

PROJECT REPORT
ON
BANK MANAGEMENT SYSTEM

(Session 2015-2016)

Made By:

**Yash Bhatnagar
& Yash Kaveeshwar**

Under the guidance of

Mrs. Lalita Chatterjee

For the partial fulfillment of class XII CBSE

Submitted to

JAWAHARLAL NEHRU SCHOOL,

B.H.E.L. BHOPAL

ACKNOWLEDGEMENT

We are thankful to our teacher cum guide “Mrs. Lalita Chatterjee” for her most inspiring guidance and encouragement. It is her guidance and inspiration which cleared rocks of difficulties during the completion of this practical file.

We shall also pay our gratitude to our parents and friends for their kind support without which this practical file was difficult to get completed.

Yash Bhatnagar

Yash Kaveeshwar

CERTIFICATE

This is to certify that “Yash Bhatnagar and Yash Kaveeshwar” has successfully completed the project file with help of Java Programming Language and MySQL under my guidance and supervision.

I am satisfied with their initiative and efforts on the completion of this project file as a part of CBSE Class XII Examination.

Mrs. Lalita Chatterjee

1.1 Java and MySQL

An introduction to java –

Java is both a programming language as well as a platform. It is a programming language as it can be used to write various applications and a platform where the same application can be developed, designed and can be made secure and highly interactive.

History of java –

Java started as an elite project with a code name as “*Green*” to find a way for allowing different devices to use a common language.

This language was originally named as “Oak” and later named as java. James Gosling developed “Oak”. Later java was incorporated by different web browsers like Internet Explorer, Mozilla Firefox and Opera.

Features-

1. WORA (Write Once Run Anywhere) - Java programs are written once and can be run on different platforms.
2. Light weight code – With java big and powerful programs can be made with a very little coding.
3. Security – Java offers many security features to make the program safe and secure.
4. Built-in graphics – Java offers many graphic features which can be used to make our application visual.

5. OOP (Object Oriented Programming) – Object Oriented means we organize as a combination of different types of objects that incorporates both data and behavior. Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules. Basic concepts of OOPs are:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

6. Platform independent – Change of platform doesn't affect the functionality of a java application. This is because java compiler doesn't produce any native executable code for a particular machine but it generates java byte code which is machine interpretation for JVM (Java Virtual Machine).

JVM is a virtual machine which runs on a real system to execute the java byte code.

7. Open products – It is freely available on the net and can be downloaded from - <https://java.com/download> .

8. *Java is RAD – Java is RAD (Rapid Application Development) language i.e. using java one can develop an application in a shorter time than any other conventional programming language.*

9. Multi-threaded - A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

Java Packages -

A package is a collection of related java classes and interfaces. The following list, gives examples of some java packages and what they cover.

- JAVA.IO- Classes those manage reading data from input streams and writing data to the output streams.
- JAVA.AWT- Classes that manage user interface components such as windows, dialog boxes, buttons, check boxes, lists, menus, scrollbars, and text fields; the 'AWT' stands Abstract Window Toolkit.
- JAVA.APPLET- The applet class, which provides the ability to write applets, this package also includes several interfaces that connect an applet to its document and to resources for playing audio.
- JAVA.AWT.EVENT- GUIs are event driven; it means they generate events when the user of the program interacts with the GUI.
- JAVAX.SWING- This package enables the user to create interfaces which performs the GUI operations.

- JAVA.SQL- The JDBC API, classes and interfaces that access database and send SQL. In Java, packages serve as basis for building other package.

The NetBeans IDE –

NetBeans IDE is a free, open source, integrated development environment (IDE) that enables you to develop desktop, mobile and web applications. The IDE supports application development in various languages, including Java, HTML5, PHP and C++. The IDE provides integrated support for the complete development cycle, from project creation through debugging, profiling and deployment. The IDE runs on Windows, Linux, Mac OS X, and other UNIX-based systems.

The NetBeans Platform is a framework for simplifying the development of Java Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required.

Java - MySql connectivity (JDBC) -

JDBC is an API that provides universal database access for the Java programming language. JDBC is trademark name and is not an acronym. But JDBC is often thought of as standing for “Java Database Connectivity”.

JDBC allows Java programs to interact with any SQL-compliant database. Sincerely, nearly all RDBMSs support SQL, and because Java itself runs on most platforms, JDBC makes it possible to write a single database application that can run on different platforms and interact with different DBMSs.

SUN prepares and maintains the JDBC specifications. Since JDBC is just a specification, third-party vendors develop JDBC drivers adhering to this specification. JDBC developers then use these drivers to access data sources.

Features of JDBC –

1. Flexibility to businesses: With the help of JDBC technology, businesses can continue to use installed databases and access information easily.
2. Easy and economical: The JDBC is simple to learn, easy to deploy and inexpensive to maintain.
3. No configuration is required on client side because with JDBC API the connection is completely defined by the JDBC URL.
4. Full access to metadata.

5. Portable and easy maintainable code by using Internet-standard URLs to identify database connections.

JDBC Drivers –

There are four types of JDBC drivers. Commonest and most efficient of which are type 4 drivers.

- **JDBC Type 1 Driver.** These are JDBC-ODBC Bridge drivers. They delegate the work of data access to ODBC API. These are slowest of all.
- **JDBC Type 2 Driver.** They mainly use native API for data access and provide Java wrapper classes to be able to invoked using JDBC drivers.
- **JDBC Type 3 Driver.** They are written in 100% java and use vendor independent Net-protocol to access a vendor dependent ones .This extra step adds complexity and decreases the data access efficiency.
- **JDBC Type 4 Driver.** They are also written in 100% java and are most efficient among all driver types.

The JDBC API -

In general, JDBC API does three things:

- Establishes connection with a database
- Sends SQL statements
- Processes the results

➤ The JDBC API provides the following interfaces and classes –

- **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.
- **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 manage objects of this type. It also abstracts the details associated with working with Driver objects.
- **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.
- **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.
- **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.

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

An introduction to MySQL-

MySQL is a free open source RDBMS (Relational Database Management System). It can be downloaded from - <https://www.mysql.com/downloads/>. MySQL databases store information in a table and can have thousands of records in a single table.

MySQL was created and supported by MySQL AB Company based in Sweden. This company is now a subsidiary of Sun micro systems. Chief engineer of MySQL was Michael Widnium aka Monty. MySQL was named after his daughter My and later named as MySQL.

Key features of MySQL -

1. *Speed*
2. *Ease of use*
3. *Portability*
4. *Free of cost*
5. *Query language support*
6. *Security*
7. *Stability*
8. *Connectivity*
9. *Localization*

10. Client and tools

Processing capabilities of MySQL -

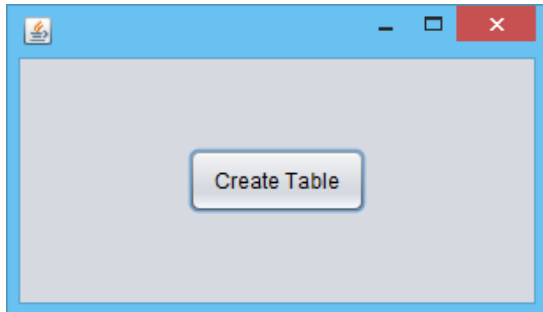
1. *DDL (Data Definition Language) – It provides commands for creation, modification and deletion of the tables and indexes .Ex – CREATE table command and ALTER table commands.*
2. *DML (Data Manipulation Language) – These commands are used to insert, modify and delete records in the table .Ex – SELECT command, INSERT command, and UPDATE command.*
3. *TCL (Transaction Control Language) – A transaction is complete unit of work. A transaction is successfully completed when all the steps of that transaction are successfully completed. The commands used to control a transaction are called TCL commands .Ex – COMMIT, ROLLBACK, SAVEPOINT commands.*
4. *DCL (Data Control Language) – These commands are used to grant or revoke access focusing a table .Ex – GRANT, REVOKE commands.*

1.2 Programming

Steps :

1. Create a database named bank.
2. Run the table frame from table package and click on 'create table' button.
3. Now run the Welcome frame from the default package.

The Table frame:



Click on create table button.

Import statements used:

```
import com.mysql.jdbc.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.Statement;
```

```
import javax.swing.JOptionPane;
```

Global variables:

```
String pwd="Jns";
```

Code:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
long Account=102581800;  
  
    try {  
  
        Class.forName("com.mysql.jdbc.Driver");  
  
        Connection con=(Connection)  
  
        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);  
  
        Statement stmt=con.createStatement();  
  
        int Bank_Table=stmt.executeUpdate("create table Bank_Table"  
  
            + "(Account_no varchar(20) primary key,"  
  
            + "Name_of_Account_Holder varchar(50),"  
  
            + "Balance varchar(20),"  
  
            + "Card_no varchar(20),"  
  
            + "Phone_no varchar(20),"  
  
            + "Date_of_Account_Creation date,"  
  
            + "Account_Type varchar(20))");  
  
        int Acc=stmt.executeUpdate("create table Accno"  
  
            + "(Acc varchar(20) primary key)");  
  
        int Accno=stmt.executeUpdate("insert into Accno values"  
  
            + "(102581820)");  
  
        int Bank_Account=stmt.executeUpdate("insert into bank_table"  
  
            +  
            "(Account_no,Name_of_Account_Holder,Balance,Card_no,Phone_no,Date_of_Account_Creatio  
n,Account_Type)"
```

+ "VALUES('102581801','Asha Sharma',530000,'100200580','9632758741','2010-09-04','Saving Account'),"

+ "('102581802','Ashish Mishra',746200,'100200581','9872456328','2011-07-04','Saving Account'),"

+ "('102581803','Amit Pathak',365900,'100200582','8745981264','2006-04-07','Saving Account'),"

+ "('102581804','Anmol Yadav',563000,'100200583','9124753681','2012-08-11','Current Account'),"

+ "('102581805','Rahul Singh',196000,'102586000','8723419853','2012-04-16','Saving Account'),"

+ "('102581806','Sunil Mishra',325000,'100258001','9172283671','2007-01-25','Current Account'),"

+ "('102581807','Anil Khanna',126340,'102500810','8126677291','2014-08-26','Saving Account'),"

+ "('102581808','Shyam Mishra',150500,'104500800','8845112769','1999-05-13','Saving Account'),"

+ "('102581809','Anand Dubey',96000,'100520080','8477521198','2013-04-07','Saving Account'),"

+ "('102581810','Anand Bakshi',563200,'152800200','9996135718','2013-04-07','Saving Account'),"

+ "('102581811','Sanjay Khanna',410000,'100240080','9912745581','2013-04-08','Saving Account'),"

+ "('102581812','Anand Bakshi',198522,'100500580','9877413528','2010-05-17','Saving Account'),"

+ "('102581813','Sapan Dubey',130299,'100250081','8884124793','2009-02-24','Current Account'),"

+ "('102581814','Alok Sharma',23500,'102058100','9841225748','2010-05-21','Current Account'),"

+ "('102581815','Amit Khanna',5995,'100205081','9871245781','2013-05-01','Saving Account'),"


```
        + "('102581816','Anil Yadav',96320,'100200501','2562230111','2001-06-21','Saving  
Account'),"
```

```
        + "('102581818','Anil Kumar',326000,'100200581','9685124781','2010-05-  
01','Current Account'),"
```

```
        + "('102581819','Ashish Dubey',36200,'150020081','2235142111','2013-05-  
07','Saving Account'),"
```

```
        + "('102581820','Ashish Mishra',36200,'100502800','9652102475','2013-04-  
21','Saving Account')");
```

```
    } catch (Exception e) {
```

```
        JOptionPane.showMessageDialog(this, e.getMessage());
```

```
    }
```

```
}
```

The Welcome frame :



Import statements:

```
import java.awt.Color;  
  
import java.awt.Dimension;  
  
import java.net.URL;  
  
import javax.swing.ImageIcon;
```

Constructor:

```
public Welcome() {  
  
    initComponents();  
  
    this.setSize(660, 515);  
  
    this.setLocation(300,120);  
  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
  
    ImageIcon icon = new ImageIcon(resource);  
  
    this.setIconImage(icon.getImage());  
  
}
```

Codes:

```
private void ContinueMouseEntered(java.awt.event.MouseEvent evt) {  
  
    Continue.setForeground(Color.WHITE);  
  
}  
  
private void ContinueMouseExited(java.awt.event.MouseEvent evt) {  
  
    Continue.setForeground(Color.BLACK);  
  
}  
  
private void ContinueMouseClicked(java.awt.event.MouseEvent evt) {  
  
    new Login().setVisible(true);  
  
    this.setVisible(false);  
  
}
```

On clicking Continue of the welcome frame it directs to secure frame:



On entering two times wrong password:



If we enter wrong password one more time the frame will get closed automatically and if we enter the correct user name and password it directs us to the Bank management frame

The user name is: 'yash' and password is: '12'

Import statements:

```
import javax.swing.JOptionPane;
```

```
import java.awt.event.KeyEvent;
```

```
import java.net.URL;
```

```
import javax.swing.ImageIcon;
```

Constructor

```
public Login() {  
    initComponents();  
    this.setLocation(260,120);  
    this.setSize(745,479);  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
    ImageIcon icon = new ImageIcon(resource);  
    this.setIconImage(icon.getImage());  
}
```

Methods

```
public int log=0;  
private int login(){  
    String Username = NameTF.getText().trim();  
    String Password = new String(PassPF.getPassword()).trim();  
    if ("yash".equals(Username)&&"12".equals>Password))  
    {  
        new Bank_management_system().setVisible(true);  
        this.setVisible(false);  
    }  
    else
```

```

{
log++;
NameTF.setText("");
PassPF.setText("");
if (log==1) {
    JOptionPane.showMessageDialog(this, "Incorrect Details\nTwo chances left");
}
if (log==2) {
    JOptionPane.showMessageDialog(this, "Incorrect Details\nOne chance left");
}
else if (log==3) {
    JOptionPane.showMessageDialog(this, "Incorrect Details\nTerminating Session");
    System.exit(0);
}
}
Return log;
}

```

Code

```

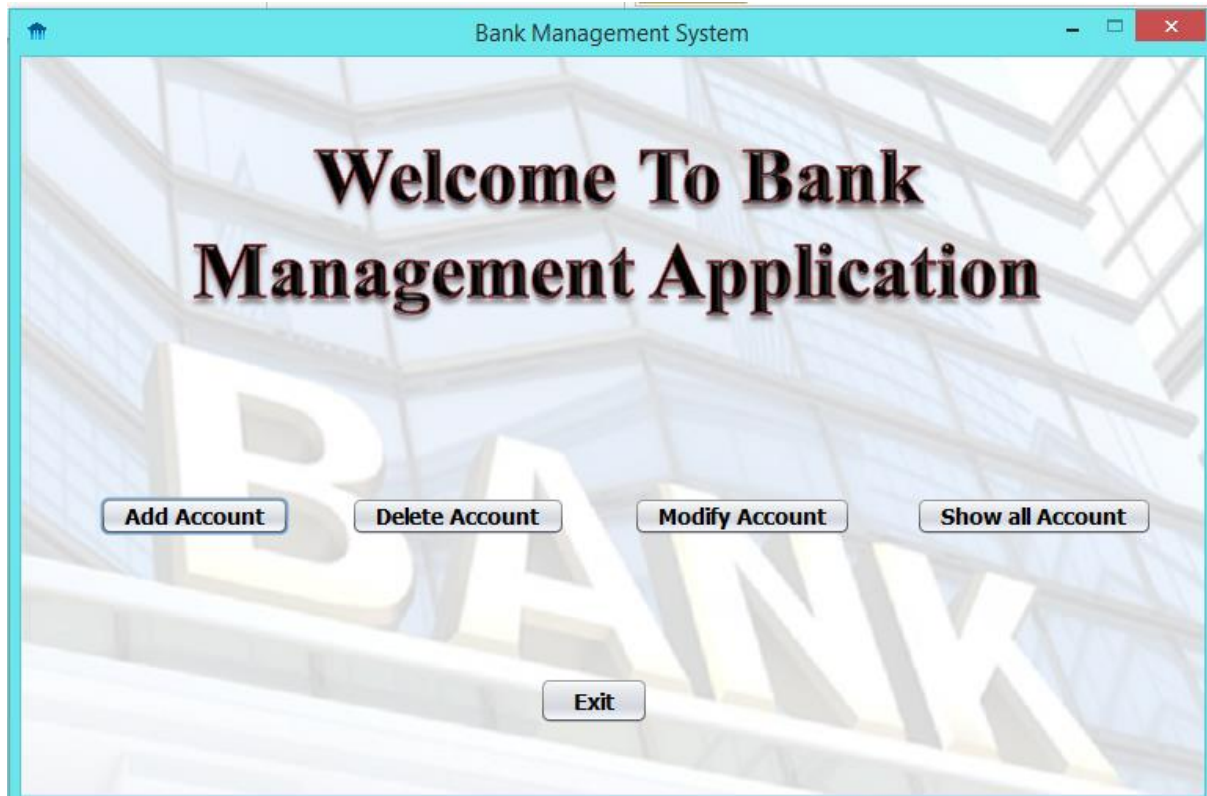
private void LoginBTNActionPerformed(java.awt.event.ActionEvent evt) {
login();
}

private void PassPFKeyPressed(java.awt.event.KeyEvent evt) {
if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode()))) {
login();
}
}

```

```
}  
  
private void LoginBTNKeyPressed(java.awt.event.KeyEvent evt) {  
    if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode()))) {  
        login();  
    }  
  
    private void ExitBTNActionPerformed(java.awt.event.ActionEvent evt) {  
        System.exit(0);  
    }  
}
```

This is our main frame:



Import statement

```
import java.awt.event.KeyEvent;  
  
import java.net.URL;  
  
import javax.swing.ImageIcon;
```

Constructor

```
public Bank_management_system() {  
  
    initComponents();  
  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
  
    ImageIcon icon = new ImageIcon(resource);  
  
    this.setIconImage(icon.getImage());  
  
    this.setSize(760, 500);  
  
    this.setLocation(260,120);  
  
}
```

Codes:

```
private void AddBTNActionPerformed(java.awt.event.ActionEvent evt) {  
new Add_Account().setVisible(true); // commands for displaying frame  
  
    this.setVisible(false);  
  
}  
  
private void AddBTNKeyPressed(java.awt.event.KeyEvent evt) {  
if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode()))) {  
new Add_Account().setVisible(true); // commands for displaying frame  
  
    this.setVisible(false);  
  
    }  
  
}
```

```

private void DeleteBTNActionPerformed(java.awt.event.ActionEvent evt) {

    new Delete_Acccount().setVisible(true); // commands for displaying frame

    this.setVisible(false);

}

private void DeleteBTNKeyPressed(java.awt.event.KeyEvent evt) {

if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {

new Delete_Acccount().setVisible(true); // commands for displaying frame

    this.setVisible(false);

    }

}

private void ModifyBTNActionPerformed(java.awt.event.ActionEvent evt) {

new Modify_Account().setVisible(true); // commands for displaying frame

    this.setVisible(false);

}

private void ModifyBTNKeyPressed(java.awt.event.KeyEvent evt) {

if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {

new Modify_Account().setVisible(true); // commands for displaying frame

    this.setVisible(false);

    }

}

private void ShowAllBTNActionPerformed(java.awt.event.ActionEvent evt) {

new Show_all_Account().setVisible(true); // commands for displaying frame

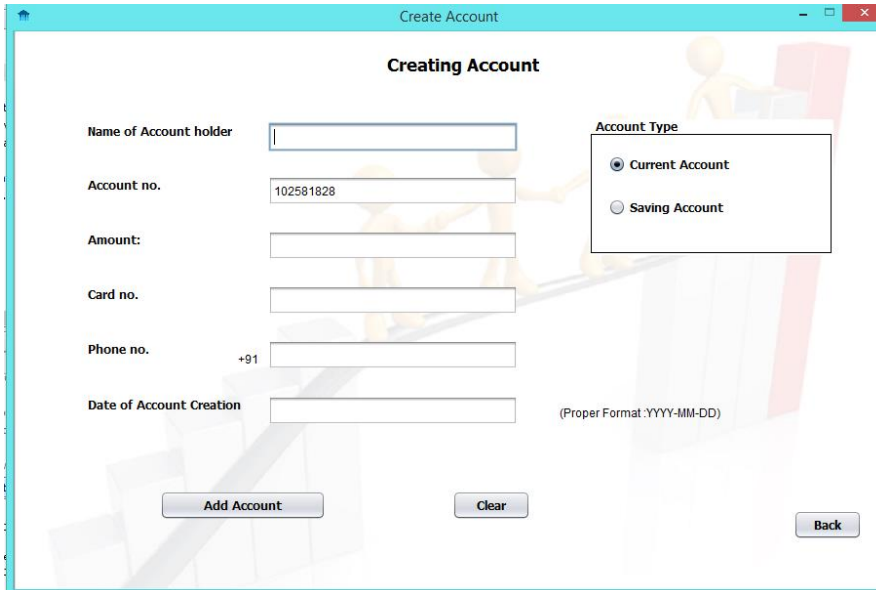
    this.setVisible(false);

}

```

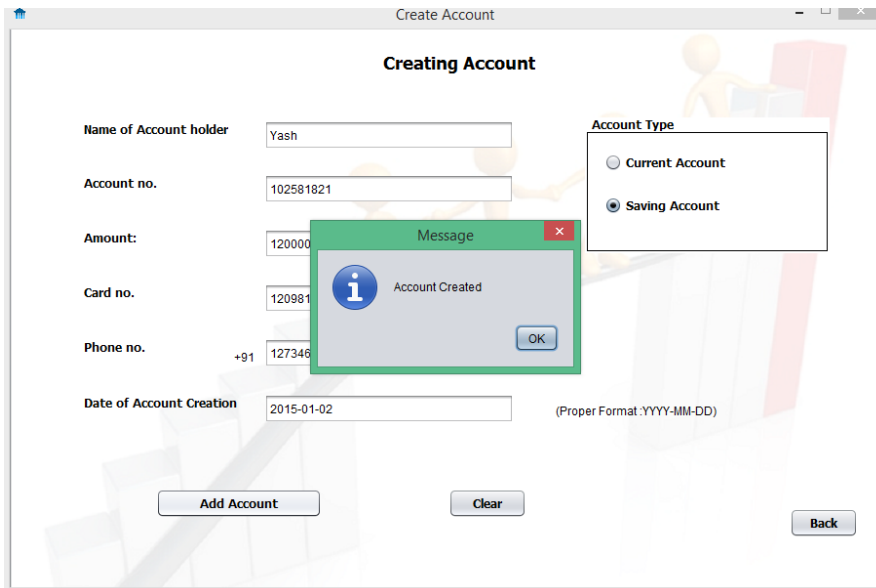
```
private void ShowAllBTNKeyPressed(java.awt.event.KeyEvent evt) {  
    if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {  
        new Show_all_Account().setVisible(true); // commands for displaying frame  
        this.setVisible(false);  
    }  
}  
  
private void ExitBTNActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0); // exit the application  
}
```

On clicking Add Account:



The screenshot shows a web application window titled "Create Account". Inside, there's a section titled "Creating Account". It contains several input fields: "Name of Account holder", "Account no." (with the value "102581828"), "Amount:", "Card no.", "Phone no." (with a "+91" prefix), and "Date of Account Creation" (with a note "(Proper Format :YYYY-MM-DD)"). To the right, there's a "Account Type" section with two radio buttons: "Current Account" (selected) and "Saving Account". At the bottom, there are three buttons: "Add Account", "Clear", and "Back".

On completing all the details properly and clicking Add Account button:



The screenshot shows the same "Create Account" form, but now all fields are filled: "Name of Account holder" is "Yash", "Account no." is "102581821", "Amount:" is "120000", "Card no." is "120981", "Phone no." is "+91 127346", and "Date of Account Creation" is "2015-01-02". The "Account Type" section now has "Saving Account" selected. A green "Message" dialog box is overlaid in the center, displaying an information icon, the text "Account Created", and an "OK" button. The "Add Account" button is still visible at the bottom.

Import statements:

```
import java.net.URL;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

global variables:

String pwd="Jns";
```

Constructor:

```
public Add_Account() {

    initComponents();

    this.setLocation(200,80);

    this.setSize(900,430);

    CurrentRB.setSelected(true);

    AccTF.setEditable(false);

    AccTF.setFocusable(false);

    URL resource = Login.class.getResource("Images/institution_icon.png");

    ImageIcon icon = new ImageIcon(resource);

    this.setIconImage(icon.getImage());

    ctd();

}
```

Methods used:

```
public void ctd()
{
String Acc="";

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        ResultSet rs=stmt.executeQuery("Select * from Accno");

        while (rs.next()) {

            Acc=rs.getString("Acc");

        }

        int Ac=Integer.parseInt(Acc);

        int ac=Ac+1;

        AccTF.setText(""+ac);

    } catch (Exception e) {

        JOptionPane.showMessageDialog(this, e.getMessage());

    }

}
```

Codes:

```
private void AddAccBTNActionPerformed(java.awt.event.ActionEvent evt) {  
String NameAcc,Balance,DoJ,Acctype = null,AccNo;  
  
long CardNo=0,Phone=0;  
  
int a=0,a1=0;  
  
    NameAcc = NameTF.getText();  
  
    AccNo = AccTF.getText();  
  
    Balance = BalanceTF.getText();  
  
    DoJ = DoJTF.getText();  
  
    SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd");  
  
    try {  
  
        Date Acda=df.parse(DoJ);  
  
        a1=1;  
  
    } catch (Exception e) {  
  
        JOptionPane.showMessageDialog(this, "Please Enter a Valid Date in proper  
format\nThat is : YYYY-MM-DD\nAnd all the details properly");  
  
        a1=0;  
  
    }  
  
    if(a1!=0)  
  
    {  
  
        try  
  
        {  
  
            CardNo = Long.parseLong(CardTF.getText());  
  
            Phone = Long.parseLong(PhoneTF.getText());  
  
            long b=Long.parseLong(Balance);  
  
            a=1;
```

```

    }

    catch(Exception e)

    {

        JOptionPane.showMessageDialog(this,"Please enter all details properly");

        a=0;

    }

if(NameAcc.trim().equals("") || AccNo.trim().equals("") || Balance.trim().equals("") || DoJ.trim().equals(""))

{

    if(a==1)

    {

        JOptionPane.showMessageDialog(this,"Please enter all details properly");

    }

}

else{

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        if (CurrentRB.isSelected())

        {

            Acctype="Current Account";

        }

        else if(SavingRB.isSelected())

        {

```



```

        Acctype="Saving Account";
    }

    if (Phone<1000000000 || Phone>100000000000L) {

        JOptionPane.showMessageDialog(this, "Please Enter a Valid Phone No");

        return;
    }

    if (Integer.parseInt(Balance)<500) {

        JOptionPane.showMessageDialog(this, "Please Create Account with more than Rs. 500");

        return;
    }

    int rows=stmt.executeUpdate("insert into
Bank_Table(Account_no,Name_of_Account_Holder,Balance,Card_no,Phone_no,Date_of_Account_Creation,Account_Type)
values('"+(AccNo)+"','"+(NameAcc)+"','"+(Balance)+"','"+(CardNo)+"','"+(Phone)+"','"+(DoJ)+"','"+(Acctype)+"'");

    JOptionPane.showMessageDialog(this, "Account Created");

    int nac=Integer.parseInt(AccNo)-1;

    int Ac=Integer.parseInt(AccNo);

    int ac=Ac+1;

    int rowss=stmt.executeUpdate("update Accno set Acc='"+(AccNo)+"' where Acc='"+(nac)+"'");

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, e.getMessage());

}

}

}

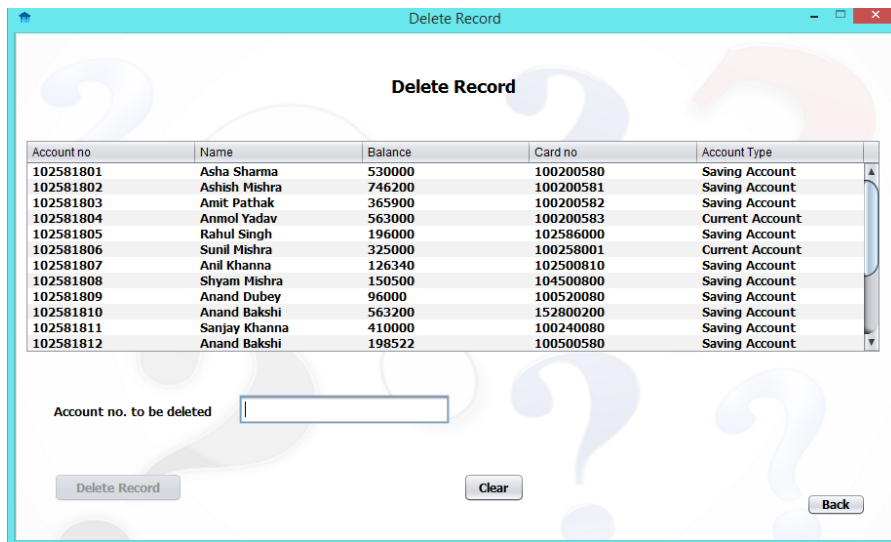
}

}

```

```
private void BackBTNActionPerformed(java.awt.event.ActionEvent evt) {  
    new Bank_management_system().setVisible(true);  
    this.setVisible(false);  
}  
  
private void ClearBTNActionPerformed(java.awt.event.ActionEvent evt) {  
    BalanceTF.setText("");  
    CardTF.setText("");  
    DoJTF.setText("");  
    NameTF.setText("");  
    PhoneTF.setText("");  
    CurrentRB.setSelected(true);  
    SavingRB.setSelected(false);  
}
```

On selecting Delete Account option in the main frame:



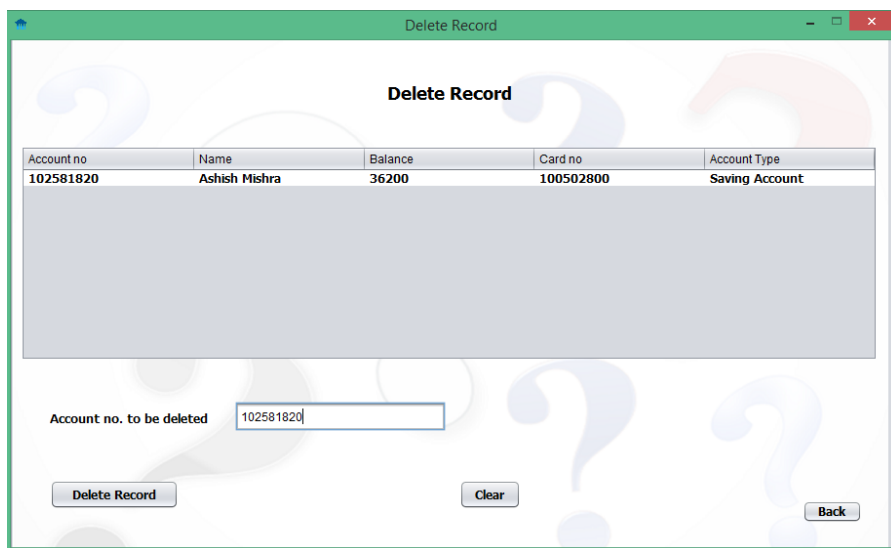
The screenshot shows a window titled "Delete Record" with a light blue border. Inside, there's a table with 5 columns: Account no, Name, Balance, Card no, and Account Type. The table lists 12 accounts. Below the table is a text input field labeled "Account no. to be deleted". At the bottom, there are three buttons: "Delete Record", "Clear", and "Back".

| Account no | Name | Balance | Card no | Account Type |
|------------|---------------|---------|-----------|-----------------|
| 102581801 | Asha Sharma | 530000 | 100200580 | Saving Account |
| 102581802 | Ashish Mishra | 746200 | 100200581 | Saving Account |
| 102581803 | Amit Pathak | 365900 | 100200582 | Saving Account |
| 102581804 | Anmol Yadav | 563000 | 100200583 | Current Account |
| 102581805 | Rahul Singh | 196000 | 102586000 | Saving Account |
| 102581806 | Sunil Mishra | 325000 | 100258001 | Current Account |
| 102581807 | Anil Khanna | 126340 | 102500810 | Saving Account |
| 102581808 | Shyam Mishra | 150500 | 104500800 | Saving Account |
| 102581809 | Anand Dubey | 96000 | 100520080 | Saving Account |
| 102581810 | Anand Bakshi | 563200 | 152800200 | Saving Account |
| 102581811 | Sanjay Khanna | 410000 | 100240080 | Saving Account |
| 102581812 | Anand Bakshi | 198522 | 100500580 | Saving Account |

Account no. to be deleted:

Buttons: Delete Record, Clear, Back

On giving proper value:

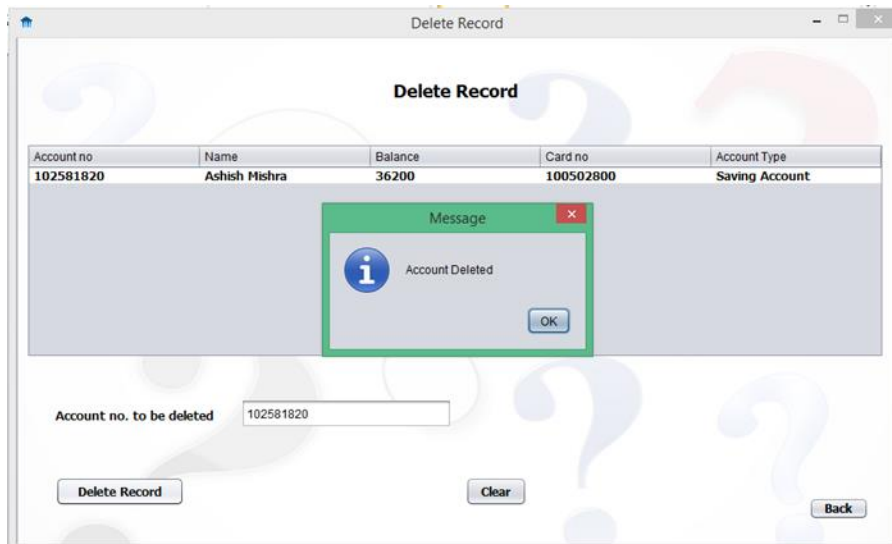


The screenshot shows the same "Delete Record" window, but with the account number "102581820" entered in the text field. The table now only shows one record for Ashish Mishra. The rest of the interface is the same.

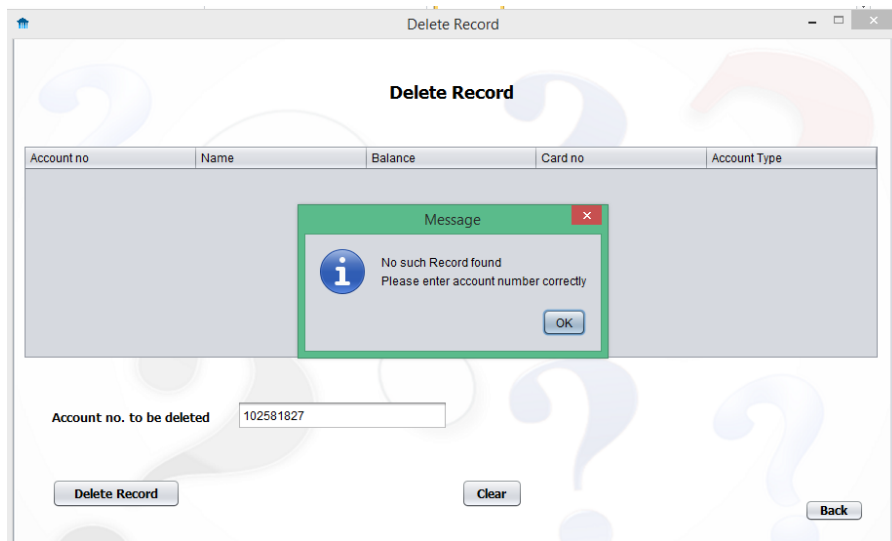
| Account no | Name | Balance | Card no | Account Type |
|------------|---------------|---------|-----------|----------------|
| 102581820 | Ashish Mishra | 36200 | 100502800 | Saving Account |

Account no. to be deleted:

Buttons: Delete Record, Clear, Back



On enetering wrong value:



Import statements:

```
import java.awt.event.KeyEvent;  
  
import java.net.URL;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
  
import java.sql.ResultSet;  
  
import java.sql.Statement;  
  
import javax.swing.ImageIcon;  
  
import javax.swing.JOptionPane;  
  
import javax.swing.table.DefaultTableModel;
```

Constructor and global variables:

```
int r=0,ctr=0;  
  
String pwd="Jns";  
  
public Delete_Acccount()  
{  
  
    initComponents();  
  
    this.setSize(910,548);  
  
    this.setLocation(220, 92);  
  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
  
    ImageIcon icon = new ImageIcon(resource);  
  
    this.setIconImage(icon.getImage());  
  
    ctd("select * from Bank_Table");  
  
    DeleteBTN.setEnabled(false);  
  
}
```

Methods:

```

public void ctd(String qr)
{
DefaultTableModel tm=(DefaultTableModel)jTable1.getModel();

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        if(r==1)

        {

            int rows1=stmt.executeUpdate(qr);

            JOptionPane.showMessageDialog(this, "Account Deleted");

            qr="select * from Bank_Table";

            DeleteBTN.setEnabled(false);

        }

        ResultSet rs=stmt.executeQuery(qr);

        int rows=tm.getRowCount();

        if (rows>0){

            for (int i = 0; i < rows; i++) {

                tm.removeRow(0);

            }

        }

        while (rs.next()) {

            String acc=rs.getString("Account_no");

            String name=rs.getString("Name_of_Account_Holder");

```

```

        String balance=rs.getString("Balance");

        String card=rs.getString("Card_no");

        String Acctype=rs.getString("Account_Type");

        tm.addRow(new Object[]{acc,name,balance,card,Acctype});

    }

    if(tm.getRowCount()==0)

    {

        JOptionPane.showMessageDialog(this, "No such Record found\nPlease enter account number
correctly");

        AccTF.setText("");

        ctd("select * from Bank_Table");

        DeleteBTN.setEnabled(false);

    }

    else

    {

        DeleteBTN.setEnabled(true);

    }

}

catch (Exception e) {

    JOptionPane.showMessageDialog(this, e.getMessage());

}

}

```

Codes:

```
private void DeleteBTNActionPerformed(java.awt.event.ActionEvent evt) {  
  
    r=1;  
  
    String Acc=AccTF.getText();  
  
    if(Acc.trim().equals("")){JOptionPane.showMessageDialog(this, "Please enter Account number");}  
  
    else  
  
    {  
  
        Acc="delete from Bank_Table where Account_no="+Acc+"";  
  
        ctd(Acc);  
  
    }  
DeleteBTN.setEnabled(false);  
  
}  
  
private void BackBTNActionPerformed(java.awt.event.ActionEvent evt) {  
  
    new Bank_management_system().setVisible(true);  
  
    this.setVisible(false); // exit the frame  
  
}  
  
private void formWindowActivated(java.awt.event.WindowEvent evt) {  
  
}  
  
private void AccTFKeyPressed(java.awt.event.KeyEvent evt) {  
  
r=2;  
  
if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {  
  
    String Acc=AccTF.getText();  
  
if(Acc.trim().equals("")){JOptionPane.showMessageDialog(this, "Please enter Account number");}  
  
else  
  
{
```



```
Acc="select * from Bank_Table where Account_no="+Acc+"";
ctd(Acc);
}
}
}

private void ClearBTNActionPerformed(java.awt.event.ActionEvent evt) {
r=3;
AccTF.setText("");
String d="select * from Bank_Table";
ctd(d);
DeleteBTN.setEnabled(false);
}
```

On selecting Modify Account option in the main frame:

Modifying the Record

| Account no | Name | Balance | Account Type |
|------------|---------------|---------|-----------------|
| 102581801 | Asha Sharma | 530000 | Saving Account |
| 102581802 | Ashish Mishra | 746200 | Saving Account |
| 102581803 | Amit Pathak | 365900 | Saving Account |
| 102581804 | Anmol Yadav | 563000 | Current Account |
| 102581805 | Rahul Singh | 196000 | Saving Account |
| 102581806 | Sunil Mishra | 325000 | Current Account |
| 102581807 | Anil Khanna | 126340 | Saving Account |
| 102581808 | Shyam Mishra | 150500 | Saving Account |

Search By:

☒ Search by Account no

OR

☐ Search by Name

Name

Account no

Change Balance:

Modifying the Record

| Account no | Name | Balance | Account Type |
|------------|---------------|---------|-----------------|
| 102581801 | Asha Sharma | 530000 | Saving Account |
| 102581802 | Ashish Mishra | 746200 | Saving Account |
| 102581803 | Amit Pathak | 365900 | Saving Account |
| 102581804 | Anmol Yadav | 563000 | Current Account |
| 102581805 | Rahul Singh | 196000 | Saving Account |
| 102581806 | Sunil Mishra | 325000 | Current Account |
| 102581807 | Anil Khanna | 126340 | Saving Account |
| 102581808 | Shyam Mishra | 150500 | Saving Account |

Search By:

☐ Search by Account no

OR

☒ Search by Name

Name

Account no

Change Balance:

Modify Account

Modifying the Record

| Account no | Name | Balance | Account Type |
|------------|-------------|---------|----------------|
| 102581801 | Asha Sharma | 530000 | Saving Account |

Search By:

☐ Search by Account no

OR

☒ Search by Name

Name

Account no

Change Balance:

Modification occurs:

Modify Account

Modifying the Record

| Account no | Name | Balance | Account Type |
|------------|-------------|---------|----------------|
| 102581801 | Asha Sharma | 600000 | Saving Account |

Search By:

☐ Search by Account no

OR

☒ Search by Name

Name

Account no

Change Balance:

Note : Here we can change the balance only. For changing other details we click on Change other details.

Import statements:

```
import java.awt.event.KeyEvent;  
  
import java.net.URL;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
  
import java.sql.ResultSet;  
  
import java.sql.Statement;  
  
import javax.swing.ImageIcon;  
  
import javax.swing.JOptionPane;  
  
import javax.swing.table.DefaultTableModel;
```

Constructor and global variables:

```
String pwd="Jns";  
  
public Modify_Account() {  
  
    initComponents();  
  
    this.setSize(1024,665);  
  
    this.setLocation(160,50);  
  
    AccNoTF.setFocusable(false);  
  
    NameTF.setFocusable(false);  
  
    cty();  
  
    jRadioButton2.setSelected(true);  
  
    AccTF.setEnabled(true);  
  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
  
    ImageIcon icon = new ImageIcon(resource);  
  
    this.setIconImage(icon.getImage());  
  
}
```

Methods:

```
public void cta(String qry)
{
    DefaultTableModel tm=(DefaultTableModel)ModifyTBL.getModel();

    try {

        int rows=tm.getRowCount();

        if (rows>0)

        {

            for (int i = 0; i < rows; i++)

            {

                tm.removeRow(0);

            }

        }

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        ResultSet rs=stmt.executeQuery(qry);

        while (rs.next()) {

            String acc=rs.getString("Account_no");

            String name=rs.getString("Name_of_Account_Holder");

            String balance=rs.getString("Balance");

            String card=rs.getString("Card_no");

            String Acctype=rs.getString("Account_Type");

            tm.addRow(new Object[]{acc,name,balance,Acctype});
```

```

        AccNoTF.setText(acc);

        NameTF.setText(name);

    }

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, e.getMessage());

}

EnterNameTF.setText("");

if(tm.getRowCount()==0)

{

    JOptionPane.showMessageDialog(this, "No Such Record exists\nPlease enter details correctly ");

    cty();

    AccTF.setText("");

}

}

public void cty()

{

    DefaultTableModel tm=(DefaultTableModel)ModifyTBL.getModel();

    try

    {

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        ResultSet rs=stmt.executeQuery("select * from Bank_Table");

```

```
int rows=tm.getRowCount();

if (rows>0)

{

    for (int i = 0; i < rows; i++)

    {

        tm.removeRow(0);

    }

}

while (rs.next())

{

    String acc=rs.getString("Account_no");

    String name=rs.getString("Name_of_Account_Holder");

    String balance=rs.getString("Balance");

    String card=rs.getString("Card_no");

    String Acctype=rs.getString("Account_Type");

    tm.addRow(new Object[]{acc,name,balance,Acctype});

}

}

catch (Exception e)

{

    JOptionPane.showMessageDialog(this, e.getMessage());

}

}
```

Codes;

```
private void UpdateBTNActionPerformed(java.awt.event.ActionEvent evt) {  
DefaultTableModel tm=(DefaultTableModel)ModifyTBL.getModel();  
  
String Balance=BalanceTF.getText();  
  
String Acc=AccNoTF.getText();  
  
try {  
  
    Class.forName("com.mysql.jdbc.Driver");  
  
    Connection con=(Connection)  
  
    DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);  
  
    Statement stmt=con.createStatement();  
  
    ResultSet bal=stmt.executeQuery("select * from Bank_Table where Account_no='"+(Acc)+"'");  
  
    String amount="";  
  
    while (bal.next()) {  
  
        amount=bal.getString("Balance");  
  
    }  
  
    if (Integer.parseInt(amount)<500&&Integer.parseInt(Balance)>500) {  
  
        JOptionPane.showMessageDialog(this, "A Fine of Rs. 50 will be Charged");  
  
        Balance=Integer.toString(Integer.parseInt(Balance)-50);  
  
    }  
  
    int rows=stmt.executeUpdate("update Bank_Table set Balance='"+(Balance)+"' where  
Account_no='"+(Acc)+"'");  
  
    ResultSet rs=stmt.executeQuery("select * from Bank_Table where Account_no='"+(Acc)+"'");  
  
    if (rows>0) {  
  
        for (int i = 0; i < rows; i++) {  
  
            tm.removeRow(0);  
  
        }  
    }  
}
```



```

while (rs.next()) {

    String acc=rs.getString("Account_no");

    String name=rs.getString("Name_of_Account_Holder");

    String balance=rs.getString("Balance");

    String card=rs.getString("Card_no");

    String Acctype=rs.getString("Account_Type");

    tm.addRow(new Object[]{acc,name,balance,Acctype});

}

} catch (Exception e) {

    JOptionPane.showMessageDialog(this,"Plaese enter details properly");

}

}

private void BackBTNActionPerformed(java.awt.event.ActionEvent evt) {

    new Bank_management_system().setVisible(true);

    this.setVisible(false); // exit the frame

}

private void AccTFKeyPressed(java.awt.event.KeyEvent evt) {

String Acc=AccTF.getText();

Acc="select * from Bank_Table where Account_no='"+(Acc)+"'";

    if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {

        cta(Acc);

    }

}

}

private void DetailsBTNActionPerformed(java.awt.event.ActionEvent evt) {

```

```

new Admin_Login().setVisible(true);

this.setVisible(false);

}

private void ClearBTNActionPerformed(java.awt.event.ActionEvent evt) {

    AccTF.setText("");

    AccNoTF.setText("");

    NameTF.setText("");

    BalanceTF.setText("");

    cty();

}

private void EnterNameTFKeyPressed(java.awt.event.KeyEvent evt) {

String Name=EnterNameTF.getText();

Name="select * from Bank_Table where Name_of_Account_Holder='"+(Name)+"'";

if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {

    cta(Name);

}

}

private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    EnterNameTF.setEnabled(false);

    AccTF.setEnabled(true);

    EnterNameTF.setText(""); }

private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {

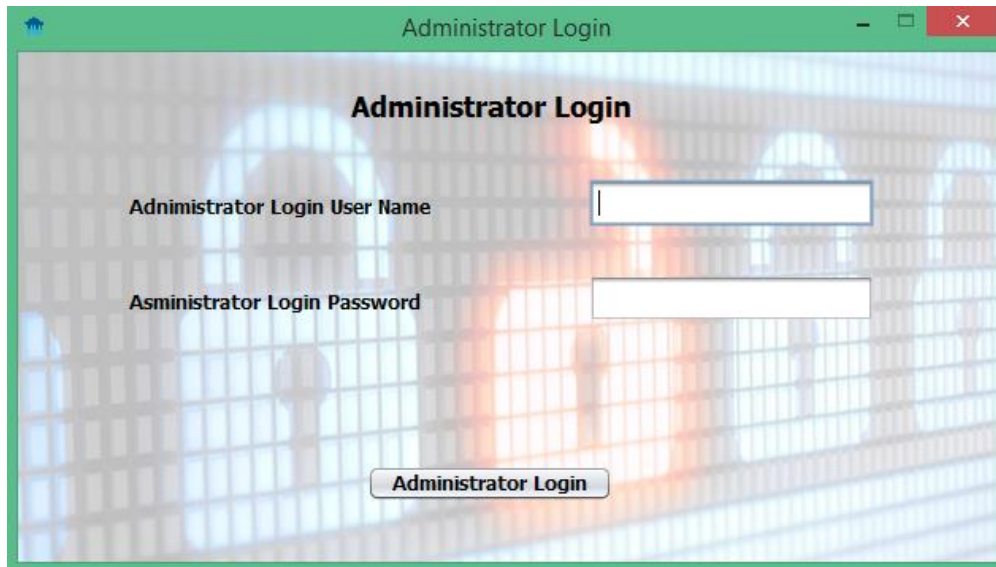
    AccTF.setEnabled(false);

    EnterNameTF.setEnabled(true);

    AccTF.setText(""); }

```

On selecting 'Change Other Details' option in the previous frame:

A screenshot of a web application window titled "Administrator Login". The window has a green title bar with standard Windows controls (minimize, maximize, close). The main content area has a background image of a person's face behind a grid. The text "Administrator Login" is centered at the top. Below it, there are two input fields: "Administrator Login User Name" and "Administrator Login Password". A button labeled "Administrator Login" is positioned below the password field.

Administrator Login

Administrator Login User Name

Administrator Login Password

Administrator Login

On adding user name as: 'bank' and password as '12' we are directed to the Change details frame.

In this frame we can change other details like Name, card no. , phone no. etc. which we can't change in Modify frame.

Impot statements:

```
import java.awt.event.KeyEvent;
```

```
import java.net.URL;
```

```
import javax.swing.ImageIcon;
```

```
import javax.swing.JOptionPane;
```

Constructor and global variables;

```
public Admin_Login() {  
    initComponents();  
    this.setSize(623,350);  
    this.setLocation(340,200);  
    URL resource = Login.class.getResource("Images/institution_icon.png");  
    ImageIcon icon = new ImageIcon(resource);  
    this.setIconImage(icon.getImage());  
}
```

Codes :

```
private void LoginBTNActionPerformed(java.awt.event.ActionEvent evt) {  
    if ("bank".equals(NameTF.getText().trim()) && "12".equals(new  
String(PassPF.getPassword()).trim())) {  
        new Change_Details().setVisible(true);  
        this.setVisible(false);  
    }else{  
        JOptionPane.showMessageDialog(this, "You do not Have Administrator Authorities");  
        new Modify_Account().setVisible(true);  
        this.setVisible(false);  
    }  
}
```

```

private void LoginBTNKeyPressed(java.awt.event.KeyEvent evt) {
    if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {
        if ("bank".equals(NameTF.getText().trim()) && "12".equals(new
String(PassPF.getPassword()).trim())) {
            new Change_Details().setVisible(true);
            this.setVisible(false);
        }else{
            JOptionPane.showMessageDialog(this, "You do not Have Administrator Authorities");
            new Modify_Account().setVisible(true);
            this.setVisible(false);
        }
    }
}

private void PassPFKeyPressed(java.awt.event.KeyEvent evt) {
    if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {
        if ("bank".equals(NameTF.getText().trim()) && "12".equals(new
String(PassPF.getPassword()).trim())) {
            new Change_Details().setVisible(true);
            this.setVisible(false);
        }else{
            JOptionPane.showMessageDialog(this, "You do not Have Administrator Authorities");
            new Modify_Account().setVisible(true);
            this.setVisible(false);
        }
    }
}

```

This is our change details frame where we can change the personal details of Account holder by entering his/her Account number.

Change Details

| Account no. | Name of Account holder | Balance | Card no. | Phone no. | Date of Account Creation | Account Type |
|-------------|------------------------|---------|-----------|------------|--------------------------|-----------------|
| 102581801 | Asha Sharma | 600000 | 100200580 | 9632758741 | 2010-09-04 | Saving Account |
| 102581802 | Ashish Mishra | 746200 | 100200581 | 9872456328 | 2011-07-04 | Saving Account |
| 102581803 | Amit Pathak | 365900 | 100200582 | 8745981264 | 2006-04-07 | Saving Account |
| 102581804 | Anmol Yadav | 563000 | 100200583 | 9124753681 | 2012-08-11 | Current Account |
| 102581805 | Rahul Singh | 196000 | 102586000 | 8723419853 | 2012-04-16 | Saving Account |
| 102581806 | Sunil Mishra | 325000 | 100258001 | 9172283671 | 2007-01-25 | Current Account |
| 102581807 | Anil Khanna | 126340 | 102500810 | 8126677291 | 2014-08-26 | Saving Account |
| 102581808 | Shyam Mishra | 150500 | 104500800 | 8845112769 | 1999-05-13 | Saving Account |
| 102581809 | Anand Dubey | 96000 | 100520080 | 8477521198 | 2013-04-07 | Saving Account |
| 102581810 | Anand Bakshi | 563200 | 152800200 | 9996135718 | 2013-04-07 | Saving Account |
| 102581811 | Sanjay Khanna | 410000 | 100240080 | 9912745581 | 2013-04-08 | Saving Account |
| 102581812 | Anand Bakshi | 198522 | 100500580 | 9877413528 | 2010-05-17 | Saving Account |
| 102581813 | Sapan Dubey | 130299 | 100250081 | 8884124793 | 2009-02-24 | Current Account |
| 102581814 | Alok Sharma | 23500 | 102058100 | 9841225748 | 2010-05-21 | Current Account |
| 102581815 | Amit Khanna | 5995 | 100205081 | 9871245781 | 2013-05-01 | Saving Account |

Enter Account Number:

Name

Change Name

Card No

Change Card No

Phone No +91

Change Phone No

Account TypeCurrent Account

Change Account Type

Date

Change Date

Clear

Exit

Back

On entering proper account no. all the fields along with the clear button gets enabled.

Change Details

| Account no. | Name of Account holder | Balance | Card no. | Phone no. | Date of Account Creation | Account Type |
|-------------|------------------------|---------|-----------|------------|--------------------------|----------------|
| 102581821 | Yash | 120000 | 120981191 | 1273467890 | 2015-01-02 | Saving Account |

Enter Account Number:

Name

Change Name

Card No

Change Card No

Phone No +91

Change Phone No

Account TypeCurrent Account

Change Account Type

Date

Change Date

Clear

Exit

Back

On entering card no. and clicking change card no. , the card no. changes :

Change Details

| Account no. | Name of Account holder | Balance | Card no. | Phone no. | Date of Account Creation | Account Type |
|-------------|------------------------|---------|-----------|------------|--------------------------|----------------|
| 102581821 | Yash | 120000 | 122256788 | 1273467890 | 2015-01-02 | Saving Account |

Enter Account Number:

102581821

Name

Change Name

Card No

122256788

Change Card No

Phone No

+91

Change Phone No

Account Type

Current Account

Change Account Type

Date

Change Date

Clear

Exit

Back

Similarly other details can be changed.

On entering improper Account no. a message is raised :

The screenshot shows the 'Change Details' application window. At the top, there is a table with headers: 'Account no.', 'Name of Account holder', 'Balance', 'Card no.', 'Phone no.', 'Date of Account Creation', and 'Account Type'. Below this table is a large grey rectangular area. In the center, a 'Message' dialog box is displayed with a green border and a red close button. The dialog contains an information icon and the text 'Please enter a valid Account Number', with an 'OK' button at the bottom right. To the left of the dialog, the 'Enter Account Number:' label is followed by a text input field containing '102581820'. Below this, there is a form with labels and input fields for 'Name', 'Card No', 'Phone No' (with a '+91' prefix), 'Account Type' (a dropdown menu showing 'Current Account'), and 'Date'. To the right of each input field is a corresponding 'Change' button: 'Change Name', 'Change Card No', 'Change Phone No', 'Change Account Type', and 'Change Date'. At the bottom right of the application window are three buttons: 'Clear', 'Exit', and 'Back'.

Import statements;

```
import java.awt.Color;

import java.awt.event.KeyEvent;

import java.net.URL;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;
```

Constructor and global variables;

```
int r=0;

String pwd="Jns";

int eq=0;

    public Change_Details() {

        initComponents();

        this.setSize(1290,740);

        URL resource = Login.class.getResource("Images/institution_icon.png");

        ImageIcon icon = new ImageIcon(resource);

        this.setIconImage(icon.getImage());

        ctd("select * from Bank_Table");

    }
```

Methods:

```
public void ctd(String qry)
{
    DefaultTableModel tm=(DefaultTableModel)DetailsTBL.getModel();

    try {

        Class.forName("com.mysql.jdbc.Driver");

        Connection con=(Connection)

        DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

        Statement stmt=con.createStatement();

        ResultSet rs=stmt.executeQuery(qry);

        int rows=tm.getRowCount();

        if (rows>0) {

            for (int i = 0; i < rows; i++) {

                tm.removeRow(0);

            }}

        while (rs.next()) {

            String acc=rs.getString("Account_no");

            String name=rs.getString("Name_of_Account_Holder");

            String balance=rs.getString("Balance");

            String card=rs.getString("Card_no");

            String Acctype=rs.getString("Account_Type");

            String date=rs.getString("Date_of_Account_creation");

            String phone=rs.getString("Phone_no");

            tm.addRow(new Object[]{acc,name,balance,card,phone,date,Acctype});

        }

    }
```

```

eq=1;

if(tm.getRowCount()==0)

{

JOptionPane.showMessageDialog(this,"Please enter a valid Account Number");

eq=0;

AccTF.setText("");

ctd("select * from Bank_Table");

}

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, e.getMessage());

}

}

public void ctup(String qr1,String qr2)

{

try {

    DefaultTableModel tm=(DefaultTableModel)DetailsTBL.getModel();

    Class.forName("com.mysql.jdbc.Driver");

    Connection con=(Connection)

    DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

    Statement stmt=con.createStatement();

    int rows=stmt.executeUpdate(qr1);

    ResultSet rs=stmt.executeQuery(qr2);

    int rowss=tm.getRowCount();

    if (rowss>0) {

        for (int i = 0; i < rowss; i++) {

```

```

        tm.removeRow(0);

    }}

    while (rs.next()) {

        String acc=rs.getString("Account_no");

        String name=rs.getString("Name_of_Account_Holder");

        String balance=rs.getString("Balance");

        String card=rs.getString("Card_no");

        String Acctype=rs.getString("Account_Type");

        String date=rs.getString("Date_of_Account_creation");

        String phone=rs.getString("Phone_no");

        tm.addRow(new Object[]{acc,name,balance,card,phone,date,Acctype});

        NameBTN.setEnabled(true);

    }

} catch (Exception e){

    JOptionPane.showMessageDialog(this, e.getMessage());

}

}

```

Codes:

```

private void BackBTNActionPerformed(java.awt.event.ActionEvent evt) {

    new Modify_Account().setVisible(true);

    this.setVisible(false);

}

private void AccTFKeyPressed(java.awt.event.KeyEvent evt) {

if ("Enter".equals(KeyEvent.getKeyText(evt.getKeyCode())) {

    int z=0;

```

```
long Acc1=0;

try

{

Acc1=Long.parseLong(AccTF.getText());

z=1;

}

catch(Exception e)

{

    JOptionPane.showMessageDialog(this,"Please enter Account number in proper format");

    AccTF.setText("");

}

if(z==1){

String Acc="" +Acc1;

Acc="select * from Bank_Table where Account_no="+Acc+"";

ctd(Acc);

if(AccTF.getText().trim().equals(""))

{

    NameBTN.setEnabled(false);

    AccTypeBTN.setEnabled(false);

    CardBTN.setEnabled(false);

    ClearBTN.setEnabled(false);

    DateBTN.setEnabled(false);

    DateTF.setEnabled(false);

    NameTF.setEnabled(false);

    PhoneBTN.setEnabled(false);

}
```

```
PhoneTF.setEnabled(false);

TypeCoB.setEnabled(false);

CardTF.setEnabled(false);
}

else{
if(eq==1)
{
NameBTN.setEnabled(true);

AccTypeBTN.setEnabled(true);

CardBTN.setEnabled(true);

ClearBTN.setEnabled(true);

DateBTN.setEnabled(true);

DateTF.setEnabled(true);

NameTF.setEnabled(true);

PhoneBTN.setEnabled(true);

PhoneTF.setEnabled(true);

TypeCoB.setEnabled(true);

CardTF.setEnabled(true);

eq=0;
}}

}

}

}

private void NameBTNActionPerformed(java.awt.event.ActionEvent evt) {

String Acc=AccTF.getText();
```

```

String Name=NameTF.getText();

if(Acc.trim().equals(""))
{
JOptionPane.showMessageDialog(this,"Please enter Account number");
}

else if(Name.trim().equals(""))
{
JOptionPane.showMessageDialog(this,"Please enter a Name");
}

else
{
String s1="update bank_table set Name_of_Account_Holder='"+(Name)+"' where
Account_no='"+(Acc)+"'";

String s2="select * from Bank_Table where Account_no='"+(Acc)+"'";

ctup(s1,s2);
}

}

private void CardBTNActionPerformed(java.awt.event.ActionEvent evt) {

String Acc=AccTF.getText();

String Card=CardTF.getText();

if(Acc.trim().equals(""))
{
JOptionPane.showMessageDialog(this,"Please enter Account number");
}

else if(Card.trim().equals(""))
{

```

```

JOptionPane.showMessageDialog(this,"Please enter a Card number");
}
else
{
    String s1="update bank_table set Card_no='"+(Card)+"' where Account_no='"+(Acc)+"'";
    String s2="select * from Bank_Table where Account_no='"+(Acc)+"'";
    ctup(s1,s2);
}
}

private void PhoneBTNActionPerformed(java.awt.event.ActionEvent evt) {
String Acc=AccTF.getText();

long Phone=0L;

int xm=0;

try{
    Phone=Long.parseLong(PhoneTF.getText());

    xm=1;
}
catch(Exception e)
{
    JOptionPane.showMessageDialog(this, "Please enter Phone number properly");

    PhoneTF.setText("");
}

if(xm==1){
    if (Phone<1000000000 || Phone>10000000000L)
    {

```



```

        JOptionPane.showMessageDialog(this, "Please Enter a Valid Phone No");

        PhoneTF.setText("");
    }
else
{
    String s1="update bank_table set Phone_no='"+(Phone)+"' where Account_no='"+(Acc)+"'";

    String s2="select * from Bank_Table where Account_no='"+(Acc)+"'";

    ctup(s1,s2);
}
}
}

private void DateBTNActionPerformed(java.awt.event.ActionEvent evt) {
int w=0;

String Acc=AccTF.getText();

String Date=DateTF.getText();

if(Acc.trim().equals(""))
{
JOptionPane.showMessageDialog(this,"Please enter Account number");
}

else if(Date.trim().equals(""))
{
JOptionPane.showMessageDialog(this,"Please enter a Date");
}

else
{

```

```

SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd");

try {

    Date Acda=df.parse(Date);

    w=1;

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, "Please Enter a Valid Date in proper format\nThat is :
YYYY-MM-DD");

    DateTF.setText("");

}

if(w==1){

    String s1="update bank_table set Date_of_Account_Creation='"+(Date)+"' where
Account_no='"+(Acc)+"'";

    String s2="select * from Bank_Table where Account_no='"+(Acc)+"'";

    ctup(s1,s2);

}

}

}

private void AccTypeBTNActionPerformed(java.awt.event.ActionEvent evt) {

String Acc=AccTF.getText();

String Type;

    if ("Current Account".equals(TypeCoB.getSelectedItem().toString())) {

        Type="Current Account";

    }else{

        Type="Saving Account";

    }

if(Acc.trim().equals(""))

```

```

{
JOptionPane.showMessageDialog(this,"Please enter Account number");
}
else
{
    String s1="update bank_table set Account_Type='"+(Type)+"' where Account_no='"+(Acc)+"'";
    String s2="select * from Bank_Table where Account_no='"+(Acc)+"'";
    ctup(s1,s2);
}
}

private void ClearBTNActionPerformed(java.awt.event.ActionEvent evt) {
AccTF.setText("");
CardTF.setText("");
DateTF.setText("");
NameTF.setText("");
PhoneTF.setText("");
TypeCoB.setSelectedIndex(0);
ctd("select * from Bank_Table");
NameBTN.setEnabled(false);
    AccTypeBTN.setEnabled(false);
    CardBTN.setEnabled(false);
    ClearBTN.setEnabled(false);
    DateBTN.setEnabled(false);
    DateTF.setEnabled(false);
    NameTF.setEnabled(false);

```

```
    PhoneBTN.setEnabled(false);

    PhoneTF.setEnabled(false);

    TypeCoB.setEnabled(false);

    CardTF.setEnabled(false);
}

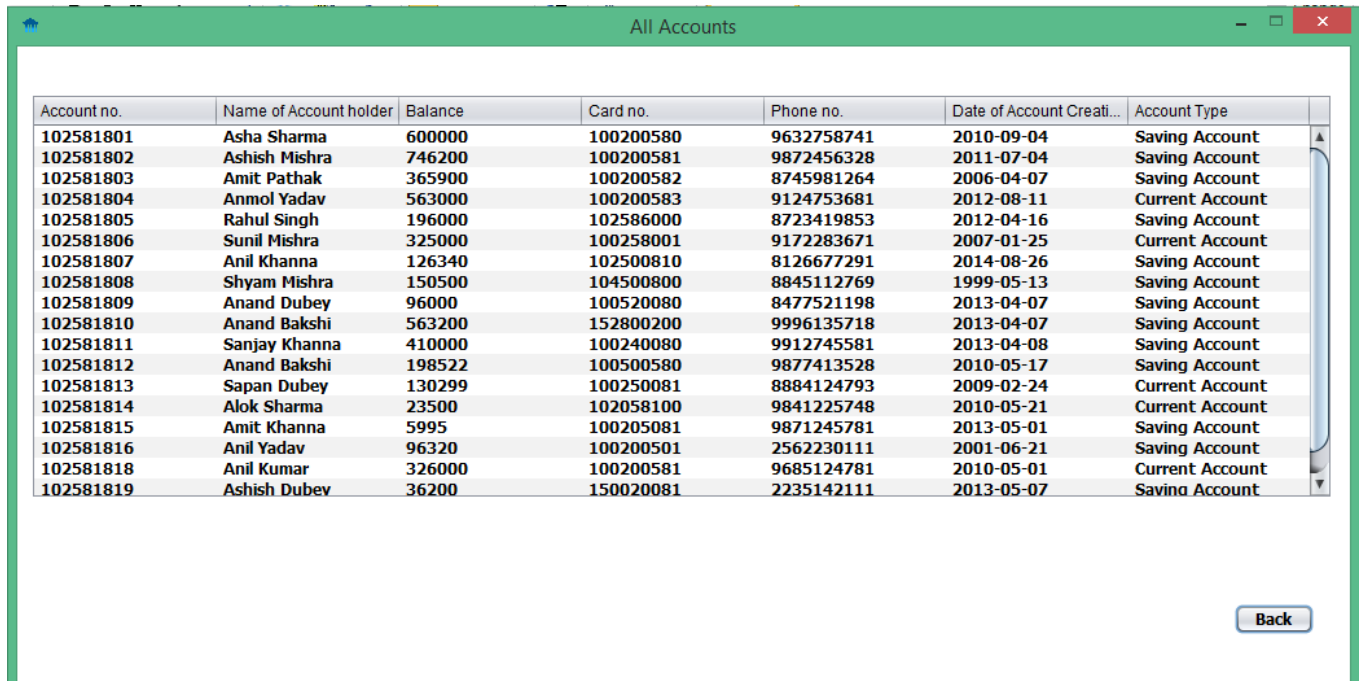
private void CardTFActionPerformed(java.awt.event.ActionEvent evt) {

}

private void ExitActionPerformed(java.awt.event.ActionEvent evt) {
System.exit(0);

}
```

On selecting Show All Accounts option in the main frame the following frame is displayed :



| Account no. | Name of Account holder | Balance | Card no. | Phone no. | Date of Account Creation | Account Type |
|-------------|------------------------|---------|-----------|------------|--------------------------|-----------------|
| 102581801 | Asha Sharma | 600000 | 100200580 | 9632758741 | 2010-09-04 | Saving Account |
| 102581802 | Ashish Mishra | 746200 | 100200581 | 9872456328 | 2011-07-04 | Saving Account |
| 102581803 | Amit Pathak | 365900 | 100200582 | 8745981264 | 2006-04-07 | Saving Account |
| 102581804 | Anmol Yadav | 563000 | 100200583 | 9124753681 | 2012-08-11 | Current Account |
| 102581805 | Rahul Singh | 196000 | 102586000 | 8723419853 | 2012-04-16 | Saving Account |
| 102581806 | Sunil Mishra | 325000 | 100258001 | 9172283671 | 2007-01-25 | Current Account |
| 102581807 | Anil Khanna | 126340 | 102500810 | 8126677291 | 2014-08-26 | Saving Account |
| 102581808 | Shyam Mishra | 150500 | 104500800 | 8845112769 | 1999-05-13 | Saving Account |
| 102581809 | Anand Dubey | 96000 | 100520080 | 8477521198 | 2013-04-07 | Saving Account |
| 102581810 | Anand Bakshi | 563200 | 152800200 | 9996135718 | 2013-04-07 | Saving Account |
| 102581811 | Sanjay Khanna | 410000 | 100240080 | 9912745581 | 2013-04-08 | Saving Account |
| 102581812 | Anand Bakshi | 198522 | 100500580 | 9877413528 | 2010-05-17 | Saving Account |
| 102581813 | Sapan Dubey | 130299 | 100250081 | 8884124793 | 2009-02-24 | Current Account |
| 102581814 | Alok Sharma | 23500 | 102058100 | 9841225748 | 2010-05-21 | Current Account |
| 102581815 | Amit Khanna | 5995 | 100205081 | 9871245781 | 2013-05-01 | Saving Account |
| 102581816 | Anil Yadav | 96320 | 100200501 | 2562230111 | 2001-06-21 | Saving Account |
| 102581818 | Anil Kumar | 326000 | 100200581 | 9685124781 | 2010-05-01 | Current Account |
| 102581819 | Ashish Dubey | 36200 | 150020081 | 2235142111 | 2013-05-07 | Saving Account |

Back

Import statements:

```
import java.net.URL;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;
```

Constructor and global variables:

```
String pwd="Jns";

    public Show_all_Account() {

        initComponents();

        this.setSize(1035,520);

        this.setLocation(120,100);

        URL resource = Login.class.getResource("Images/institution_icon.png");

        ImageIcon icon = new ImageIcon(resource);

        this.setIconImage(icon.getImage());

    }
```

Codes:

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

new Bank_management_system().setVisible(true);

        this.setVisible(false);

    }

    private void formWindowActivated(java.awt.event.WindowEvent evt) {
```

```

DefaultTableModel tm=(DefaultTableModel)ShowTBL.getModel();

try {

    Class.forName("com.mysql.jdbc.Driver");

    Connection con=(Connection)

    DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",pwd);

    Statement stmt=con.createStatement();

    ResultSet rs=stmt.executeQuery("select * from Bank_Table");

    int rows=tm.getRowCount();

    if (rows>0) {

        for (int i = 0; i < rows; i++) {

            tm.removeRow(0);

        }

    }

    while (rs.next()) {

        String acc=rs.getString("Account_no");

        String name=rs.getString("Name_of_Account_Holder");

        String balance=rs.getString("Balance");

        String card=rs.getString("Card_no");

        String Acctype=rs.getString("Account_Type");

        String date=rs.getString("Date_of_Account_creation");

        String phone=rs.getString("Phone_no");

        tm.addRow(new Object[]{acc,name,balance,card,phone,date,Acctype});

    }

} catch (Exception e) {

    JOptionPane.showMessageDialog(this, e.getMessage());

} }

```

1.3 Limitations

- There is no option for interest rate and on a current account we can save money which deffers from original bank system.
- No option for printing makes it difficult for account holder to view their account as we cant print a pass book.
- Every computer has to have its own database and record to handle the operation there is no provision to keep the data centerally on a server and use it from there.
- In the record there is no information which we can link to the record holder other than the mobile no whereas bank keeps there signature, address ,age ,sex and a photo for reference.
- No transaction log which leads to the information loss as we cannot view how many times this person have made a transaction.
- There is only one user login and pass which means we cannot have a unique set of pass and user account for each employee.

1.4 Bibliography

- Sumita Arora
- Reeta Sahoo & Gagan Sahoo Class XII and Class XI
- Stackoverflow.com
- w3schools.com