Batch: C22 | Roll Nos.: 2003094, 2003095, 2003104

Date: 10-12-2021

ATM Simulator

Problem Statement:

To develop a Java application that simulates an ATM (Automated teller Machine).

Explanation:

This project's main goal is to create a system that can imitate ATM transactions. The system will allow the user to withdraw cash, transfer money to another account, view a mini statement, and get balance information.

- At first, the user will be displayed with the screen to enter their ATM card number and pin.
- If the details entered are valid, the user will be displayed with the dashboard, else the user will be prompted with an error message. And again will be asked to enter the ATM card number and pin.
- In the dashboard, user will get an option to select either he/she wanna withdraw money, Get the account balance, Transfer money to some other account, get the debit history, and exit from the dashboard.
- If the user selects **Withdraw,** he/she will be asked to enter the amount. After entering the amount, if the user wanna proceed, will press the **Proceed** button.
- After a successful transaction, the user will then be prompted with a **Transaction Successful** message.
- If the user selects to get the **Balance amount,** he/she will be prompted with a message stating the balance amount.
- If the user selects **Transfer,** he/she will be displayed with a screen to enter the **Receiver’s Username and the amount** to be transferred. After a successful transaction the user will be notified about the successful transaction.
- If the user selects **Debit History,** The system will display the last 5 transactions.

Batch: C22 | Roll Nos.: 2003094, 2003095, 2003104

Date: 10-12-2021

Features of Java Used:

1. Swing:

   (Used to design the GUI components)

   We used **NetBeans** to create the GUI for our project

   NetBeans is an IDE for developing applications using different languages. Java programming language is one of those many languages which NetBeans supports.

   NetBeans is the **best IDE for building desktop applications using swing** since it provides a **user-friendly interface** for designing and coding the UI. It has **inbuilt libraries for swing which auto-populate code**.

   The in-built libraries in NetBeans provide a palette section in the IDE when creating a swing application. This makes it easier to **drag and drop** and components when designing an application.

   Java Swing is based on an abstract windowing toolkit API that is purely written in Java. Java Swing offers lightweight and platform-agnostic components, making it ideal for designing and creating desktop-based applications (systems).

   Common Swing components for desktop application (GUI)

   Java Swing components are contained in the `javax.swing` package. Java Swing Components are vital aspects for designing, developing, and implementing an application. The Swing Graphical User Interface widget toolkit for the Java programming language is made up of these components.

   These include:

   1. JFrame
   2. JLabel
   3. JTextField
   4. JButton

      Other include: JPanel, JComboBox, JRadioButton, JCheckBox, JTable, JList, JFileChose, JTextArea, ImageIcon.

      The features we used in our app are as follows:

      1. **JFrame**

         In the Java Swing Component hierarchy, `JFrame` is the very first component.

1. By constructing a Frame class object. That is, association-based creation.

   Example:

```
import javax.swing.JFrame;
public class Main {
     public static void main(String[] args) {
            JFrame sectionFrame = new JFrame();
            sectionFrame.setSize(600, 600); // 600 width & 600 height
            sectionFrame.setLayout(null); // no layout managers
            sectionFrame.setVisible(true); // frame visible
     }
}
```

2. By extending the Frame class. That is, OOP creation through class inheritance.

   Example:

```
import javax.swing.JFrame;
public class sectionHomeWindow extends JFrame {
    /**
     * Creates new form sectionHomeWindow
     */
    public sectionHomeWindow() {

    }
}
```

2. JLabel

`JLabel` is a Swing Container User Interface component that displays readable text or an image. The text presented in the `JLabel` cannot be edited by the application user.
The text can, however, be changed by the application itself via action events. Both plain and HTML text can be shown using the JLabel component.

Example:

```
JLabel labelName = new Jlabel(“Name”);
```

3. **JTextField**

   JTextField is a swing component that lets users enter a single line of text. JTextField is a child of the javax.swing Library's JTextComponent class.

   The enable property must be set to true for JTextField to be available and editable. JTextField can be initialised by giving an integer type parameter to the JTextField function.
   The number of characters a user can type in the text field is not limited by the integer argument given in the function. It does, however, provide the text field box's width, i.e. the number of columns to be assigned to it.

   Example:

   ```
   JTextField sectionTextField = new JTextField(20);
   ```

4. **JButton**

   JButton is one of the swing components which gives swing the property of platform independence. This component creates a click effect on the application’s user interface. It is implemented in an application by calling any of its class constructors.

   By clicking or double-clicking on it, data from databases is retrieved and displayed on the UI, or user data is collected on the UI and stored/saved in a database.
   In most applications, JButton has text or a picture that tells the user what the button does.

   ```
   JButton submitButton = new JButton("Submit");
   ```

2. Java Database Connectivity(JDBC)

JDBC is an acronym for Java Database Connectivity. JDBC is a standard API specification developed in order to move data from frontend to backend. This API consists of classes and interfaces written in Java. It basically acts as an interface (not the one we use in Java) or channel between your Java program and databases i.e it establishes a link between the two so that a programmer could send data from Java code and store it in the database for future use.

Batch: C22 | Roll Nos.: 2003094, 2003095, 2003104

Date: 10-12-2021

*** Steps for Connectivity between Java Program and a Database:**

1. Import the database
2. Load the drivers using the forName() method
3. Register the drivers using the DriverManager
4. Establish a connection using the Connection class object
5. Create a statement
6. Execute the query
7. Close the connections

Illustration

The above given **Steps from 1 to 4** can be executed in 2-lines of Java code given below.

```
Class.forName(“com.mysql.jdbc.Driver”); // load the drivers
Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/atm", "root",
"Kishan@1072"”);
```

Creating a Statement

Once a connection is established you can interact with the database. The JDBCStatement, CallableStatement, and PreparedStatement interfaces define the methods that enable you to send SQL commands and receive data from your database.
Use of JDBC Statement is as follows:

```
Statement st = con.createStatement();
```

Here, con is reference to Connection interface used in previous step.

Execute the Query

```
int m = st.executeUpdate(sql);
if (m==1)
    System.out.println("inserted successfully : "+sql);
else
    System.out.println("insertion failed");
```

Here, sql is SQL query of the type String.

Batch: C22 | Roll Nos.: 2003094, 2003095, 2003104

Date: 10-12-2021

Closing the connections

The close() method of the Connection interface is used to close the connection. It is as shown below as follows:

```
con.close();
```