**NAME**: Bhavik Ransubhe

**CLASS** : TE (B) COMP

**ROLL NO** : 39055

**Title :**

Transform the bank system from command line system to GUI based application

**Objective:**

Understand the implementation of the GUI based Application using Swing , AWT Concept.

**Problem Statement:**

Write a Java program with the help of GUI based Application Perform for bank system.

**Outcomes:**

After completion of this assignment we will be able to implement the concept of Swing and AWT in GUI based Application .

**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 AWT** (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

**Java AWT Hierarchy**

The hierarchy of Java AWT classes are given below.

**Container**

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

**Window**

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

**Panel**

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

**Frame**

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

Useful Methods of Component class

**Method** **Description**

public void add(Component c) - inserts a component on this component.

public void setSize(int width,int height) - sets the size (width and height) of the component.

public void setLayout(LayoutManager m) - defines the layout manager for the component.

public void setVisible(boolean status) - changes the visibility of the component, by default false.

To create simple awt example, you need a frame. There are two ways to create a frame in AWT.

• By extending Frame class (inheritance)

• By creating the object of Frame class (association)

**Swing**

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

**Difference between AWT and Swing**

There are many differences between java awt and swing that are given below.

**No. Java AWT Java swing com**

1) AWT components are platform-dependent. Java swing components are platform independent.

2) AWT components are heavyweight. Swing components are lightweight.

3) AWT doesn't support pluggable look and feel. Swing supports pluggable look and feel

4) AWT provides less components than Swing. Swing provides more powerful components

such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.

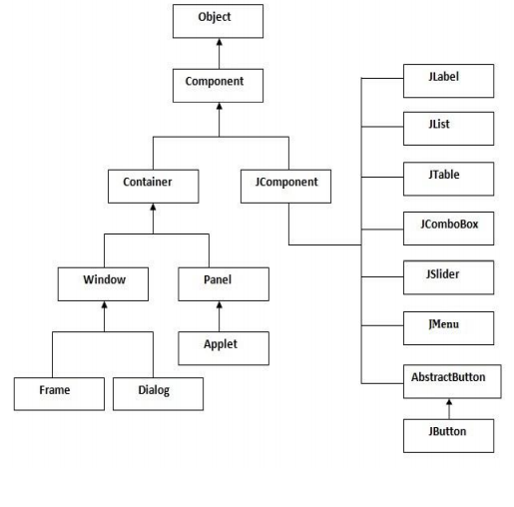
5) AWT doesn't follows MVC(Model View Controller) Swing follows MVC.

where model represents data, view represents

presentation and controller acts as an interface

between model and view.

The hierarchy of java swing API is given below:



Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below

**Method** **Description**

public void add(Component c) - add a component on another component

public void setSize(int width,int height) - sets size of the component.

public void setLayout(LayoutManager m) - sets the layout manager for the component.

public void setVisible(boolean b) - sets the visibility of the component. It is by default false

Java Swing Examples

There are two ways to create a frame:

• By creating the object of Frame class (association)

• By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

**Java ActionListener Interface**

The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods. The Java ItemListener is notified whenever you click on the checkbox. It is notified against ItemEvent. The ItemListener interface is found in java.awt.event package. It has only one method: itemStateChanged().

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package. It has three methods.

The Java WindowListener is notified whenever you change the state of window. It is notified against WindowEvent. The WindowListener interface is found in java.awt.event package. It has three methods.

Java adapter classes provide the default implementation of listener interfaces. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it saves code.

The adapter classes are found in java.awt.event, java.awt.dnd and javax.swing.event packages. The Adapter classes with their corresponding listener interfaces are given below.

**java.awt.event Adapter classes**

**Adapter class** **Listener interface**

WindowAdapter - WindowListener

KeyAdapter - KeyListener

MouseAdapter - MouseListener

MouseMotionAdapter - MouseMotionListener

FocusAdapter - FocusListener

ComponentAdapter - ComponentListener

ContainerAdapter - ContainerListener

HierarchyBoundsAdapter - HierarchyBoundsListener

**CODE:**

**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;  
 }  
}

**CLIENT SIDE:**

**1)LoginPage.java:-**

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.\*;  
import java.net.InetAddress;  
import java.net.Socket;  
  
public class LoginPage extends JFrame {  
 static Socket socket; //socket  
 static ObjectOutputStream objectOutputStream; //objectOutputStream  
 static OutputStream outputStream;  
 static InputStream inputStream;  
 static ObjectInputStream objectInputStream;  
 static DataInputStream dataInputStream;  
 static DataOutputStream dataOutputStream;  
 static InetAddress inetAddress;  
  
 //creating instance of JSwing Components  
 JLabel bankInfo=new JLabel("WELCOME TO BR14X BANK MANAGEMENT SYSTEM"); //Creates a JLabel instance with no image and with an empty string for the title  
 JLabel accNoLabel=new JLabel("Type Account Number");  
 JTextField accNoInput=new JTextField();  
 JLabel passLabel=new JLabel("Type Your Password");  
 JPasswordField passwordInput=new JPasswordField();  
 JButton loginButton=new JButton("LOGIN");  
 JTextArea area=new JTextArea("--------------------------------OR--------------------------------");  
 JButton openAccountButton=new JButton("CREATE NEW ACCOUNT");  
  
 JLabel errorLabel=new JLabel("");  
  
 Container container=getContentPane(); //provides a space where a component can be located  
  
 LoginPage(){  
  
 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();  
  
 objectOutputStream = new ObjectOutputStream(outputStream);  
 objectInputStream = new ObjectInputStream(inputStream);  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 startGUI();  
 }  
  
 private void startGUI(){  
  
 container.setLayout(null);  
 //x axis, y axis, width, height  
 bankInfo.setBounds(180,20,340,30);  
 accNoLabel.setBounds(20,90,250,25);  
 passLabel.setBounds(340,90,150,25);  
 accNoInput.setBounds(150,90,150,25);  
 passwordInput.setBounds(470,90,150,25);  
  
 loginButton.setBounds(220,150,150,35);  
  
  
 area.setBounds(175,190,270,25);  
 openAccountButton.setBounds(220,230,190,35);  
  
 errorLabel.setBounds(90,305,200,25);  
 errorLabel.setVisible(false);  
  
 container.add(bankInfo);  
 container.add(accNoLabel);  
 container.add(accNoInput);  
 container.add(passLabel);  
 container.add(passwordInput);  
 container.add(loginButton);  
 container.add(area);  
 container.add(openAccountButton);  
 container.add(errorLabel);  
  
 openAccountButton.addActionListener(e -> {  
 setVisible(false);  
 new OpenNewAccountPage();  
 });  
  
 loginButton.addActionListener(e -> getLoginCredentials());  
  
 setVisible(true); //making the frame visible  
 setSize(680,320); //setting width and height  
 setTitle("BR14x BANK SYSTEM - LOGIN");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 }  
  
 private void getLoginCredentials(){  
  
 String accountNumber= accNoInput.getText();  
 String accountPassword=passwordInput.getText();  
  
 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();  
 new LoggedInMenuPage(rAccount);  
 setVisible(false);  
 } else if(type==2) {  
 sendErrorMessage();  
 }  
 }catch(EOFException ignored){  
 }  
 catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 void sendErrorMessage(){  
 JOptionPane.showMessageDialog(null, "INVALID INPUTS | TRY AGAIN", "Error",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
  
 public static void main(String[] args) { new LoginPage();  
 }  
}

2)Accounts.java:-

package com.company;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.Date;  
  
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 Accounts(long accountNumber, String userName, String userPAN, String userAddress, String accountNominee, String userPhoneNumber, String userAadharNumber, String accountPassword, Date userDOB, double accountBalance, ArrayList<Transactions> transactionsArrayList) {  
 this.accountNumber = accountNumber;  
 this.userName = userName;  
 this.userPAN = userPAN;  
 this.userAddress = userAddress;  
 this.accountNominee = accountNominee;  
 this.userPhoneNumber = userPhoneNumber;  
 this.userAadharNumber = userAadharNumber;  
 this.accountPassword = accountPassword;  
 this.userDOB = userDOB;  
 this.accountBalance = accountBalance;  
 this.transactionsArrayList = transactionsArrayList;  
 }  
  
 public long getAccountNumber() {  
 return accountNumber;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 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;  
 }  
  
 public String getAccountPassword() {  
 return accountPassword;  
 }  
  
 public Date getUserDOB() {  
 return userDOB;  
 }  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public void setAccountBalance(double accountBalance) {  
 this.accountBalance = accountBalance;  
 }  
  
 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;  
 }  
}

3) LoggedInMenuPage.java:-

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class LoggedInMenuPage extends JFrame {  
 //creating instance of JSwing Components  
 JLabel bankInfo=new JLabel("BR14X BANK SYSTEM - LOGGED IN PAGE ");  
  
 JLabel accountNumberLabel=new JLabel("ACCOUNT NUMBER :");  
 JLabel accountNumberValue=new JLabel();  
  
 JLabel menu=new JLabel("DO YOU WANT TO :");  
 JLabel details=new JLabel("ACCOUNT DETAILS");  
  
 JLabel userNameLabel=new JLabel("User Name :");  
 JLabel userNameValue=new JLabel();  
  
 JLabel userPhoneNumberLabel=new JLabel("Phone Number :");  
 JLabel userPhoneNumberValue=new JLabel();  
  
 JLabel userAadharLabel=new JLabel("Aadhar Number :");  
 JLabel userAadharValue=new JLabel();  
  
 JLabel userPANLabel=new JLabel("PAN Number :");  
 JLabel userPANValue=new JLabel();  
  
 JLabel userDOBLabel=new JLabel("DOB :");  
 JLabel userDOBValue=new JLabel();  
  
 JLabel userAddressLabel=new JLabel("Address :");  
 JLabel userAddressValue=new JLabel();  
  
 JLabel accountNomineeLabel=new JLabel("Account Nominee :");  
 JLabel accountNomineeValue=new JLabel();  
  
 JLabel accountBalanceLabel=new JLabel("Account Balance :");  
 JLabel accountBalanceValue=new JLabel();  
  
 JRadioButton withdrawAmount=new JRadioButton("Withdraw Amount");  
 JRadioButton depositAmount=new JRadioButton("Deposit Amount");  
 JRadioButton transferAmount=new JRadioButton("Transfer Amount");  
 JRadioButton updateAccountDetails=new JRadioButton("Update Account Details");  
 JRadioButton showStatement=new JRadioButton("Show Statement");  
 Container container=getContentPane();  
  
 JButton proceed=new JButton("Proceed");  
 JButton logOut=new JButton("LOGOUT");  
  
 Accounts accountInformation;  
  
  
 LoggedInMenuPage(Accounts accountInformation){  
  
 this.accountInformation=accountInformation;  
  
 container.setLayout(null);  
 bankInfo.setBounds(120,20,270,30);  
  
 accountNumberLabel.setBounds(20,90,150,25);  
 accountNumberValue.setBounds(170,90,170,25);  
  
 logOut.setBounds(340,290,90,30);  
 details.setBounds(20,60,150,20);  
  
 userNameLabel.setBounds(40,110,160,25);  
 userNameValue.setBounds(150,110,160,25);  
  
 userPhoneNumberLabel.setBounds(40,140,160,25);  
 userPhoneNumberValue.setBounds(150,140,160,25);  
  
 userAadharLabel.setBounds(40,170,160,25);  
 userAadharValue.setBounds(150,170,160,25);  
  
 userPANLabel.setBounds(40,200,160,25);  
 userPANValue.setBounds(150,200,160,25);  
  
 userDOBLabel.setBounds(40,230,160,25);  
 userDOBValue.setBounds(150,230,160,25);  
  
 userAddressLabel.setBounds(40,260,160,25);  
 userAddressValue.setBounds(150,260,160,25);  
  
 accountNomineeLabel.setBounds(40,290,160,25);  
 accountNomineeValue.setBounds(150,290,160,25);  
  
 accountBalanceLabel.setBounds(40,320,160,25);  
 accountBalanceValue.setBounds(150,320,160,25);  
  
 menu.setBounds(340,60,160,25);  
  
 withdrawAmount.setBounds(340,90,180,25);  
 depositAmount.setBounds(340,120,180,25);  
 transferAmount.setBounds(340,150,180,25);  
 updateAccountDetails.setBounds(340,180,180,25);  
 showStatement.setBounds(340,210,180,25);  
 proceed.setBounds(340,240,100,30);  
  
 container.add(bankInfo);  
  
 container.add(userNameLabel);  
 container.add(userNameValue);  
 container.add(userPhoneNumberLabel);  
 container.add(userPhoneNumberValue);  
  
 container.add(userAadharLabel);  
 container.add(userAadharValue);  
 container.add(userPANLabel);  
 container.add(userPANValue);  
  
 container.add(userDOBLabel);  
 container.add(userDOBValue);  
 container.add(userAddressLabel);  
 container.add(userAddressValue);  
  
 container.add(accountNomineeLabel);  
 container.add(accountNomineeValue);  
 container.add(accountBalanceLabel);  
 container.add(accountBalanceValue);  
  
  
 container.add(accountNumberLabel);  
 container.add(accountNumberValue);  
  
 container.add(menu);  
 container.add(details);  
  
 ButtonGroup loggedInGroup=new ButtonGroup();  
 loggedInGroup.add(withdrawAmount);  
 loggedInGroup.add(depositAmount);  
 loggedInGroup.add(transferAmount);  
 loggedInGroup.add(updateAccountDetails);  
 loggedInGroup.add(showStatement);  
  
 container.add(withdrawAmount);  
 container.add(depositAmount);  
 container.add(transferAmount);  
 container.add(updateAccountDetails);  
 container.add(showStatement);  
  
 container.add(proceed);  
 container.add(logOut);  
  
 setData();  
  
 proceed.addActionListener(e -> {  
 if (withdrawAmount.isSelected()) {  
 setVisible(false);  
 new TransactionPage(1,accountInformation);  
 } else if (depositAmount.isSelected()) {  
 setVisible(false);  
 new TransactionPage(2,accountInformation);  
 } else if (transferAmount.isSelected()) {  
 new TransferAmountPage(accountInformation);  
 } else if (updateAccountDetails.isSelected()) {  
 new UpdateAccountDetailsPage(accountInformation);  
 }else{  
 new showStatements(accountInformation.transactionsArrayList);  
 }  
 });  
  
 logOut.addActionListener(e -> {  
 setVisible(false);  
 new LoginPage();  
 });  
  
 setVisible(true);  
 setSize(590,420);  
 setTitle("BR14X BANK MANAGEMENT SYSTEM - LOGIN");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 }  
  
 private void setData(){  
 accountNumberValue.setText(String.valueOf(accountInformation.getAccountNumber()));  
 userNameValue.setText(accountInformation.getUserName());  
 userPhoneNumberValue.setText(accountInformation.getUserPhoneNumber());  
 userAadharValue.setText(accountInformation.getUserAadharNumber());  
 userPANValue.setText(accountInformation.getUserPAN());  
 userDOBValue.setText(String.valueOf(accountInformation.getUserDOB()));  
 userAddressValue.setText(accountInformation.getUserAddress());  
 accountNomineeValue.setText(accountInformation.getAccountNominee());  
 accountBalanceValue.setText(String.valueOf(accountInformation.getAccountBalance()));  
 }  
}

4) OpenNewAccountPage.java:-

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class OpenNewAccountPage extends JFrame {  
  
 Container container=getContentPane();  
  
 JLabel bankInfo=new JLabel("WELCOME , PLEASE FILL OUT FOLLOWING DETAILS !");  
  
 JLabel userNameLabel=new JLabel("ENTER FULL NAME");  
 JTextField userNameInput=new JTextField();  
  
 JLabel userPhoneNumberLabel=new JLabel("Provide PhoneNumber ");  
 JTextField userPhoneNumberInput=new JTextField();  
  
 JLabel userAadharLabel=new JLabel("Provide AadharCard Number");  
 JTextField userAadharInput=new JTextField();  
  
 JLabel userPANLabel=new JLabel("Provide PANCARD Number");  
 JTextField userPANInput=new JTextField();  
  
 JLabel userDOBLabel=new JLabel("ENTER Date OF Birth");  
 JTextField userDOBInput=new JTextField();  
  
 JLabel userAddressLabel=new JLabel("Provide Address");  
 JTextField userAddressInput=new JTextField();  
  
 JLabel accountNomineeLabel=new JLabel("Give Nominee For Account");  
 JTextField accountNomineeInput=new JTextField();  
  
 JLabel accountPasswordLabel=new JLabel("Create Unique Password");  
 JPasswordField accountPasswordInput=new JPasswordField();  
  
 JLabel accountCPasswordLabel=new JLabel("Confirm Password Again");  
 JPasswordField accountCPasswordInput=new JPasswordField();  
  
 JLabel accountBalanceLabel=new JLabel("Enter Account Balance");  
 JTextField accountBalanceInput=new JTextField();  
  
 JButton submitButton=new JButton("CREATE ACCOUNT");  
 JButton backToLoginButton=new JButton("<-- Go BACK TO LOGIN");  
  
 String userName,userPhoneNumber;  
 String userAadharNumber;  
 String accountBalance;  
 String accountPassword;  
 String accountNominee;  
 String postalAddress;  
 String userDOB;  
 String userPANNumber;  
 double accountBalanceDbl;  
 Date userD;  
  
 OpenNewAccountPage(){  
 container.setLayout(null);  
  
 bankInfo.setBounds(70,20,380,30);  
  
 userNameLabel.setBounds(40,60,160,25);  
 userNameInput.setBounds(220,60,160,25);  
  
 userPhoneNumberLabel.setBounds(40,90,160,25);  
 userPhoneNumberInput.setBounds(220,90,160,25);  
  
 userAadharLabel.setBounds(40,120,160,25);  
 userAadharInput.setBounds(220,120,160,25);  
  
 userPANLabel.setBounds(40,150,160,25);  
 userPANInput.setBounds(220,150,160,25);  
  
 userDOBLabel.setBounds(40,180,160,25);  
 userDOBInput.setBounds(220,180,160,25);  
  
 userAddressLabel.setBounds(40,210,160,25);  
 userAddressInput.setBounds(220,210,160,25);  
  
 accountNomineeLabel.setBounds(40,240,160,25);  
 accountNomineeInput.setBounds(220,240,160,25);  
  
 accountBalanceLabel.setBounds(40,270,160,25);  
 accountBalanceInput.setBounds(220,270,160,25);  
  
 accountPasswordLabel.setBounds(40,300,160,25);  
 accountPasswordInput.setBounds(220,300,160,25);  
  
 accountCPasswordLabel.setBounds(40,330,160,25);  
 accountCPasswordInput.setBounds(220,330,160,25);  
  
 submitButton.setBounds(125,400,150,30);  
  
 backToLoginButton.setBounds(105,450,180,30);  
  
 container.add(bankInfo);  
 container.add(userNameLabel);  
 container.add(userNameInput);  
 container.add(userPhoneNumberLabel);  
 container.add(userPhoneNumberInput);  
  
 container.add(userAadharLabel);  
 container.add(userAadharInput);  
 container.add(userPANLabel);  
 container.add(userPANInput);  
  
 container.add(userDOBLabel);  
 container.add(userDOBInput);  
 container.add(userAddressLabel);  
 container.add(userAddressInput);  
  
 container.add(accountNomineeLabel);  
 container.add(accountNomineeInput);  
 container.add(accountPasswordLabel);  
 container.add(accountPasswordInput);  
  
 container.add(accountBalanceLabel);  
 container.add(accountBalanceInput);  
  
 container.add(accountPasswordLabel);  
 container.add(accountPasswordInput);  
 container.add(accountCPasswordLabel);  
 container.add(accountCPasswordInput);  
  
 container.add(submitButton);  
 container.add(backToLoginButton);  
  
 submitButton.addActionListener(e -> {  
  
 boolean stat=false;  
 userName=userNameInput.getText();  
 userPhoneNumber=userPhoneNumberInput.getText();  
 userAadharNumber=userAadharInput.getText();  
 userPANNumber = userPANInput.getText();  
 userDOB = userDOBInput.getText();  
 postalAddress = userAddressInput.getText();  
 accountNominee = accountNomineeInput.getText();  
 accountPassword =accountPasswordInput.getText();  
 accountBalance=accountBalanceInput.getText();  
  
 if(userName.isEmpty()){  
 sendErrorMessage("Empty User Name");  
 }else {  
 if (userPhoneNumber.isEmpty()) {  
 sendErrorMessage("Empty Phone Number");  
 }else {  
 do {  
 if(userPhoneNumber.length()!=10){  
 sendErrorMessage("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{  
 sendErrorMessage("Phone Number Should Not Contain Any Letters");  
 System.out.println();  
 userPhoneNumber=null;  
 stat=true;  
 }  
 }  
 }  
  
 }while(stat);  
 if (userAadharNumber.isEmpty()) {  
 sendErrorMessage("Empty Aadhar Number");  
 } else {  
 do {  
 if(userAadharNumber.length()!=12){  
 sendErrorMessage("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{  
 sendErrorMessage("Aadhar Number Should Not Contain Any Letters");  
 System.out.println();  
 userAadharNumber=null;  
 stat=true;  
 }  
 }  
 }  
 }while(stat);  
  
 if (userPANNumber.isEmpty()) {  
 sendErrorMessage("Empty PAN Number");  
 } else {  
 if (userDOB.isEmpty()) {  
 sendErrorMessage("Empty DOB");  
 } else {  
 String date =userDOB;  
  
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");  
 userD=null;  
 try {  
 userD = dateFormat.parse(date);  
 } catch (ParseException e2) {  
 sendErrorMessage("Enter in DD-MM-YYYY");  
 }  
 if (postalAddress.isEmpty()) {  
 sendErrorMessage("Empty Postal Address");  
 } else {  
 if (accountNominee.isEmpty()) {  
 sendErrorMessage("Empty Account Nominee");  
 } else {  
 if (accountPassword.isEmpty()) {  
 sendErrorMessage("Empty Password");  
 }  
 else {  
 boolean stat1=false;  
 String pass1=accountPassword;  
 do {  
 //System.out.print("\nPlease Enter New Password \nShould be of minimum size 8 with Letter, Digit And Special Characters\n(Remember For Next Login) : ");  
 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 {  
 sendErrorMessage("Password must contain Letters, minimum 1 Digit And 1 Special Characters");  
 pass1 = null;  
 stat1=true;  
 }  
  
 } else {  
 sendErrorMessage("Invalid Password Should be of minimum size 8.");  
 stat1=true;  
 }  
 }while (stat1);  
  
 if (accountBalance.isEmpty()) {  
 sendErrorMessage("Empty Account Balance");  
 }else{  
 accountBalanceDbl = Double.parseDouble(accountBalance);  
 if(accountBalanceDbl<3000){  
 sendErrorMessage("Balance Should Be Above 3000");  
 }else{  
 addNewUser();  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 });  
  
 backToLoginButton.addActionListener(e -> {  
 new LoginPage();  
 setVisible(false);  
 });  
 setVisible(true);  
 setSize(500,550);  
 setTitle("BR14X BANK SYSTEM -OPEN ACCOUNT");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 }  
  
 void addNewUser(){  
 ArrayList<Transactions> transactions=new ArrayList<>();  
 Accounts account=new Accounts(0,userName,userPANNumber,postalAddress,accountNominee,userPhoneNumber,userAadharNumber,accountPassword,userD,accountBalanceDbl,transactions);  
 try {  
 int send=1;  
 LoginPage.objectOutputStream.writeInt(send);  
 LoginPage.objectOutputStream.flush();  
  
 LoginPage.objectOutputStream.writeObject(account);  
 LoginPage.objectOutputStream.flush();  
  
 String msg=(String) LoginPage.objectInputStream.readObject();//Print Message From Server  
 sendMessage(msg);  
 new LoginPage();  
 setVisible(false);  
  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 void sendErrorMessage(String msg){  
 JOptionPane.showMessageDialog(null, msg, "Error",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
 void sendMessage(String msg){  
 JOptionPane.showMessageDialog(null,  
 msg,  
 "Success",  
 JOptionPane.PLAIN\_MESSAGE);  
 }  
}

5)showStatements.java:-

package com.company;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.util.ArrayList;  
  
public class showStatements extends JFrame {  
  
 JTable table = new JTable();  
  
 JScrollPane scroll;  
  
 String headers[] = { "Date", "Amount", "Balance", "Type", "Description" };  
  
 showStatements(ArrayList<Transactions> transactionsArrayList){  
 DefaultTableModel model = new DefaultTableModel(new String[]{"Date", "Amount", "Balance", "Type", "Description"},0);  
  
 for (Transactions transactions : transactionsArrayList) {  
 model.addRow(new Object[]  
 {  
 transactions.getTransactionDateAndTime(),  
 transactions.getTransactionAmount(),  
 transactions.getAccountBalance(),  
 transactions.getTransactionType(),  
 transactions.getTransactionDescription()  
 });  
 }  
  
 table.setModel(model);  
 table.setBounds(100,100,1000,400);  
 scroll = new JScrollPane(table);;  
 add(scroll);  
 setSize(1010, 400);  
 setVisible(true);  
 }  
}

6)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;  
 }  
}

7)TransactionPage.java:-

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import static com.company.LoginPage.objectInputStream;  
import static com.company.LoginPage.objectOutputStream;  
  
public class TransactionPage extends JFrame {  
  
 int type;  
  
 JLabel bankInfo=new JLabel("BR14X BANK MANAGEMENT SYSTEM");  
  
 JLabel accountNumberLabel=new JLabel("ACCOUNT NUMBER :");  
 JLabel accountNumberValue=new JLabel();  
  
  
 JLabel accountBalanceLabel=new JLabel("ACCOUNT BALANCE :");  
 JLabel accountBalanceValue=new JLabel();  
  
 JLabel amountTransferLabel=new JLabel("AMOUNT:");  
 JTextField amountTransferInput=new JTextField();  
  
 JButton proceed=new JButton("Proceed");  
 JButton cancel=new JButton("EXIT");  
  
 static JButton returnBack=new JButton("Exit");  
  
 Container container=getContentPane();  
  
 Accounts accountInformation;  
  
 TransactionPage(int type, Accounts loggedInAccount){  
 this.type=type;  
 this.accountInformation =loggedInAccount;  
  
 container.setLayout(null);  
  
 bankInfo.setBounds(50,5,270,30);  
 accountNumberLabel.setBounds(20,55,160,25);  
 accountNumberValue.setBounds(200,55,70,25);  
 accountBalanceLabel.setBounds(20,90,160,25);  
 accountBalanceValue.setBounds(200,90,70,25);  
  
 accountNumberValue.setText(String.valueOf(accountInformation.getAccountNumber()));  
 accountBalanceValue.setText(String.valueOf(accountInformation.getAccountBalance()));  
  
  
 amountTransferLabel.setBounds(70,120,160,25);  
 amountTransferInput.setBounds(180,120,160,25);  
  
 proceed.setBounds(20,170,100,25);  
 cancel.setBounds(20,230,90,25);  
  
 container.add(accountNumberLabel);  
 container.add(accountNumberValue);  
 container.add(accountBalanceLabel);  
 container.add(accountBalanceValue);  
  
 container.add(amountTransferLabel);  
 container.add(amountTransferInput);  
 container.add(proceed);  
 container.add(cancel);  
  
  
 proceed.addActionListener(e -> {  
 String amount=amountTransferInput.getText();  
 if(!amount.isEmpty()){  
 switch (type){  
 case 1:withdrawAmount(amount);  
 break;  
 case 2:depositAmount(amount);  
 }  
 }  
 });  
  
 cancel.addActionListener(e -> {  
 setVisible(false);  
 new LoggedInMenuPage(loggedInAccount);  
 });  
  
 returnBack.addActionListener(e -> {  
 setVisible(false);  
 new LoggedInMenuPage(loggedInAccount);  
 });  
  
  
 setVisible(true);  
 setSize(430,320);  
 setTitle("BR14X BANK MANAGEMENT SYSTEM");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 }  
  
  
 private void withdrawAmount(String amt) {  
  
 double amount = accountInformation.getAccountBalance();  
 double withdrawAmount=Double.parseDouble(amt);  
  
 if (withdrawAmount > amount) {  
 sendErrorMessage("Account Balance is low to withdraw amount " + withdrawAmount);  
 } else {  
 try {  
 objectOutputStream.writeInt(5);  
 objectOutputStream.flush();  
 Trans\_Req request = new Trans\_Req(accountInformation.accountNumber, accountInformation.accountNumber, withdrawAmount);  
 objectOutputStream.writeObject(request);  
 objectOutputStream.flush();  
 String msg = (String) objectInputStream.readObject();  
 if (msg.equals("Successful")) {  
 amount -= withdrawAmount;  
 accountInformation.setAccountBalance(amount);  
 sendMessage("Amount Withdrawn \nAccount Balance : " + amount);  
 addTransactionStatement(accountInformation, withdrawAmount, amount, "Self", "Self Withdraw");  
 }else{  
 sendErrorMessage("Unsuccessful | Try Again");  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private void depositAmount(String amt){  
 double amount = accountInformation.getAccountBalance();  
 double depositAmount=Double.parseDouble(amt);  
 try {  
 objectOutputStream.writeInt(6);  
 objectOutputStream.flush();  
 Trans\_Req request=new Trans\_Req(accountInformation.accountNumber, accountInformation.accountNumber,depositAmount);  
 objectOutputStream.writeObject(request);  
 objectOutputStream.flush();  
 String msg=(String)objectInputStream.readObject();  
 if(msg.equals("Successful")){  
 amount += depositAmount;  
 accountInformation.setAccountBalance(amount);  
 sendMessage("Amount Deposited \nAccount Balance : " + amount);  
 addTransactionStatement(accountInformation,depositAmount,amount,"Self","Self Deposit");  
 }else{  
 sendErrorMessage("Unsuccessful | Try Again");  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void addTransactionStatement(Accounts loggedInAccount, double amount, double balance, String type, String description){  
  
 SimpleDateFormat formattedDate = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss a");  
 String date = formattedDate.format(System.currentTimeMillis());  
  
 Transactions transactions=new Transactions(date,amount,balance,type,description);  
 loggedInAccount.transactionsArrayList.add(transactions);  
  
 }  
 void sendErrorMessage(String msg){  
 JOptionPane.showMessageDialog(null, msg, "Error",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
 void sendMessage(String msg){  
 JOptionPane.showMessageDialog(null,  
 msg,  
 "Success",  
 JOptionPane.PLAIN\_MESSAGE);  
 }  
}

8)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(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 String getTransactionDateAndTime() {  
 return transactionDateAndTime;  
 }  
  
 public double getTransactionAmount() {  
 return transactionAmount;  
 }  
  
 public double getAccountBalance() {  
 return accountBalance;  
 }  
  
 public String getTransactionType() {  
 return transactionType;  
 }  
  
 public String getTransactionDescription() {  
 return transactionDescription;  
 }  
}

9)TransferAmountPage.java:-

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
import java.text.SimpleDateFormat;  
import static com.company.LoginPage.objectInputStream;  
import static com.company.LoginPage.objectOutputStream;  
  
public class TransferAmountPage extends JFrame {  
  
 JLabel bankInfo=new JLabel("BR14X BANK MANAGEMNT SYSTEM");  
  
 JLabel accountNumberLabel=new JLabel("ACCOUNT NUMBER :");  
 JLabel accountNumberValue=new JLabel();  
  
 JLabel accountBalanceLabel=new JLabel("ACCOUNT BALANCE :");  
 JLabel accountBalanceValue=new JLabel();  
  
 JLabel transferAccountNumberLabel=new JLabel("Transfer Account Number:");  
 JTextField transferAccountNumberInput=new JTextField();  
  
 JLabel amountTransferLabel=new JLabel("AMOUNT :");  
 JTextField amountTransferInput=new JTextField();  
  
 JButton proceed=new JButton("Proceed");  
 JButton cancel=new JButton("EXIT");  
  
 static JButton returnBack=new JButton("Exit");  
  
 Container container=getContentPane();  
 Accounts accountInformation;  
  
 TransferAmountPage(Accounts information){  
 this.accountInformation =information;  
  
 container.setLayout(null);  
  
 bankInfo.setBounds(70,5,270,30);  
 accountNumberLabel.setBounds(20,55,150,25);  
 accountNumberValue.setBounds(170,55,70,25);  
 accountBalanceLabel.setBounds(20,90,150,25);  
 accountBalanceValue.setBounds(170,90,70,25);  
  
 transferAccountNumberLabel.setBounds(20,120,160,25);  
 transferAccountNumberInput.setBounds(180,120,160,25);  
  
 amountTransferLabel.setBounds(20,160,160,25);  
 amountTransferInput.setBounds(180,160,160,25);  
  
 proceed.setBounds(20,190,100,25);  
 cancel.setBounds(20,230,90,20);  
  
 returnBack.setBounds(120,360,100,25);  
  
 accountNumberValue.setText(String.valueOf(accountInformation.getAccountNumber()));  
 accountBalanceValue.setText(String.valueOf(accountInformation.getAccountBalance()));  
  
 container.add(bankInfo);  
  
 container.add(accountNumberLabel);  
 container.add(accountNumberValue);  
 container.add(accountBalanceLabel);  
 container.add(accountBalanceValue);  
  
 container.add(amountTransferLabel);  
 container.add(amountTransferInput);  
 container.add(transferAccountNumberLabel);  
 container.add(transferAccountNumberInput);  
 container.add(proceed);  
 container.add(cancel);  
  
 proceed.addActionListener(e -> {  
 String accNo=transferAccountNumberInput.getText();  
 String amt=amountTransferInput.getText();  
  
 if(!accNo.isEmpty() && !amt.isEmpty()){  
 transferAmount(accNo,amt);  
 }  
 });  
  
 cancel.addActionListener(e -> {  
 setVisible(false);  
 new LoggedInMenuPage(information);  
 });  
  
 returnBack.addActionListener(e -> {  
 setVisible(false);  
 new LoggedInMenuPage(information);  
 });  
 setVisible(true);  
 setSize(430,300);  
 setTitle("BR14X BANK MANAGEMENT SYSTEM");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 }  
  
 private void transferAmount(String accNo,String amt){  
  
 int transAccNo=Integer.parseInt(accNo);  
 double amount=Double.parseDouble(amt);  
 String msg="@";  
 System.out.println("TRANSFER MONEY PORTAL");  
  
 try {  
 objectOutputStream.writeInt(4);  
 objectOutputStream.flush();  
  
 Trans\_Req request=new Trans\_Req(accountInformation.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")){  
 accountInformation.setAccountBalance((accountInformation.accountBalance-amount));  
 addTransactionStatement(accountInformation,amount,(accountInformation.accountBalance-amount),"Transfer","Transferred To "+transAccNo);  
 sendMessage();  
 }else{  
 sendErrorMessage();  
 }  
 }  
  
 public static void addTransactionStatement(Accounts loggedInAccount, double amount, double balance, String type, String description){  
  
 SimpleDateFormat formattedDate = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss a");  
 String date = formattedDate.format(System.currentTimeMillis());  
  
 Transactions transactions=new Transactions(date,amount,balance,type,description);  
 loggedInAccount.transactionsArrayList.add(transactions);  
 }  
  
 void sendErrorMessage(){  
 JOptionPane.showMessageDialog(null, "Unsuccessful | Try Again", "Error",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
 void sendMessage(){  
 JOptionPane.showMessageDialog(null,  
 "Transfer Amount Successful",  
 "Success",  
 JOptionPane.PLAIN\_MESSAGE);  
 }  
}

10) UpdateAccountDetailsPage.java:-

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.IOException;  
  
import static com.company.LoginPage.objectInputStream;  
import static com.company.LoginPage.objectOutputStream;  
  
public class UpdateAccountDetailsPage extends JFrame {  
  
 Container container=getContentPane(); //provides a space where a component can be located  
 //creating instances of JSwing Components

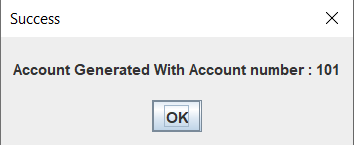
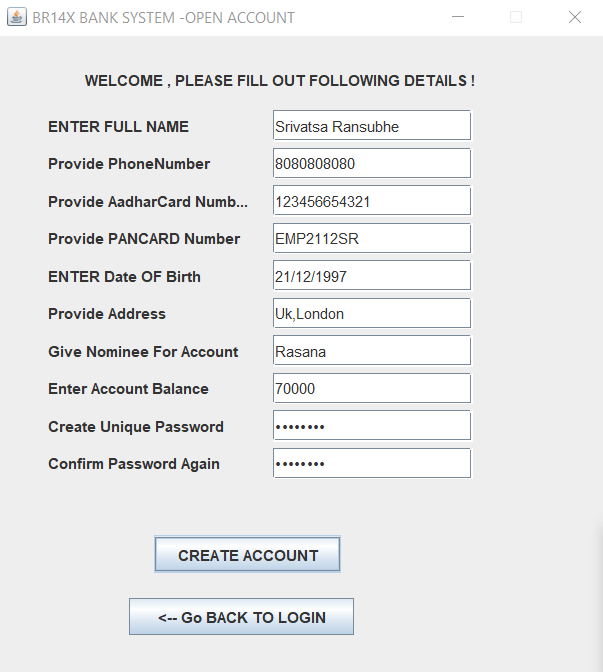
JLabel bankInfo=new JLabel("BR14X BANK MANAGEMENT SYSTEM - UPDATE DETAILS PAGE");  
 JLabel accountNumberLabel=new JLabel("ACCOUNT NUMBER :");  
 JLabel accountNumberValue=new JLabel();  
  
 JLabel userNameLabel=new JLabel("Name");  
 JLabel userNameValue=new JLabel();  
  
 JLabel userPhoneNumberLabel=new JLabel("Phone Number ");  
 JTextField userPhoneNumberInput=new JTextField();  
  
 JLabel userAadharLabel=new JLabel("Aadhar Number");  
 JTextField userAadharInput=new JTextField();  
  
 JLabel userPANLabel=new JLabel("PAN Number");  
 JTextField userPANInput=new JTextField();  
  
 JLabel userDOBLabel=new JLabel("DOB");  
 JLabel userDOBInput=new JLabel();  
  
 JLabel userAddressLabel=new JLabel("Address");  
 JTextField userAddressInput=new JTextField();  
  
 JLabel accountNomineeLabel=new JLabel("Account Nominee");  
 JTextField accountNomineeInput=new JTextField();  
  
 JLabel accountBalanceLabel=new JLabel("Account Balance");  
 JLabel accountBalanceValue=new JLabel();  
  
  
 JButton submitButton=new JButton("PROCEED");  
 JButton cancel=new JButton("Cancel");  
  
 private String userPhoneNumber;  
 private String userName;  
 private String userAadharNumber;  
 private String accountNominee;  
 private String postalAddress;  
 private String userPANNumber;  
  
 Accounts accountInformation;  
  
 UpdateAccountDetailsPage(Accounts information){  
  
 this.accountInformation=information;  
  
 container.setLayout(null);  
  
 bankInfo.setBounds(40,20,380,30);  
  
 accountNumberLabel.setBounds(40,60,150,25);  
 accountNumberValue.setBounds(200,60,70,25);  
  
 userNameLabel.setBounds(40,110,160,25);  
 userNameValue.setBounds(150,110,160,25);  
  
 userPhoneNumberLabel.setBounds(40,140,160,25);  
 userPhoneNumberInput.setBounds(150,140,160,25);  
  
 userAadharLabel.setBounds(40,170,160,25);  
 userAadharInput.setBounds(150,170,160,25);  
  
 userPANLabel.setBounds(40,200,160,25);  
 userPANInput.setBounds(150,200,160,25);  
  
 userDOBLabel.setBounds(40,230,160,25);  
 userDOBInput.setBounds(150,230,160,25);  
  
 userAddressLabel.setBounds(40,260,160,25);  
 userAddressInput.setBounds(150,260,160,25);  
  
 accountNomineeLabel.setBounds(40,290,160,25);  
 accountNomineeInput.setBounds(150,290,160,25);  
  
 accountBalanceLabel.setBounds(40,320,160,25);  
 accountBalanceValue.setBounds(150,320,160,25);  
  
 submitButton.setBounds(145,350,100,25);  
  
 cancel.setBounds(145,400,90,20);  
  
 accountNumberValue.setText(String.valueOf(accountInformation.getAccountNumber()));  
 container.add(cancel);  
 container.add(bankInfo);  
 container.add(userNameLabel);  
 container.add(userNameValue);  
 container.add(userPhoneNumberLabel);  
 container.add(userPhoneNumberInput);  
  
 container.add(userAadharLabel);  
 container.add(userAadharInput);  
 container.add(userPANLabel);  
 container.add(userPANInput);  
  
 container.add(userDOBLabel);  
 container.add(userDOBInput);  
 container.add(userAddressLabel);  
 container.add(userAddressInput);  
  
 container.add(accountNomineeLabel);  
 container.add(accountNomineeInput);  
  
 container.add(accountBalanceLabel);  
 container.add(accountBalanceValue);  
  
 container.add(accountNumberLabel);  
 container.add(accountNumberValue);  
  
 container.add(submitButton);  
  
 setData();  
  
 submitButton.addActionListener(e -> validateData());  
  
 cancel.addActionListener(e -> {  
 setVisible(false);  
 new LoggedInMenuPage(information);  
 });  
  
 setVisible(true); //making the frame visible

setSize(500,500); ////setting width and height

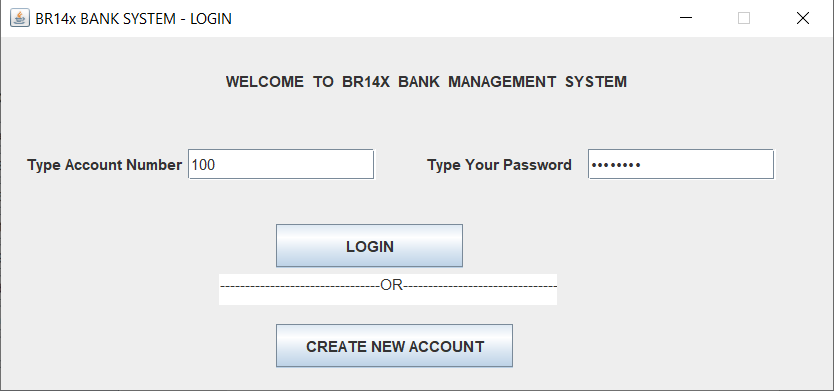
setTitle("BANK MANAGEMENT SYSTEM");  
 setResizable(false);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 }  
  
 private void validateData(){  
  
 boolean stat=false;  
  
 userPhoneNumber=userPhoneNumberInput.getText();  
 userAadharNumber=userAadharInput.getText();  
 userPANNumber = userPANInput.getText();  
 String userDOB = userDOBInput.getText();  
 postalAddress = userAddressInput.getText();  
 accountNominee = accountNomineeInput.getText();  
 if (userPhoneNumber.isEmpty()) {  
 sendErrorMessage("Empty Phone Number");  
 }else {  
 do {  
 if(userPhoneNumber.length()!=10){  
 sendErrorMessage("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{  
 sendErrorMessage("Phone Number Should Not Contain Any Letters");  
 System.out.println();  
 userPhoneNumber=null;  
 stat=true;  
 }  
 }  
 }  
  
 }while(stat);  
 if (userAadharNumber.isEmpty()) {  
 sendErrorMessage("Empty Aadhar Number");  
 } else {  
 do {  
 if(userAadharNumber.length()!=12){  
 sendErrorMessage("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{  
 sendErrorMessage("Aadhar Number Should Not Contain Any Letters");  
 System.out.println();  
 userAadharNumber=null;  
 stat=true;  
 }  
 }  
 }  
 }while(stat);  
  
 if (userPANNumber.isEmpty()) {  
 sendErrorMessage("Empty PAN Number");  
 } else {  
 if (userDOB.isEmpty()) {  
 sendErrorMessage("Empty DOB");  
 } else {  
 if (postalAddress.isEmpty()) {  
 sendErrorMessage("Empty Postal Address");  
 } else {  
 if (accountNominee.isEmpty()) {  
 sendErrorMessage("Empty Account Nominee");  
 } else {  
 updateDatabase();  
  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 private void updateDatabase(){  
 accountInformation.setUserPhoneNumber(userPhoneNumber);  
 accountInformation.setUserAadharNumber(userAadharNumber);  
 accountInformation.setAccountNominee(accountNominee);  
 accountInformation.setUserAddress(postalAddress);  
 // accountInformation.setUserDOB(userD);  
 accountInformation.setUserPAN(userPANNumber);  
  
 try { //Update in Server  
 objectOutputStream.writeInt(3);  
 objectOutputStream.writeObject(accountInformation);  
 objectOutputStream.flush();  
 String message=(String)objectInputStream.readObject();  
 sendMessage(message);  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void setData(){  
   
 accountNumberValue.setText(String.valueOf(accountInformation.getAccountNumber()));  
  
 userPhoneNumberInput.setText(accountInformation.getUserPhoneNumber());  
 userAadharInput.setText(accountInformation.getUserAadharNumber());  
 userPANInput.setText(accountInformation.getUserPAN());  
  
 String[] arr;  
  
 arr=String.valueOf(accountInformation.getUserDOB()).split("-");  
  
 String newDOB=arr[2]+"/"+arr[1]+"/"+arr[0];  
  
 userDOBInput.setText(newDOB);  
 userAddressInput.setText(accountInformation.getUserAddress());  
 accountNomineeInput.setText(accountInformation.getAccountNominee());  
 accountBalanceValue.setText(String.valueOf(accountInformation.getAccountBalance()));  
 }  
 void sendErrorMessage(String msg){  
 JOptionPane.showMessageDialog(null, msg, "Error",  
 JOptionPane.ERROR\_MESSAGE);  
 }  
 void sendMessage(String msg){  
 JOptionPane.showMessageDialog(null,  
 msg,  
 "Success",  
 JOptionPane.PLAIN\_MESSAGE);  
 }  
}

**OUTPUT:**

**1)CREATING ACCOUNT:-**

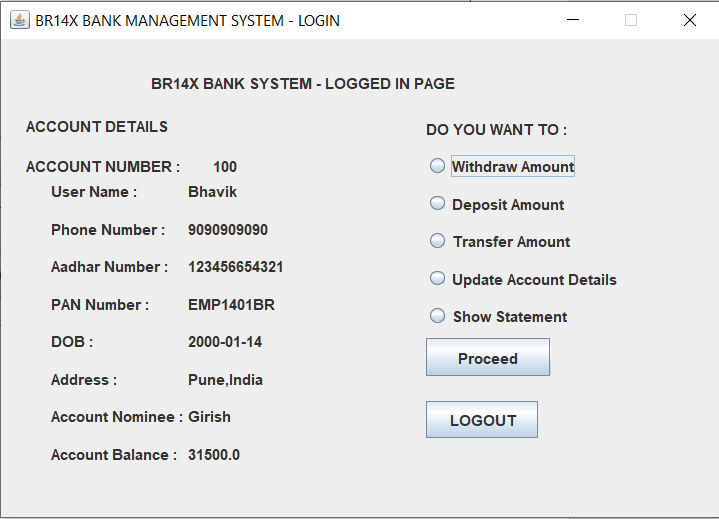


**2)LOGIN PAGE:-**

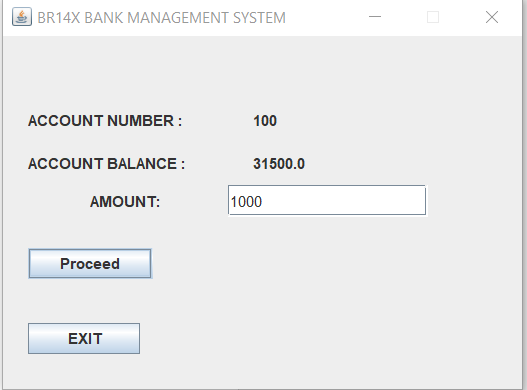
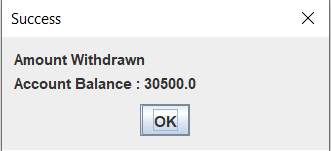


**3)Logged In Page :-**

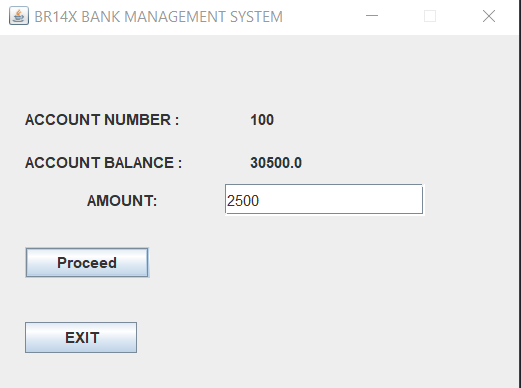
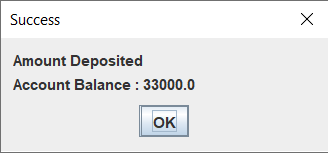
*Showing Account Details And Performing Transactions:*



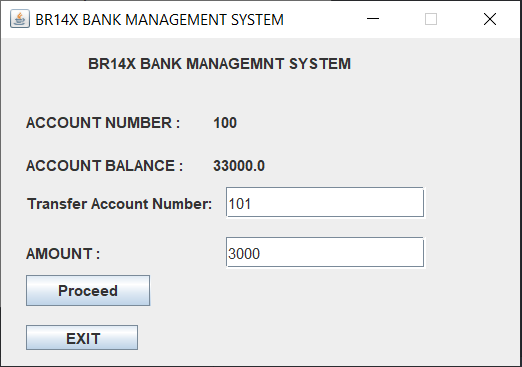
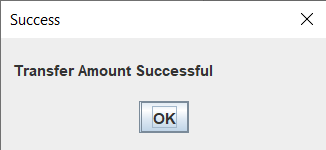
**4)Withdrawing Amount:-**

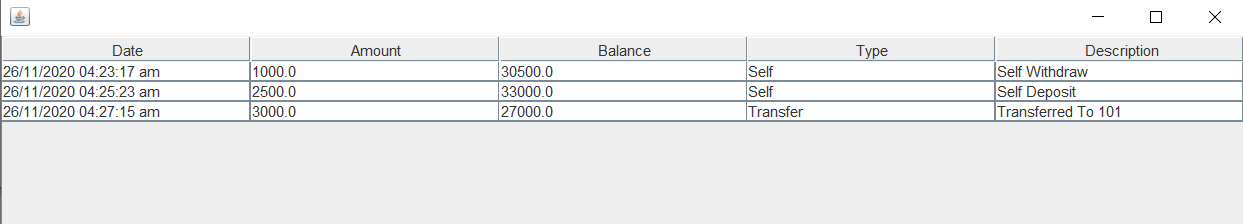
**5)Depositing Amount:-**

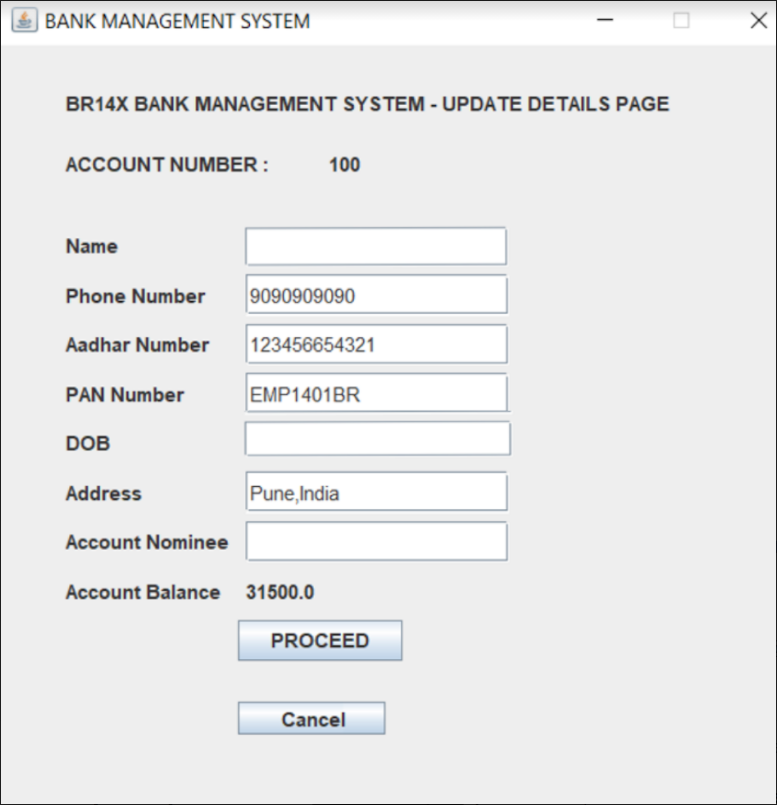
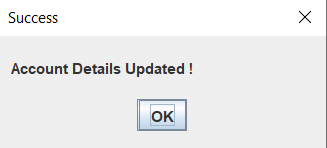
**6)Transfer Amount:**

**7)Showing Statement:-**



**8)Updating Details:**

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

**END**

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