

Account Number : 14012 And Password : Sr!@2112

| WELCOME TO BR14x BANK SYSTEM |

Enter Following Choice

1)Register The User

2)Login User

3)Exit : 2

Enter Account Number :- 14011

Enter Account Password :- Bh@vik14

Welcome User !

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Account No Name Phone Number Aadhar No PAN No DOB Address Nominee Account Balance

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14011 Bhavik 3723081038 123456654321 EMP1401BR 14-01-2000 Pune,India Girish 15000.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 2

Enter The Amount You Withdraw : 1000

Amount Withdrawn

Account Balance : 14000.0

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 3

Enter The Amount You Deposit : 6000

Amount Deposited

Account Balance : 20000.0

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 4

Enter Field To Change

1)Name

2)DOB

3)PAN

4)Address

5)Nominee

6)PhoneNumber

7)Aadhar No

8)EXIT : 1

Current Name :Bhavik

Enter Change Name To :Bhavik Ransubhe

Change Name [1/0]? :1

Enter Field To Change

1)Name

2)DOB

3)PAN

4)Address

5)Nominee

6)PhoneNumber

7)Aadhar No

8)EXIT : 8

Updation Process Completed !

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Account No Name Phone Number Aadhar No PAN No DOB Address Nominee Account Balance

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14011 Bhavik Ransubhe 3723081038 123456654321 EMP1401BR 14-01-2000 Pune,India Girish 20000.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 5

Enter Account Number Of User Whom You Want To Transfer Amount :14012

Enter Amount :2000

Transferred The Money

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 6

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date Amount Type Balance Description

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

21:11:2020 09:57 pm 1000.0 Withdraw 14000.0 Withdrawn Self

21:11:2020 09:58 pm 6000.0 Deposit 20000.0 Deposited Self

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)LogOut : 7

| WELCOME TO BR14x BANK SYSTEM |

Enter Following Choice

1)Register The User

2)Login User

3)Exit : 3

Process finished with exit code 0

* **Server side:**

Client is Successfully Connected

Account Created Successfully

Account Created Successfully

Accounts :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Account No Name Phone Number Aadhar No PAN No DOB Address Nominee Account Balance

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

14011 Bhavik Ransubhe 3723081038 123456654321 EMP1401BR 14-01-2000 Pune,India Girish 18000.0

14012 Srivatsa Ransubhe 8080808080 987654456789 EMP2112Sr 21-12-1997 London,Uk Rasana 32000.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

END

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*