NAME: Bhavik Ransubhe

CLASS : TE (B) COMP

ROLL NO : 39055

**Title :**

Enhance above system by using JDBC, Multithreading, concurrency, synchronous and asynchronous callbacks, Thread Pools using Executor Service.

**Objective:**

To learn Java database connectivity.

To learn Multithreading, concurrency, synchronous and asynchronous callbacks.

To learn Thread Pools using Executor Service.

**Problem Statement:**

Write a Java program with the help of JDBC and Multithreading concurrency, synchronous and asynchronous callbacks, Thread Pools using Executor Service to enhance the previos system (Bank Management)

**Outcomes:**

After completion of assignment student will be able to implement the concept of JDBC and Multithreading Concept in easy ways.

**Software Requirements:**

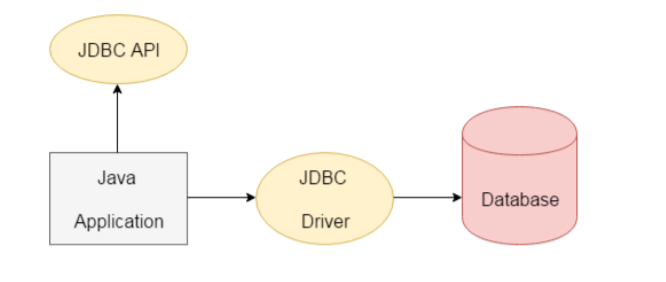
Intellij IDEA Community Edition 2020.1.1

**Hardware Requirement:**

PC/Laptop with min 4GB RAM, 500 GB HDD, Intel Core i5 processor .

**Theory Concepts in brief:**

Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.



Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

**JDBC Driver**

JDBC Driver is a software component that e

nables java application to interact with the database.There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver

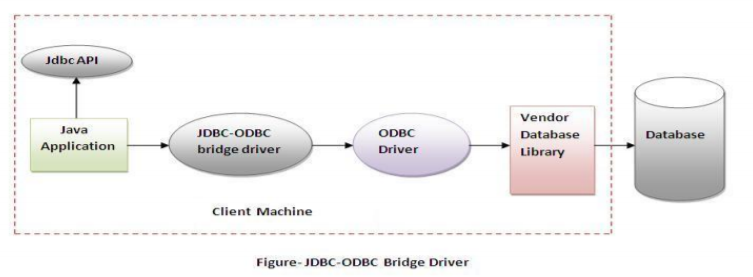
2. Native-API driver (partially java driver)

3. Network Protocol driver (fully java driver)

4. Thin driver (fully java driver)

**JDBC-ODBC bridge driver**

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.



**Advantages:**

• easy to use.

• can be easily connected to any database.

**Disadvantages:**

• Performance degraded because JDBC method call is converted into the ODBC function calls.

• The ODBC driver needs to be installed on the client machine.

There are 5 steps to connect any java application with the database in java using JDBC. They are as follows:

• Register the driver class

• Creating connection

• Creating statement

• Executing queries

• Closing connection

**Common JDBC Components :**

The JDBC API provides the following interfaces and classes −

1. DriverManager : This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.

2. Driver: This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.

3. Connection: This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.

4. Statement: You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.

5. ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.

6. SQLException: This class handles any errors that occur in a database application.

**Multithreading in Java**

• Multithreading in java is a process of executing multiple threads simultaneously.

• Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

• But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

• Java Multithreading is mostly used in games, animation etc

**Advantages of Java Multithreading**

1) It doesn't block the user because threads are independent and you can perform multiple operations at same time.

2) You can perform many operations together so it saves time.

3) Threads are independent so it doesn't affect other threads if exception occur in a single thread.

**Multitasking**

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved by two ways:

• Process-based Multitasking(Multiprocessing)

• Thread-based Multitasking(Multithreading)

**Process-based Multitasking (Multiprocessing)**

• Each process have its own address in memory i.e. each process allocates separate memory area.

• Process is heavyweight.

• Cost of communication between the process is high.

• Switching from one process to another require some time for saving and loading registers, memory maps, updating lists etc.

**Thread-based Multitasking (Multithreading)**

• Threads share the same address space.

• Thread is lightweight.

• Cost of communication between the thread is low.

**What is Thread in java**

A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution. Threads are independent, if there occurs exception in one thread, it doesn't affect other threads. It shares a common memory area.

**Life Cycle of Thread**

A thread can be in one of the five states. According to sun, there is only 4 states in thread life cycle in java new, runnable, non-runnable and terminated.

There is no running state. But for better understanding the threads, we are explaining it in the 5 states. The life cycle of the thread in java is controlled by JVM.

The java thread states are as follows:

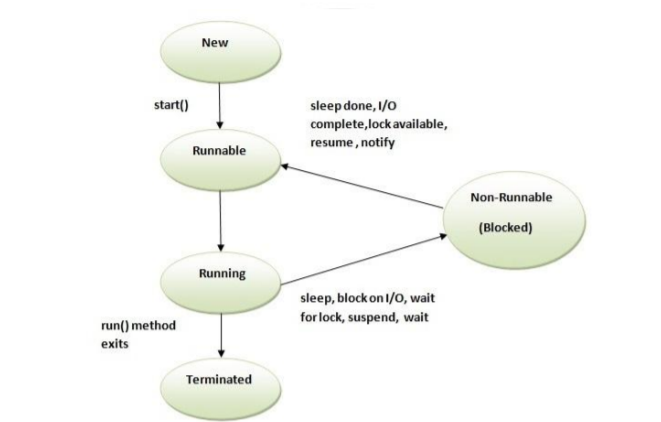
1. New

2. Runnable

3. Running

4. Non-Runnable (Blocked)

5. Terminated



**Java Thread pool** represents a group of worker threads that are waiting for the job and reuse many times.

In case of thread pool, a group of fixed size threads are created. A thread from the thread pool is pulled out and assigned a job by the service provider. After completion of the job, thread is contained in the thread pool again.

Advantage of Java Thread Pool

Better performance :It saves time because there is no need to create new thread.

Real time usage

It is used in Servlet and JSP where container creates a thread pool to process the request.

**Synchronous and Asynchronous Thread**

When you execute something synchronously, you wait for it to finish before moving on to another task. When you execute something asynchronously, you can move on to another task before it finishes.

That being, said, in the context of computers this translates into executing a process or task on another "thread." A thread is a series of commands--a block of code--that exists as a unit of work. The operating system can manage multiple threads and assign a thread a piece ("slice") of processor time before switching to another thread to give it a turn to do some work. At its core (pardon the pun), a processor can simply execute a command--it has no concept of doing two things at one time. The operating system simulates this by allocating slices of time to different threads.

Now, if you introduce multiple cores/processors into the mix, then things CAN actually happen at the same time. The operating system can allocate time to one thread on the first processor, then allocate the same block of time to another thread on a different processor.

All of this is about allowing the operating system to manage the completion of your task while you can go on in your code and do other things. Asynchronous programming is a complicated topic because of the semantics of how things tie together when you can do them at the same time.

**Thread Pool Executor:-**

Java thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed. We can use ThreadPoolExecutor to create thread pool in java.

**Features Of MySQL:**

**Ease of Management –**The software very easily gets downloaded and also uses an event scheduler to schedule the tasks automatically.

**Robust Transactional Support –**Holds the ACID (Atomicity, Consistency, Isolation, Durability) property, and also allows distributed multi-version support.

**Comprehensive Application Development –**MySQL has plugin libraries to embed the database into any application. It also supports stored procedures, triggers, functions, views and many more for application development.

**High Performance –**Provides fast load utilities with distinct memory caches and table index partitioning.

**Low Total Cost Of Ownership –**This reduces licensing costs and hardware expenditures.

**Open Source & 24 \* 7 Support –**This RDBMS can be used on any platform and offers

24\*7 support for open source and enterprise edition.

**Secure Data Protection –**MySQL supports powerful mechanisms to ensure that only authorized users have access to the databases.

**High Availability –**MySQL can run high-speed master/slave replication configurations and it offers cluster servers.

**Scalability & Flexibility –**With MySQL you can run deeply embedded applications and create data warehouses holding a humongous amount of data.

**Conclusion:**

In this way we are able to implement the JDBC and Multithreading Concept in java successfully in this assignment.

**Program code :**

**CLIENT SIDE:**

**1)** **BMS\_Client.java :-**

package com.company;  
  
import java.io.\*;  
import java.net.InetAddress;  
import java.net.Socket;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.Scanner;  
  
public class BMS\_Client {  
 Socket socket; //socket  
 ObjectOutputStream objectOutputStream; //objectOutputStream  
 OutputStream outputStream;  
 InputStream inputStream;  
 ObjectInputStream objectInputStream;  
 Scanner scanner=new Scanner(System.in);  
 DataInputStream dataInputStream;  
 DataOutputStream dataOutputStream;  
 InetAddress inetAddress;  
  
 BMS\_Client(){ //Constructor  
 try {  
 inetAddress=InetAddress.getLocalHost();  
 socket=new Socket(inetAddress,1401); //connecting to localhost  
 dataInputStream=new DataInputStream(socket.getInputStream());  
 dataOutputStream=new DataOutputStream(socket.getOutputStream());  
  
 outputStream=socket.getOutputStream();  
 inputStream=socket.getInputStream();  
  
 this.objectOutputStream = new ObjectOutputStream(outputStream);  
 this.objectInputStream = new ObjectInputStream(inputStream);  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 menu(); //Call Menu  
 }  
  
  
 public void menu(){  
 int ch;  
 do{  
 System.out.print("\n| WELCOME TO BR14x BANK SYSTEM |\n1)Login User \n2)User Registration\n3)Exit : ");  
 ch=scanner.nextInt();  
 switch (ch){  
 case 1: loginUser();  
 break;  
 case 2:registerUser();  
 break;  
 }  
 }while (ch<3);  
 }  
//  
 public void registerUser(){ //register New Account  
 Accounts account= new Accounts(); //create object  
 account.getAccountDetails(); //getData  
 try {  
 int send=1;  
 this.objectOutputStream.writeInt(send);  
 this.objectOutputStream.flush();  
  
 this.objectOutputStream.writeObject(account);  
 this.objectOutputStream.flush();  
  
 String msg=(String)this.objectInputStream.readObject(); //Print Message From Server  
 System.out.print(msg);  
 } catch (IOException | ClassNotFoundException 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  
 objectOutputStream.writeInt(2);  
 objectOutputStream.flush();  
 objectOutputStream.writeObject(accountNumber);  
 objectOutputStream.flush();  
 objectOutputStream.writeObject(accountPassword);  
 objectOutputStream.flush();  
  
 int type=objectInputStream.readInt();  
 if(type==1){ //if Account Exist in System  
 Accounts rAccount= (Accounts) objectInputStream.readObject();  
 System.out.print("\nWELCOME !\n");  
 int ch;  
 do{ //Menu  
 System.out.print("\n1)Show Account Details \n2)Withdraw Amount\n3)Deposit Amount\n4)Update Account Details\n5)Transfer Amount\n6)Show Account Statement\n7)Exit : ");  
 ch = scanner.nextInt();  
 switch (ch){  
 case 1:rAccount.showAccountDetails(); //Show Data  
 break;  
 case 2:withdrawAmount(rAccount); //Withdraw Amount  
 break;  
 case 3:depositAmount(rAccount); //Deposit Amount  
 break;  
 case 4:updateAccountInformation(rAccount); //Update Data  
 break;  
 case 5:transferAmount(rAccount); //Transfer Amount  
 break;  
 case 6:showStatement(rAccount); //Show Statement  
 break;  
 case 7:rAccount=null; //Logout/Exit  
 break;  
 }  
 }while(ch<7);  
 } else if(type==2) {  
 System.out.print("Invalid Credentials"); //If account doesnt Exist  
 }  
 }catch(EOFException e){  
 }  
 catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void withdrawAmount(Accounts loggedInAccount) {  
 double amount = loggedInAccount.getAccountBalance();  
 double withdrawAmount;  
 System.out.print("Enter The Amount You Withdraw : ");  
 withdrawAmount = scanner.nextDouble();  
 if (withdrawAmount > amount) {  
 System.out.println("Account Balance is low to withdraw amount " + withdrawAmount);  
 } else {  
 try {  
 objectOutputStream.writeInt(5);  
 objectOutputStream.flush();  
 Trans\_Req request=new Trans\_Req(loggedInAccount.accountNumber,loggedInAccount.accountNumber,withdrawAmount);  
 objectOutputStream.writeObject(request);  
 objectOutputStream.flush();  
 String msg=(String)objectInputStream.readObject();  
 if(msg.equals("Successful")){  
 amount -= withdrawAmount;  
 loggedInAccount.setAccountBalance(amount);  
 System.out.println("Amount Withdrawn \nAccount Balance : " + amount);  
 addTransactionStatement(loggedInAccount,withdrawAmount,amount,"Self","Self Withdraw");  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private void addTransactionStatement(Accounts loggedInAccount, double amount, double balance, String type, String description){  
 java.util.Date dateT=null;  
 SimpleDateFormat formattedDate = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss a");  
 String date = formattedDate.format(System.currentTimeMillis());  
  
 try {  
 dateT = formattedDate.parse(date);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 Transactions transactions=new Transactions(date,amount,balance,type,description);  
 loggedInAccount.transactionsArrayList.add(transactions);  
  
 }  
  
 private void depositAmount(Accounts loggedInAccount){  
 double amount = loggedInAccount.getAccountBalance();  
 double depositAmount;  
 System.out.print("Enter The Amount You Deposit : ");  
 depositAmount = scanner.nextDouble();  
 try {  
 objectOutputStream.writeInt(6);  
 objectOutputStream.flush();  
 Trans\_Req request=new Trans\_Req(loggedInAccount.accountNumber,loggedInAccount.accountNumber,depositAmount);  
 objectOutputStream.writeObject(request);  
 objectOutputStream.flush();  
 String msg=(String)objectInputStream.readObject();  
 if(msg.equals("Successful")){  
 amount += depositAmount;  
 loggedInAccount.setAccountBalance(amount);  
 System.out.println("Amount Deposited \nAccount Balance : " + amount);  
 addTransactionStatement(loggedInAccount,depositAmount,amount,"Self","Self Deposit");  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void showStatement(Accounts loggedInAccount){  
 for (int i = 0; i < 110; i++)  
 System.out.print('\_');  
 Transactions transactions;  
 System.out.println();  
 System.out.format("%1$-25s%2$-10s%3$-12s%4$-10s%5$-21s",  
 "Date", "Amount", "Balance", "Type", "Description");  
 System.out.println();  
 for (int i = 0; i < 110; i++)  
 System.out.print('\_');  
 System.out.println();  
 for(int i=0;i<loggedInAccount.transactionsArrayList.size();i++){  
 transactions=loggedInAccount.transactionsArrayList.get(i);  
 transactions.showStatement();  
 }  
 System.out.println();  
 for (int i = 0; i < 110; i++)  
 System.out.print('\_');  
 }  
  
  
 private void transferAmount(Accounts information){  
 Scanner scanner=new Scanner(System.in);  
 int transAccNo;  
 double amount;  
 String msg="@";  
 System.out.print("Enter The Account Number Of User Whom You Want To Transfer Amount :"); //Ask Acc Number  
 transAccNo= scanner.nextInt();  
  
 System.out.print("Enter Amount :"); //Ask amount  
 amount = scanner.nextDouble();  
  
 try {  
 objectOutputStream.writeInt(4);  
 objectOutputStream.flush();  
  
 Trans\_Req request=new Trans\_Req(information.accountNumber,transAccNo,amount); //Send Transfer Request  
 objectOutputStream.writeObject(request);  
 objectOutputStream.flush();  
 msg=(String)objectInputStream.readObject();  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 System.out.print(msg);  
 if(msg.equals("Transfer Successful")){  
 information.setAccountBalance((information.accountBalance-amount));  
 addTransactionStatement(information,amount,(information.accountBalance-amount),"Transfer","Transferred To "+transAccNo);  
 }  
 }  
  
 public void updateAccountInformation(Accounts 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("\nSelect Field You Want To Update/Change\n1)Name\n2)DOB\n3)PAN\n4)Address\n5)Nominee\n6)PhoneNumber\n7)Aadhar No\n8)SAVE & 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 :");  
 String date = scanner.nextLine();  
 System.out.print("Change DOB [1/0]? :");  
 ch2 = scanner.nextInt();  
 if (ch2 == 1) {  
  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");  
 Date userDOB;  
 try {  
 userDOB = dateFormat.parse(date);  
 information.setUserDOB(userDOB);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 } 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: 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++) {  
 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);  
 break;  
 } else {  
 System.out.print("Phone Number Unchanged");  
 }  
 stat=false;  
 break;  
 } else {  
 System.out.print("Phone Number Should Not Contain Any Letters");  
 System.out.println();  
 phoneNum = null;  
 stat = true;  
 }  
 }  
 }  
 } while (stat);  
 break;  
  
 case 7:  
 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++) {  
 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);  
 break;  
 } else {  
 System.out.print("Aadhar Number Unchanged");  
 }  
 stat = false;  
 break;  
 } 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  
 objectOutputStream.writeInt(3);  
 objectOutputStream.writeObject(information);  
 objectOutputStream.flush();  
 String message=(String)objectInputStream.readObject();  
 System.out.print(message);  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 System.out.println();  
 information.showAccountDetails();  
 process=false;  
 }  
 } while (process);  
 }  
  
 public static void main(String[] args){  
 new BMS\_Client(); //Initialize Object  
 }  
  
}

**2)Accounts.java:-**

package com.company;  
  
import java.io.Serializable;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.Scanner;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class Accounts implements Serializable {  
 private static final long serialVersionUID = 6128016096756071380L;  
 long accountNumber;  
 String userName,userPAN,userAddress,accountNominee,userPhoneNumber,userAadharNumber,accountPassword;  
 Date userDOB;  
 double accountBalance;  
 ArrayList<Transactions> transactionsArrayList;  
  
  
 public void getAccountDetails(){  
 Scanner scanner=new Scanner(System.in);  
 boolean stat = false;  
 System.out.println("Please Fill Out Following Details :\n");  
 System.out.print("Enter Your Full Name :");  
 userName=scanner.nextLine();  
  
 System.out.print("Enter Date Of Birth :");  
  
 String date = scanner.nextLine();  
  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");  
 userDOB=null;  
 try {  
 userDOB = dateFormat.parse(date);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 System.out.print("Enter Address :");  
 userAddress=scanner.nextLine();  
  
 System.out.print("Enter Nominee For Your Account :");  
 accountNominee = scanner.nextLine();  
  
 System.out.print("Enter Your PAN :");  
 userPAN = scanner.nextLine();  
  
 do {  
  
 System.out.print("Enter Your 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 {  
  
 System.out.print("Enter Your Aadhar Number :");  
 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 Number Should Not Contain Any Letters");  
 System.out.println();  
 userAadharNumber=null;  
 stat=true;  
 }  
 }  
 }  
  
  
 }while(stat);  
  
 do {  
 System.out.print("Enter Initial Balance (Must Be Above Rs.10000) : ");  
 accountBalance = scanner.nextDouble();  
 }while(accountBalance<10000);  
 scanner.nextLine();  
 boolean stat1;  
 String pass1;  
 do {  
 System.out.print("\nPlease Enter A New & 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("Password must contain Letters, minimum 1 Digit And 1 Special Characters");  
 pass1 = null;  
 stat1=true;  
 }  
  
 } else {  
 System.out.println("Invalid Password Should be of minimum size 8.");  
 stat1=true;  
 }  
 }while (stat1);  
  
 System.out.print("Please Enter Password Again To Confirm :");  
 String 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.");  
 }  
 }  
 transactionsArrayList=new ArrayList<>();  
 }  
  
 public void showAccountDetails(){  
 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",  
 getAccountNumber(), getUserName(), getUserPhoneNumber(), getUserAadharNumber(), getUserPAN(),  
 getUserDOB(),getUserAddress(), getAccountNominee(), getAccountBalance());  
 System.out.println();  
 for (int i = 0; i < 155; i++)  
 System.out.print('\_');  
 }  
  
 public long getAccountNumber() {  
 return accountNumber;  
 }  
  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public String getUserPAN() {  
 return userPAN;  
 }  
  
 public void setUserPAN(String userPAN) {  
 this.userPAN = userPAN;  
 }  
  
 public String getUserAddress() {  
 return userAddress;  
 }  
  
 public void setUserAddress(String userAddress) {  
 this.userAddress = userAddress;  
 }  
  
 public String getAccountNominee() {  
 return accountNominee;  
 }  
  
 public void setAccountNominee(String accountNominee) {  
 this.accountNominee = accountNominee;  
 }  
  
 public String getUserPhoneNumber() {  
 return userPhoneNumber;  
 }  
  
 public void setUserPhoneNumber(String userPhoneNumber) {  
 this.userPhoneNumber = userPhoneNumber;  
 }  
  
 public String getUserAadharNumber() {  
 return userAadharNumber;  
 }  
  
 public void setUserAadharNumber(String userAadharNumber) {  
 this.userAadharNumber = userAadharNumber;  
 }  
  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 public Date getUserDOB() {  
 return userDOB;  
 }  
  
 public void setUserDOB(Date userDOB) {  
 this.userDOB = userDOB;  
 }  
}

**3)** **Trans\_Req.java:-**

package com.company;  
  
import java.io.Serializable;  
  
public class Trans\_Req implements Serializable { //Class to Transfer Amount  
 private static final long serialVersionUID = 6128016096756071380L;  
 long accountNumberSender,accountNumberReceiver;  
 double amount;  
  
 public Trans\_Req(long accountNumberSender, long accountNumberReceiver, double senderAmount) { //Constructor  
 this.accountNumberSender = accountNumberSender;  
 this.accountNumberReceiver = accountNumberReceiver;  
 this.amount = senderAmount;  
 }  
}

**4) Transactions.java:**

package com.company;  
  
import java.io.Serializable;  
import java.util.Date;  
  
public class Transactions implements Serializable {  
 private static final long serialVersionUID = 6128016096756071380L;  
  
 private String transactionDateAndTime;  
 private double transactionAmount;  
 private double accountBalance;  
 private String transactionType;  
 private String transactionDescription;  
  
 public Transactions(String transactionDateAndTime, double transactionAmount, double accountBalance, String transactionType, String transactionDescription) {  
 this.transactionDateAndTime = transactionDateAndTime;  
 this.transactionAmount = transactionAmount;  
 this.accountBalance = accountBalance;  
 this.transactionType = transactionType;  
 this.transactionDescription = transactionDescription;  
 }  
  
 public void showStatement(){  
 System.out.format("%1$-25s%2$-10s%3$-12s%4$-10s%5$-21s",  
 getTransactionDateAndTime(),  
 getTransactionAmount(),  
 getAccountBalance(),  
 getTransactionType(),  
 getTransactionDescription()  
 );  
 System.out.println();  
 }  
  
 public String getTransactionDateAndTime() {  
 return transactionDateAndTime;  
 }  
  
 public void setTransactionDateAndTime(String transactionDateAndTime) {  
 this.transactionDateAndTime = transactionDateAndTime;  
 }  
  
 public double getTransactionAmount() {  
 return transactionAmount;  
 }  
  
 public void setTransactionAmount(double transactionAmount) {  
 this.transactionAmount = transactionAmount;  
 }  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 public String getTransactionType() {  
 return transactionType;  
 }  
  
 public void setTransactionType(String transactionType) {  
 this.transactionType = transactionType;  
 }  
  
 public String getTransactionDescription() {  
 return transactionDescription;  
 }  
  
 public void setTransactionDescription(String transactionDescription) {  
 this.transactionDescription = transactionDescription;  
 }  
}

**SERVER SIDE:**

**1)** Server.java:-

package com.company;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.sql.\*;  
  
public class Server {  
  
 ServerSocket serverSocket;  
 Socket socket;  
 ObjectInputStream objectInputStream;  
 ObjectOutputStream objectOutputStream;  
 Connection connect;  
 Statement statement;  
 PreparedStatement preparedStatement;  
 ResultSet resultSet;  
 static int clientNo=0;  
  
 Server(){  
 try {  
 serverSocket=new ServerSocket(1401);  
 try {  
 Class.forName("com.mysql.cj.jdbc.Driver");  
 connect = DriverManager.getConnection("jdbc:mysql://localhost:3307/bms", "root", "");  
 statement = connect.createStatement();  
 } catch (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 while (true){  
 try {  
 assert serverSocket != null;  
 socket=serverSocket.accept();  
  
 objectOutputStream = new ObjectOutputStream(socket.getOutputStream());  
 objectOutputStream.flush();  
 objectInputStream = new ObjectInputStream(socket.getInputStream());  
  
 Thread thread=new BMS(socket,objectOutputStream,objectInputStream,connect,statement,preparedStatement,resultSet);  
 thread.start();  
  
 String name="Client "+getClientNo();  
 thread.setName(name);  
  
 if(thread.isAlive()){  
 System.out.print("\n"+name+" Is Connected");  
 }  
 if(socket.isClosed()){  
 removeClient();  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 public int getClientNo(){  
 return ++clientNo;  
 }  
 public void removeClient(){  
 --clientNo;  
 }  
 public static void main(String[] args){  
 new Server();  
 }  
}

**2)BMS.java:-**

package com.company;  
  
import java.io.\*;  
import java.net.Socket;  
import java.net.SocketException;  
import java.sql.\*;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
  
public class BMS extends Thread{  
 Socket socket ;  
 ObjectInputStream objectInputStream;  
 ObjectOutputStream objectOutputStream;  
 Connection connect;  
 Statement statement;  
 PreparedStatement preparedStatement;  
 ResultSet resultSet;  
  
 public BMS(Socket socket, ObjectOutputStream objectOutputStream, ObjectInputStream objectInputStream, Connection connect, Statement statement, PreparedStatement preparedStatement, ResultSet resultSet) {  
 this.socket = socket;  
 this.objectInputStream = objectInputStream;  
 this.objectOutputStream = objectOutputStream;  
 this.connect=connect;  
 this.statement=statement;  
 this.preparedStatement=preparedStatement;  
 this.resultSet=resultSet;  
 }  
 private void receive() throws IOException, ClassNotFoundException, SQLException {  
 int ch=this.objectInputStream.readInt();  
 switch (ch){  
 case 1: createAccount();  
 break;  
 case 2: authenticateAccount();  
 break;  
 case 3: updateAccountInfo();  
 break;  
 case 4: transferAmount();  
 break;  
 case 5: withdrawAmount();  
 break;  
 case 6: depositAmount();  
 break;  
 }  
 }  
  
 private void updateAccountInfo() throws IOException, ClassNotFoundException {  
 Accounts information=(Accounts)objectInputStream.readObject();  
  
 String query ="UPDATE customers SET userName = '"+information.getUserName()+  
 "' , userDOB = '"+information.getUserDOB()+  
 "' , userPAN = '"+information.getUserPAN()+  
 "' , userAddress ='"+information.getUserAddress()+  
 "' , accountNominee ='"+information.getAccountNominee()+  
 "' , userPhoneNumber ='"+information.getUserPhoneNumber()+  
 "' , userAadharNumber ='"+information.getUserAadharNumber()+  
 "' , accountBalance = "+information.getAccountBalance()+ " WHERE accountNumber ="+information.getAccountNumber()+";";  
 try {  
 int stat=this.statement.executeUpdate(query);  
 if(stat==1) {  
 objectOutputStream.writeObject("Account Details Updated !");  
 }else{  
 objectOutputStream.writeObject("Account Details Not Updated !");  
 }  
 objectOutputStream.flush();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 private void createAccount() throws IOException, ClassNotFoundException { //Create Account  
 Accounts newAccount=(Accounts) this.objectInputStream.readObject(); //Get Object from Client  
 long newAccountNumber=0;  
 String query0="SELECT \* FROM customers WHERE accountNumber=(SELECT MAX(accountNumber) FROM customers)";  
 try {  
 this.resultSet=this.statement.executeQuery(query0);  
 if (resultSet.next()){  
 newAccountNumber=resultSet.getInt("accountNumber");  
 newAccountNumber+=1;  
 }else{  
 newAccountNumber=100;  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 newAccount.accountNumber=newAccountNumber;  
 String query = "INSERT INTO customers (accountNumber, userName, userDOB, userPAN, userAddress, accountNominee, userPhoneNumber, userAadharNumber, accountBalance, accountPassword)"  
 +"VALUES (?,?,?,?,?,?,?,?,?,?)";  
 try {  
 this.preparedStatement = connect.prepareStatement(query,Statement.RETURN\_GENERATED\_KEYS);  
 this.preparedStatement.setLong(1, newAccountNumber);  
 this.preparedStatement.setString(2, newAccount.getUserName());  
 this.preparedStatement.setDate(3, convertUtilToSql(newAccount.userDOB));  
 this.preparedStatement.setString(4, newAccount.getUserPAN());  
 this.preparedStatement.setString(5, newAccount.getUserAddress());  
 this.preparedStatement.setString(6, newAccount.getAccountNominee());  
 this.preparedStatement.setString(7, newAccount.getUserPhoneNumber());  
 this.preparedStatement.setString(8, newAccount.getUserAadharNumber());  
 this.preparedStatement.setDouble(9, newAccount.getAccountBalance());  
 this.preparedStatement.setString(10, newAccount.getAccountPassword());  
  
 int rowAffected = this.preparedStatement.executeUpdate();  
 if(rowAffected > 0) {  
 System.out.print("\nAccount Created Successfully\n");  
 this.objectOutputStream.writeObject("Account Generated With Account number : " +"\033[0;31m"+newAccountNumber+"\033[0m");  
  
 String query1="INSERT INTO logincredentials(accountNumber,password,userId) VALUES ("+newAccountNumber+" , '"  
 +newAccount.getAccountPassword()+"' , (SELECT userId from customers WHERE accountNumber ="+newAccountNumber +" ))";  
 int result=statement.executeUpdate(query1);  
 if(result>0){  
 this.objectOutputStream.writeObject("\nLogin To Proceed !");  
 this.objectOutputStream.flush();  
 }  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 newAccount.showData();  
 }  
  
 private void authenticateAccount() throws IOException, ClassNotFoundException { //Authenticate Client  
  
 Accounts account;  
 String accNum=(String)objectInputStream.readObject(); //Receive AccountNumber  
 String pass=(String)objectInputStream.readObject(); // Receive Password  
 int flag = 0;  
 long accNo=Long.parseLong(accNum); //Check if account Exist  
  
 String query1="SELECT \* FROM logincredentials WHERE accountNumber = "+accNo;  
 try {  
 resultSet = statement.executeQuery(query1);  
 while (resultSet.next()) {  
 String password = resultSet.getString("password");  
 if (pass.equals(password)) {  
 account=getAccountData(accNo);  
 this.objectOutputStream.writeInt(1);  
 this.objectOutputStream.flush();  
 this.objectOutputStream.writeObject(account);  
 this.objectOutputStream.flush();  
 flag=2;  
 }else{  
 flag=-1;  
 }  
 }  
 if(flag==-1) {  
 this.objectOutputStream.writeInt(2);  
 this.objectOutputStream.flush();  
 }  
 }catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 private Accounts getAccountData(long accNo){  
 Accounts accountData=new Accounts();  
 accountData.transactionsArrayList=new ArrayList<>();  
 try {  
 String query = "SELECT \* FROM customers WHERE accountNumber = " + accNo;  
 resultSet = statement.executeQuery(query);  
 while (resultSet.next()) {  
 accountData.setUserName(resultSet.getString("userName"));  
 accountData.setUserDOB(convertSqlToUtil(resultSet.getDate("userDOB")));  
 accountData.setUserPAN(resultSet.getString("userPAN"));  
 accountData.setUserAddress(resultSet.getString("userAddress"));  
 accountData.setAccountNominee(resultSet.getString("accountNominee"));  
 accountData.setUserPhoneNumber(resultSet.getString("userPhoneNumber"));  
 accountData.setUserAadharNumber(resultSet.getString("userAadharNumber"));  
 accountData.setAccountBalance(resultSet.getDouble("accountBalance"));  
 accountData.setAccountNumber(accNo);  
 }  
 String query1="SELECT \* FROM transactionstatements WHERE accountNumber ="+accNo;  
 resultSet=statement.executeQuery(query1);  
 while (resultSet.next()) {  
 Transactions transactions=new Transactions();  
 transactions.setTransactionDateAndTime(resultSet.getString("dateAndTime"));  
 transactions.setTransactionAmount(resultSet.getDouble("amount"));  
 transactions.setAccountBalance(resultSet.getDouble("accountBalance"));  
 transactions.setTransactionType(resultSet.getString("type"));  
 transactions.setTransactionDescription(resultSet.getString("description"));  
 accountData.transactionsArrayList.add(transactions);  
 }  
 return accountData;  
 }catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return accountData;  
 }  
 private void withdrawAmount() throws IOException, ClassNotFoundException, SQLException {  
 Trans\_Req request=(Trans\_Req)objectInputStream.readObject();  
 long fromAcc=request.getAccountNumberSender();  
 double amount=request.getAmount();  
 String query ="UPDATE customers SET accountBalance = (SELECT accountBalance FROM customers where accountNumber ="+fromAcc+") - "+amount+" WHERE accountNumber ="+fromAcc;  
 int result=statement.executeUpdate(query);  
 if(result>0){  
 updateTransaction(fromAcc,fromAcc,amount,"Self","Self Withdraw");  
 objectOutputStream.writeObject("Successful");  
 }else{  
 objectOutputStream.writeObject("Unsuccessful");  
 }  
  
 }  
 private void depositAmount() throws IOException, ClassNotFoundException, SQLException {  
 Trans\_Req request=(Trans\_Req)objectInputStream.readObject();  
 long fromAcc=request.getAccountNumberSender();  
 double amount=request.getAmount();  
 String query ="UPDATE customers SET accountBalance = (SELECT accountBalance FROM customers where accountNumber ="+fromAcc+") + "+amount+" WHERE accountNumber ="+fromAcc;  
 int result=statement.executeUpdate(query);  
 if(result>0){  
 updateTransaction(fromAcc,fromAcc,amount,"Self","Self Deposit");  
 objectOutputStream.writeObject("Successful");  
 }else{  
 objectOutputStream.writeObject("Unsuccessful");  
 }  
 }  
 public void transferAmount() throws IOException, ClassNotFoundException { //transfer Account  
 Trans\_Req amountT=(Trans\_Req)objectInputStream.readObject(); //get Transaction Information  
 long transAccNo=amountT.accountNumberSender;  
 long receiverAccNo=amountT.accountNumberReceiver;  
 double transferAmount=amountT.amount;  
  
 String query=" SELECT \* FROM customers WHERE accountNumber = "+transAccNo;  
 try {  
 resultSet=statement.executeQuery(query);  
 if(resultSet.next()){  
 double balance=resultSet.getDouble("accountBalance");  
 if ( balance > transferAmount) {  
 String query1="UPDATE customers SET accountBalance = (SELECT accountBalance FROM customers where accountNumber ="+transAccNo+") - "+transferAmount+" WHERE accountNumber ="+transAccNo;  
 statement.executeUpdate(query1);  
 String query2="UPDATE customers SET accountBalance = (SELECT accountBalance FROM customers where accountNumber ="+receiverAccNo+") + "+transferAmount+" WHERE accountNumber ="+receiverAccNo;  
 statement.executeUpdate(query2);  
 updateTransaction(transAccNo,receiverAccNo,transferAmount,"Transfer","Transferred To "+receiverAccNo);  
 updateTransaction(receiverAccNo,transAccNo,transferAmount,"Transfer","Transferred From "+transAccNo);  
 objectOutputStream.writeObject("Transfer Successful");  
 }  
 }else{  
 objectOutputStream.writeObject("Transfer Unsuccessful");  
 }  
 }catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void updateTransaction(long senderAccNo,long receiverAccNo,double amount,String type,String description) throws SQLException {  
  
 double accBalance=0;  
 SimpleDateFormat formattedDate = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss a");  
 String date = formattedDate.format(System.currentTimeMillis());  
  
 String query0 ="SELECT accountBalance FROM customers where accountNumber ="+senderAccNo;  
 resultSet = statement.executeQuery(query0);  
 if(resultSet.next()){  
 accBalance=resultSet.getDouble("accountBalance");  
 }  
  
 String query1 = "INSERT INTO transactionstatements (accountNumber, toAccountNumber, dateAndTime,accountBalance,type,amount,description)"  
 +"VALUES (?,?,?,?,?,?,?)";  
 try {  
 this.preparedStatement = this.connect.prepareStatement(query1,Statement.RETURN\_GENERATED\_KEYS);  
 this.preparedStatement.setLong(1, senderAccNo);  
 this.preparedStatement.setLong(2, receiverAccNo);  
 this. preparedStatement.setString(3, date);  
 this.preparedStatement.setDouble(4,accBalance);  
 this.preparedStatement.setString(5, type);  
 this.preparedStatement.setDouble(6, amount);  
 this.preparedStatement.setString(7, description);  
 this.preparedStatement.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static java.sql.Date convertUtilToSql(java.util.Date uDate) {  
 java.sql.Date sDate = new java.sql.Date(uDate.getTime());  
 return sDate;  
 }  
  
 private static java.util.Date convertSqlToUtil(java.sql.Date sDate) {  
 java.util.Date uDate = new java.sql.Date(sDate.getTime());  
 return uDate;  
 }  
  
 @Override  
 public void run() {  
 while (true){  
 try {  
 receive();  
 }catch(EOFException | SocketException ignored){  
 } catch (IOException | ClassNotFoundException | SQLException e ) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

**3)Accounts.java:-**

package com.company;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.Date;  
  
public class Accounts implements Serializable { //AccountInformation Class  
 private static final long serialVersionUID = 6128016096756071380L;  
 //Data Fields  
 long accountNumber;  
 String userName,userPAN,userAddress,accountNominee,userPhoneNumber,userAadharNumber,accountPassword;  
 Date userDOB;  
 double accountBalance;  
 ArrayList<Transactions> transactionsArrayList;  
  
 //getter steer Functions  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 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$-12s%6$-22s%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$-12s%6$-22s%7$-30s%8$-20s%9$-17s", //Print Data  
 accountNumber, userName, userPhoneNumber, userAadharNumber,  
 userPAN, userDOB, userAddress, accountNominee, accountBalance);  
 System.out.println();  
 for (int i = 0; i < 155; i++)  
 System.out.print('\_');  
 }  
  
 public long getAccountNumber() {  
 return accountNumber;  
 }  
 public String getUserAddress() {  
 return userAddress;  
 }  
 public String getAccountNominee() {  
 return accountNominee;  
 }  
 public String getUserPhoneNumber() {  
 return userPhoneNumber;  
 }  
 public String getUserAadharNumber() {  
 return userAadharNumber;  
 }  
 public String getAccountPassword() {  
 return accountPassword;  
 }  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
 public Date getUserDOB() {  
 return userDOB;  
 }  
 public String getUserPAN() {  
 return userPAN;  
 }  
 public void setAccountNumber(long accountNumber) {  
 this.accountNumber = accountNumber;  
 }  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
 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 setUserDOB(Date userDOB) {  
 this.userDOB = userDOB;  
 }  
  
}

**4)** **Trans\_Req.java:-**

package com.company;  
  
import java.io.Serializable;  
  
public class Trans\_Req implements Serializable { //Transfer Amount Class  
 private static final long serialVersionUID = 6128016096756071380L;  
 long accountNumberSender,accountNumberReceiver; // fields  
 double amount;  
  
 public Trans\_Req(long accountNumberSender, long accountNumberReceiver, double amount) { //Constructor  
 this.accountNumberSender = accountNumberSender;  
 this.accountNumberReceiver = accountNumberReceiver;  
 this.amount = amount;  
 }  
 public long getAccountNumberSender() {  
 return accountNumberSender;  
 }  
 public double getAmount() {  
 return amount;  
 }  
}

5) **Transactions.java:-**

package com.company;  
  
import java.io.Serializable;  
  
public class Transactions implements Serializable {  
 private static final long serialVersionUID = 6128016096756071380L;  
 private String transactionDateAndTime;  
 private double transactionAmount;  
 private double accountBalance;  
 private String transactionType;  
 private String transactionDescription;  
  
 public Transactions() {  
 }  
 public void setTransactionDateAndTime(String transactionDateAndTime) {  
 this.transactionDateAndTime = transactionDateAndTime;  
 }  
 public void setTransactionAmount(double transactionAmount) {  
 this.transactionAmount = transactionAmount;  
 }  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
 public void setTransactionType(String transactionType) {  
 this.transactionType = transactionType;  
 }  
 public void setTransactionDescription(String transactionDescription) {  
 this.transactionDescription = transactionDescription;  
 }  
}

**Output:**

**CLIENT SIDE:**

| WELCOME TO BR14x BANK SYSTEM |

1)Login User

2)User Registration

3)Exit : 1

Enter Account Number : 100

Enter Account Password : Bh@vik14

WELCOME !

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 1

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

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

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

100 Bhavik Ransubhe 9090909090 123456654321 EMP1401BR 2000-01-14 Pune,India Girish Ransubhe 16500.0

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

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 7

| WELCOME TO BR14x BANK SYSTEM |

1)Login User

2)User Registration

3)Exit : 2

Please Fill Out Following Details :

Enter Your Full Name :Kylie Jenner

Enter Date Of Birth :10-08-1997

Enter Address :LA,USA

Enter Nominee For Your Account :Travis

Enter Your PAN :EMP1997KJ

Enter Your Phone Number :20202020200

Phone Number Is Not Valid

Enter Your Phone Number :2020202020

Enter Your Aadhar Number :123456789789

Enter Initial Balance (Must Be Above Rs.10000) : 70000

Please Enter A New & Unique Password : Kylie@10

Please Enter Password Again To Confirm :Kylie@10

Account Generated With Account number : 102

| WELCOME TO BR14x BANK SYSTEM |

1)Login User

2)User Registration

3)Exit : 1

Enter Account Number : 102

Enter Account Password : Kylie@10

WELCOME !

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 1

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

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

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

102 Kylie Jenner 2020202020 123456789789 EMP1997KJ 1997-08-10 LA,USA Travis 70000.0

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

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 2

Enter The Amount You Withdraw : 5000

Amount Withdrawn

Account Balance : 65000.0

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 3

Enter The Amount You Deposit : 2000

Amount Deposited

Account Balance : 67000.0

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 4

Select Field You Want To Update/Change

1)Name

2)DOB

3)PAN

4)Address

5)Nominee

6)PhoneNumber

7)Aadhar No

8)SAVE & EXIT : 5

Current Nominee :Travis

Enter Change Nominee :Travis Scott

Change Nominee [1/0]? :1

Select Field You Want To Update/Change

1)Name

2)DOB

3)PAN

4)Address

5)Nominee

6)PhoneNumber

7)Aadhar No

8)SAVE & EXIT : 8

Account Details Updated !

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

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

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

102 Kylie Jenner 2020202020 123456789789 EMP1997KJ 1997-08-10 LA,USA Travis Scott 67000.0

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

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 5

Enter The Account Number Of User Whom You Want To Transfer Amount :100

Enter Amount :5000

Transfer Successful

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 6

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

Date Amount Balance Type Description

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

25/11/2020 11:07:47 am 5000.0 65000.0 Self Self Withdraw

25/11/2020 11:07:55 am 2000.0 67000.0 Self Self Deposit

25/11/2020 11:08:51 am 5000.0 57000.0 Transfer Transferred To 100

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

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 1

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

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

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

102 Kylie Jenner 2020202020 123456789789 EMP1997KJ 1997-08-10 LA,USA Travis Scott 62000.0

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

1)Show Account Details

2)Withdraw Amount

3)Deposit Amount

4)Update Account Details

5)Transfer Amount

6)Show Account Statement

7)Exit : 7

| WELCOME TO BR14x BANK SYSTEM |

1)Login User

2)User Registration

3)Exit : 3

Process finished with exit code 0

**SERVER SIDE:**

Client 1 Is Connected

Account Created Successfully

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

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

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

102 Kylie Jenner 2020202020 123456789789 EMP1997KJ Sun Aug 10 00:00:00 IST 1997LA,USA Travis 70000.0

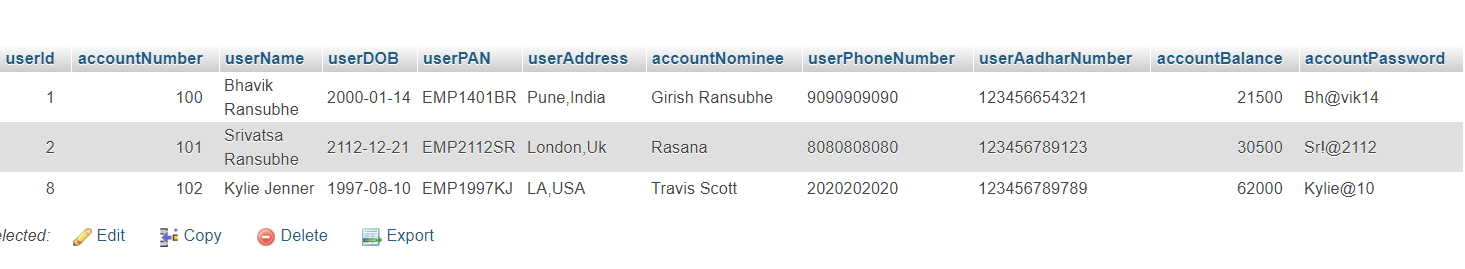
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Client 2 Is Connected

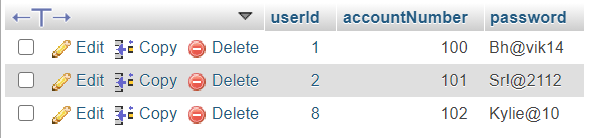
Process finished with exit code 0

**PHPMYADMIN TABLES:**

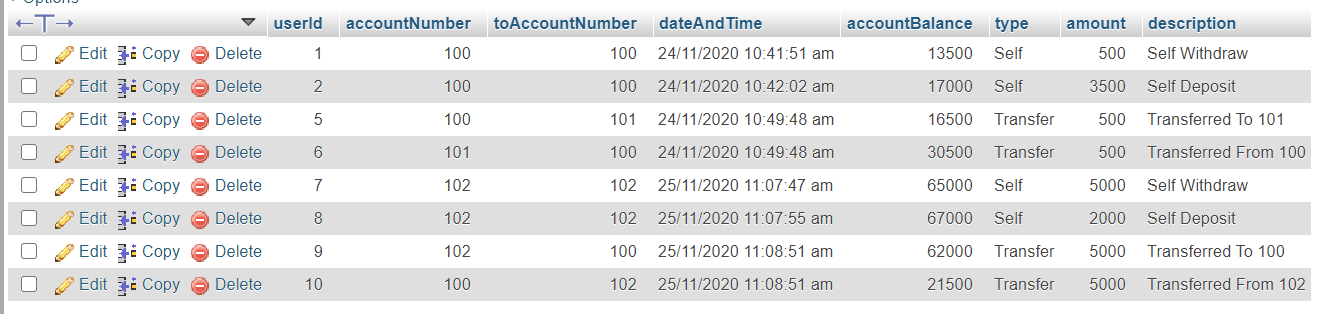
**1)**

****

**2)**

****

**3)**

****

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