Name:- Amruta S. Bagde

Email:- amrutabagde7@gmail.com

Task – 1: ATM Interface Project

Table of Contents

Acknowledgement

I would like to express my sincere gratitude to **Veritech Software Services** for providing me with the opportunity to work on this ATM Interface project. It has been a rewarding experience, allowing me to gain practical insights into Java programming and user interface design.and gain valuable hands-on experience in Java development.

Introduction

This Java project develops a graphical user interface (GUI) for an ATM simulation using Swing. The program allows users to perform typical ATM operations like checking balances, withdrawing money, and depositing funds. Through input validation and error handling, it provides a secure and intuitive experience. This project aims to replicate real-world ATM functionalities within a user-friendly interface, facilitating easy navigation and ensuring a seamless banking experience for users.

In the ATM Interface project, Swing, a Java GUI toolkit, is used to create an interactive user interface. It employs components like JFrame, JPanel, JTextField, and JButton for functionalities such as entering ATM details, checking balances, withdrawing, depositing funds, and viewing mini statements. Swing's event handling ensures efficient responses to user actions, contributing to a user-friendly and visually appealing ATM simulation.

Here are the features of ATM Interface Project....

- **GUI Development**: Utilizes Swing framework for creating a graphical user interface.
- **Authentication**: Users are required to input their ATM number and PIN for access.
- **Balance Inquiry**: Allows users to check their account balance.
- **Withdrawal**: Facilitates the withdrawal of funds from the account.
- **Deposit**: Enables users to deposit money into their account.

- **Mini Statement**: Provides a mini statement feature to view recent transactions.
- **Error Handling**: Implements mechanisms to handle invalid inputs and insufficient funds.
- **User-Friendly Interface**: Offers intuitive navigation and informative messages for ease of use.

This project combines these features to create a functional ATM simulation, aiming to provide users with a secure, efficient, and seamless banking experience.

Technologies

JAVA:-

- Java is a popular programming language developed by Sun Microsystems (now owned by Oracle Corporation) in 1995.
- It's high-level and object-oriented, designed to be easy to use and platform-independent.
- One of Java's key features is its platform independence, thanks to the Java Virtual Machine (JVM).
- This means Java programs can run on any device with a JVM installed, making it versatile and accessible.
- Java is widely used for building web applications, with millions of Java apps in use today.
- It's suitable for various devices and applications, including mobile apps, business software, and server-side technologies.
- Java is known for its speed, security, and reliability, making it ideal for handling large-scale projects and managing servers.
- It offers safety features like automatic memory management and strong typing.
- Java provides GUI frameworks like Swing, JavaFX, and Processing for creating user interfaces.
- These frameworks simplify the process of designing and implementing graphical elements in games and applications.
- The ATM Interface Application, a classic arcade game, can be developed using Java with GUI frameworks like Processing, Swing
- Java's platform independence, rich standard library, and support for object-oriented programming make it suitable for creating games like Snake.

Overall, Java's versatility, ease of use, platform independence, and robust features make it a preferred choice for a wide range of software development projects, including web applications and Interface like the ATM Machine using Swing framework.

Swing Framework :-

The Swing framework is a graphical user interface (GUI) toolkit for Java applications. It provides a comprehensive set of components for building desktop applications, offering features such as buttons, text fields, menus, and more. Swing follows the Model-View-Controller (MVC) architecture, separating the user interface from the application's logic.

One of Swing's key strengths is its platform independence, allowing developers to create applications that run seamlessly on different operating systems. It also offers a rich set of customizable components and supports advanced features like drag-and-drop functionality, internationalization, and accessibility.

<u>Goals in creation of JAVA</u>

The creation of the Java language aimed to achieve five key objectives:

- Ease of Use and Object-Oriented Design:
- Reliability and Safety:
- Cross-Platform Compatibility and Portability:
- Optimized Performance:
- Dynamic and Interpreted Execution with Threading Support:

<u>System Specifications</u>

Software Specifications:-

- Operating System:-Linux
- Coding Language:- JAVA
- GUI Framework:- Swing

Hardware Specification:-

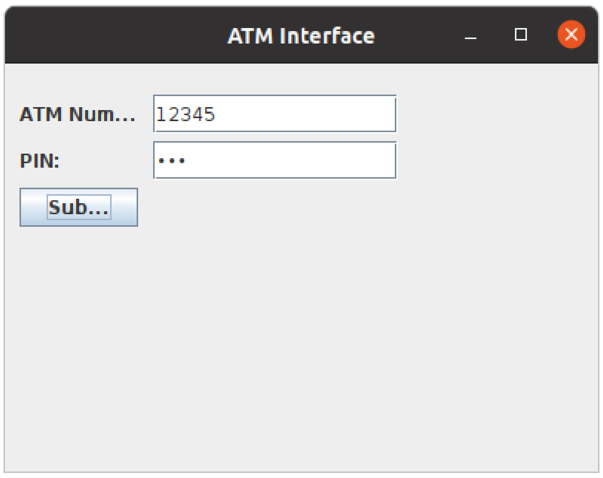
- System:-Intel core i5
- RAM:-7.7GB

<u>Functionality</u>

This Java code creates a basic Swing GUI for an ATM interface, featuring input fields for ATM number and PIN, as well as a submit button.

```
import javax.swing.*;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class Demo{
```

```
// Member variables for ATM details, balance, and mini statement
private int atmNumber = 12345;
private int atmPin = 123;

// Components for GUI
private JFrame frame;
private JPanel panel;
private JTextField atmNumberField;
private JPasswordField pinField;
private JButton submitButton;

// Constructor to set up GUI
public Demo() {
frame = new JFrame("ATM Interface");
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

panel = new JPanel();
frame.add(panel);

// Method to place components on the panel
placeComponents(panel);

frame.setVisible(true);
}

// Method to place components on the panel
private void placeComponents(JPanel panel) {
panel.setLayout(null);

// Label and text field for ATM number
JLabel atmNumberLabel = new JLabel("ATM Number:");
atmNumberLabel.setBounds(10, 20, 80, 25);
panel.add(atmNumberLabel);
```


ATM Interface

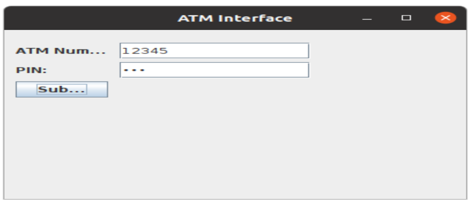
ATM Num... 12345

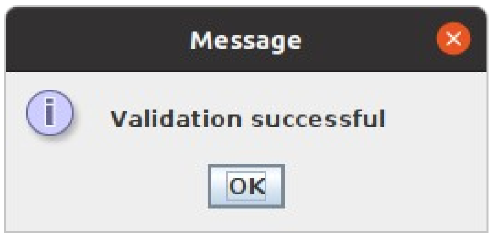
PIN: •••

Sub...

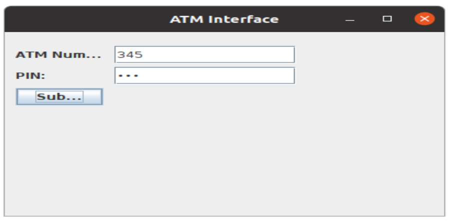
```
atmNumberField = new JTextField(20);
atmNumberField.setBounds(100, 20, 165, 25);
panel.add(atmNumberField);

// Label and password field for PIN
JLabel pinLabel = new Jlabel("PIN:");
pinLabel.setBounds(10, 50, 80, 25);
panel.add(pinLabel);

pinField = new JPasswordField(20);
pinField.setBounds(100, 50, 165, 25);
panel.add(pinField);

// Submit button
submitButton = new JButton("Submit");
submitButton.setBounds(10, 80, 80, 25);
panel.add(submitButton);




// Adding action listener to the submit button
submitButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
}
});
}

public static void main(String[] args) {
new Demo();
}
}
```

This code makes the submit button check if the entered ATM number and PIN are correct, showing either a success or error message.




Message

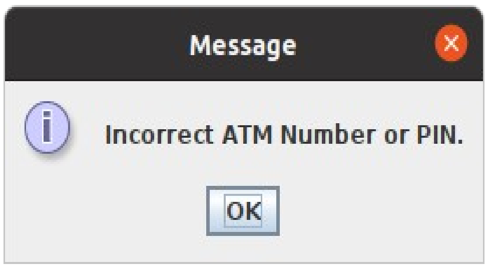
Validation successful



ATM Num... 345

PIN: •••

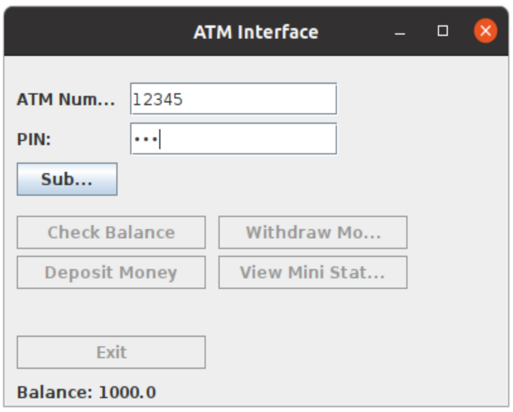
```
// Adding action listener to the submit button
submitButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
int enteredAtmNumber = Integer.parseInt(atmNumberField.getText());
int enteredPin = Integer.parseInt(new String(pinField.getPassword()));
if (atmNumber == enteredAtmNumber && atmPin == enteredPin) {
JOptionPane.showMessageDialog(null, "Validation successful")
} else {
JOptionPane.showMessageDialog(null, "Incorrect ATM Number or PIN.");
}
}
});
```


Message

Incorrect ATM Number or PIN.

This code creates a simple ATM interface where users can enter their ATM number and PIN to access features like checking balance, withdrawing money and many more.

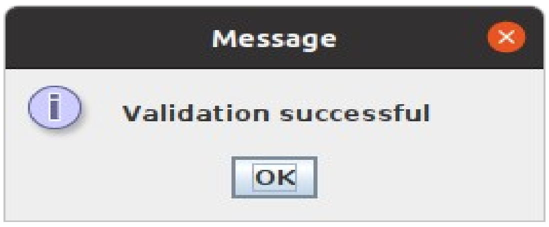
```
public class Demo {
    private float balance = 1000;

    // Components for GUI
    private JFrame frame;
    private JPanel panel;
    private JTextField atmNumberField;
    private JPasswordField pinField;
    private JButton submitButton;
    private JButton checkBalanceBtn;
    private JButton withdrawBtn;
    private JButton depositBtn;
    private JButton viewMiniStmtBtn;
    private JButton exitBtn; // New button for exiting
    private JLabel balanceLabel;

    // Constructor to set up GUI
```



ATM Num... 12345

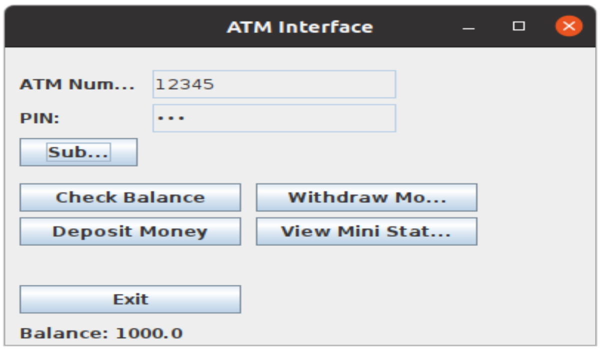
PIN: •••

Balance: 1000.0

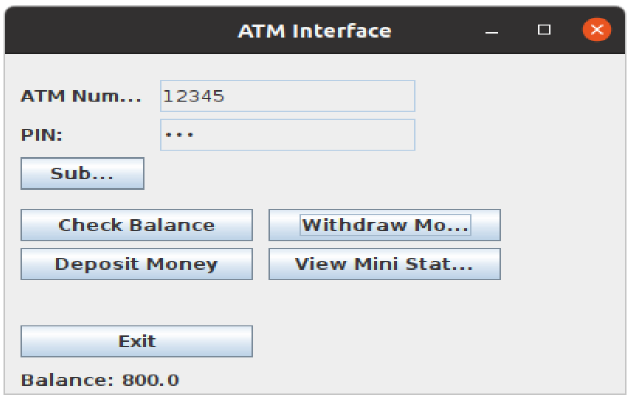

Message

Validation successful

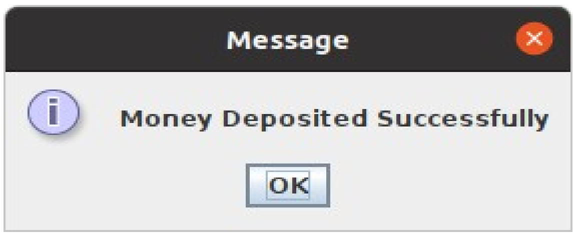
```
public Demo() {
    frame = new JFrame("ATM Interface");
    frame.setSize(400, 300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    panel = new Jpanel();
    frame.add(panel);

    // Method to place components on the panel
    placeComponents(panel);

    frame.setVisible(true);
}
```



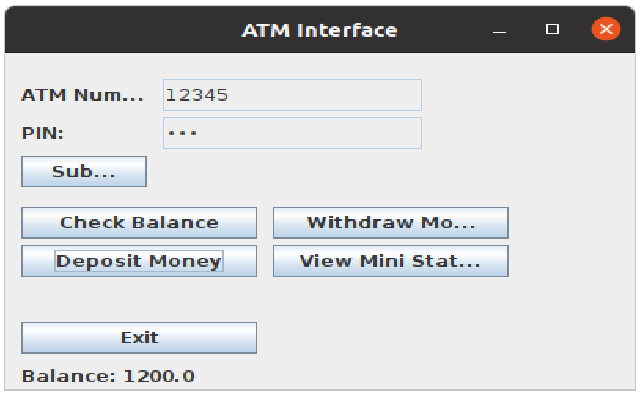
```
// Method to place components on the panel
private void placeComponents(JPanel panel) {
    panel.setLayout(null);

    // Label and text field for ATM number
    JLabel atmNumberLabel = new JLabel("ATM Number:");
    atmNumberLabel.setBounds(10, 20, 80, 25);
    panel.add(atmNumberLabel);

    atmNumberField = new JTextField(20);
    atmNumberField.setBounds(100, 20, 165, 25);
    panel.add(atmNumberField);

    // Label and password field for PIN
    JLabel pinLabel = new JLabel("PIN:");
    pinLabel.setBounds(10, 50, 80, 25);
    panel.add(pinLabel);

    pinField = new JPasswordField(20);
    pinField.setBounds(100, 50, 165, 25);
    panel.add(pinField);
```



```
// Submit button
submitButton = new JButton("Submit");
submitButton.setBounds(10, 80, 80, 25);
panel.add(submitButton);

checkBalanceBtn = new JButton("Check Balance");
checkBalanceBtn.setBounds(10, 120, 150, 25);
panel.add(checkBalanceBtn);
checkBalanceBtn.setEnabled(false);

withdrawBtn = new JButton("Withdraw Money");
withdrawBtn.setBounds(170, 120, 150, 25);
panel.add(withdrawBtn);
withdrawBtn.setEnabled(false);

depositBtn = new JButton("Deposit Money");
depositBtn.setBounds(10, 150, 150, 25);
panel.add(depositBtn);
depositBtn.setEnabled(false);

viewMiniStmtBtn = new JButton("View Mini Statement");
viewMiniStmtBtn.setBounds(170, 150, 150, 25);
panel.add(viewMiniStmtBtn);
viewMiniStmtBtn.setEnabled(false);

// New exit button
exitBtn = new JButton("Exit");
exitBtn.setBounds(10, 210, 150, 25);
panel.add(exitBtn);
exitBtn.setEnabled(false);

balanceLabel = new JLabel("Balance: " + balance);
balanceLabel.setBounds(10, 240, 150, 25);
panel.add(balanceLabel);
```


Message

Money Deposited Successfully


ATM Interface

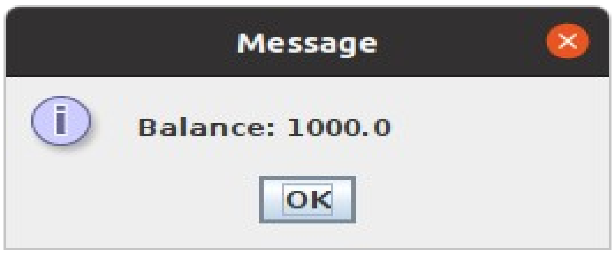
ATM Num... 12345
PIN: •••

Balance: 1200.0

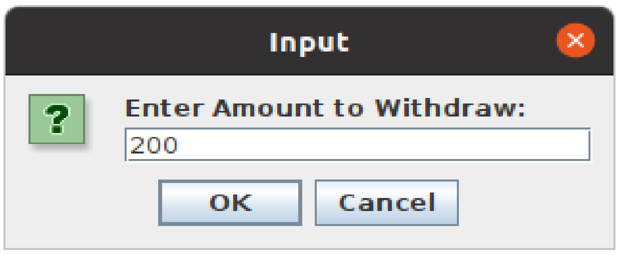
```
        // Adding action listener to the submit button
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int enteredAtmNumber = Integer.parseInt(atmNumberField.getText());
                int enteredPin = Integer.parseInt(new String(pinField.getPassword()));
                if (atmNumber == enteredAtmNumber && atmPin == enteredPin) {
                    JOptionPane.showMessageDialog(null, "Validation successful");
                    enableButtons(true);
                    atmNumberField.setEditable(false);
                    pinField.setEditable(false);
                } else {
                    JOptionPane.showMessageDialog(null, "Incorrect ATM Number or PIN.");
                    enableButtons(false);
                }
            }
        });
    }

    private void enableButtons(boolean enable) {
        checkBalanceBtn.setEnabled(enable);
        withdrawBtn.setEnabled(enable);
        depositBtn.setEnabled(enable);
        viewMiniStmtBtn.setEnabled(enable);
        exitBtn.setEnabled(enable);
    }
    public static void main(String[] args) {
        new Demo();
    }
}
```

This Java code creates a simple ATM interface with Swing GUI components, allowing users to perform basic banking operations like checking balance, withdrawing money, and depositing funds. It includes event listeners for user interactions and updates the interface dynamically based on user input.

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;


public class Demo {
    // Member variables for ATM details, balance, and mini statement
    private float balance = 1000;
    private int atmNumber = 12345;
    private int atmPin = 123;
    private HashMap<Double, String> miniStmt = new HashMap<>();

    // Components for GUI
    private JFrame frame;
    private JPanel panel;
    private JTextField atmNumberField;
    private JPasswordField pinField;
    private JButton submitButton;
    private JButton checkBalanceBtn;
    private JButton withdrawBtn;
    private JButton depositBtn;
    private JButton viewMiniStmtBtn;
    private JButton exitBtn; // New button for exiting
    private JLabel balanceLabel;

    // Constructor to set up GUI
    public Demo() {
        frame = new JFrame("ATM Interface");
```


Balance: 1000.0

```
    frame.setSize(400, 300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    panel = new JPanel();
    frame.add(panel);

    // Method to place components on the panel
    placeComponents(panel);

    frame.setVisible(true);
}

// Method to place components on the panel
private void placeComponents(JPanel panel) {
    panel.setLayout(null);

    // Label and text field for ATM number
    JLabel atmNumberLabel = new JLabel("ATM Number:");
    atmNumberLabel.setBounds(10, 20, 80, 25);
    panel.add(atmNumberLabel);

    atmNumberField = new JTextField(20);
    atmNumberField.setBounds(100, 20, 165, 25);
    panel.add(atmNumberField);

    // Label and password field for PIN
    JLabel pinLabel = new JLabel("PIN:");
    pinLabel.setBounds(10, 50, 80, 25);
    panel.add(pinLabel);

    pinField = new JPasswordField(20);
    pinField.setBounds(100, 50, 165, 25);
    panel.add(pinField);
```

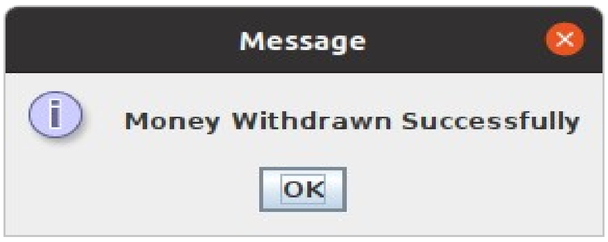


Input

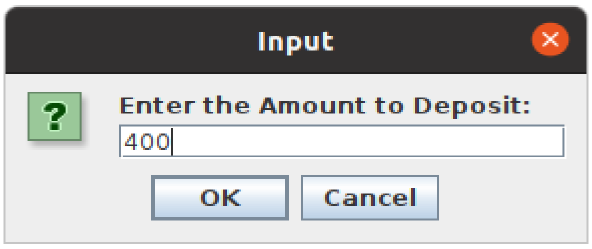
Enter Amount to Withdraw:

200

```
// Submit button
submitButton = new JButton("Submit");
submitButton.setBounds(10, 80, 80, 25);
panel.add(submitButton);

checkBalanceBtn = new JButton("Check Balance");
checkBalanceBtn.setBounds(10, 120, 150, 25);
panel.add(checkBalanceBtn);
checkBalanceBtn.setEnabled(false);

withdrawBtn = new JButton("Withdraw Money");
withdrawBtn.setBounds(170, 120, 150, 25);
panel.add(withdrawBtn);
withdrawBtn.setEnabled(false);

depositBtn = new JButton("Deposit Money");
depositBtn.setBounds(10, 150, 150, 25);
panel.add(depositBtn);
depositBtn.setEnabled(false);

viewMiniStmtBtn = new JButton("View Mini Statement");
viewMiniStmtBtn.setBounds(170, 150, 150, 25);
panel.add(viewMiniStmtBtn);
viewMiniStmtBtn.setEnabled(false);

// New exit button
exitBtn = new JButton("Exit");
exitBtn.setBounds(10, 210, 150, 25);
panel.add(exitBtn);
exitBtn.setEnabled(false);

balanceLabel = new JLabel("Balance: " + balance);
balanceLabel.setBounds(10, 240, 150, 25);
panel.add(balanceLabel);
```

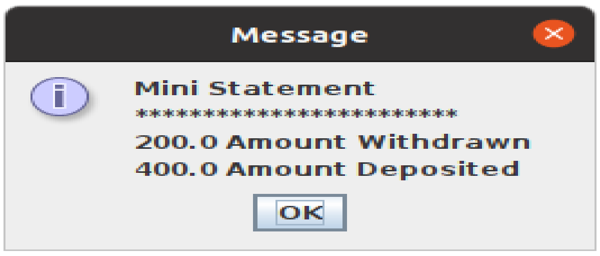

Message

Money Withdrawn Successfully


Input

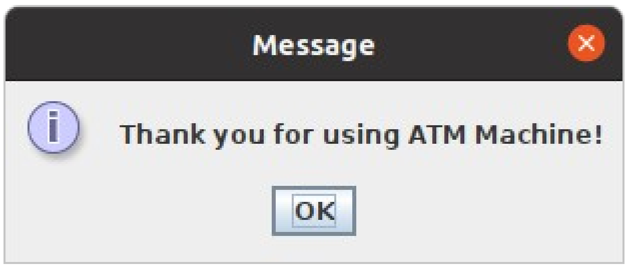
Enter the Amount to Deposit:

400


Message

```
Mini Statement
*************************
200.0 Amount Withdrawn
400.0 Amount Deposited
```

```
// Adding action listener to the submit button
submitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int enteredAtmNumber = Integer.parseInt(atmNumberField.getText());
        int enteredPin = Integer.parseInt(new String(pinField.getPassword()));
        if (atmNumber == enteredAtmNumber && atmPin == enteredPin) {
            JOptionPane.showMessageDialog(null, "Validation successful");
            enableButtons(true);
            atmNumberField.setEditable(false);
            pinField.setEditable(false);
        } else {
            JOptionPane.showMessageDialog(null, "Incorrect ATM Number or PIN.");
            enableButtons(false);
        }
    }
});

// Adding action listener to the existing buttons
checkBalanceBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "Balance: " + balance);
    }
});

withdrawBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String amountString = JOptionPane.showInputDialog("Enter Amount to Withdraw:");
        if (amountString != null) {
            float amount = Float.parseFloat(amountString);
            if (amount > balance) {
                JOptionPane.showMessageDialog(null, "Insufficient Balance");
            } else {
                balance -= amount;
                JOptionPane.showMessageDialog(null, "Money Withdrawn Successfully");
                updateBalanceLabel();
                miniStmt.put((double) amount, "Amount Withdrawn");
              }
            }
          }
        });

        depositBtn.addActionListener(new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
            String amountString = JOptionPane.showInputDialog("Enter the Amount to Deposit:");
            if (amountString != null) {
              float amount = Float.parseFloat(amountString);
              balance + = amount;
              JOptionPane.showMessageDialog(null, "Money Deposited Successfully");
              updateBalanceLabel();
              miniStmt.put((double) amount, "Amount Deposited");
            }
          }
        });

        viewMiniStmtBtn.addActionListener(new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
            StringBuilder miniStatement = new StringBuilder("Mini Statement\n************************\
n");

            for (Map.Entry<Double, String> entry : miniStmt.entrySet()) {
              miniStatement.append(entry.getKey()).append(" ").append(entry.getValue()).append("\n");
            }
            JOptionPane.showMessageDialog(null, miniStatement.toString());
            enableButtons(true); // Enable the exit button after viewing the mini statement
          }
        });

        // Action listener for the exit button
```



Message

Thank you for using ATM Machine!

```
        exitBtn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                displayThankYouMessage();
            }
        });
    }

    private void enableButtons(boolean enable) {
        checkBalanceBtn.setEnabled(enable);
        withdrawBtn.setEnabled(enable);
        depositBtn.setEnabled(enable);
        viewMiniStmtBtn.setEnabled(enable);
        exitBtn.setEnabled(enable);
    }

    private void updateBalanceLabel() {
        balanceLabel.setText("Balance: " +  balance);
    }

    private void displayThankYouMessage() {
        JOptionPane.showMessageDialog(null, "Thank you for using ATM Machine!");
        System.exit(0); // Exit the application
    }

    public static void main(String[] args) {
        new Demo();
}
```

Future Scope of the Project

In the future, our ATM Interface project will get better. We'll add more things you can do, like paying bills, transferring money to other banks, and getting account statements. We'll also make it safer by adding features like using your fingerprint or a code sent to your phone to log in. You'll be able to change how it looks and works to suit your needs. Plus, you'll be able to do your banking on your phone too. These changes will make using our ATM Interface easier and safer for everyone.

Conclusion

In short, the ATM Interface project made with Java is a user–friendly way to do banking stuff. It's easy to use and lets you do things like checking your balance and taking out money safely. In the future, it might get even safer and have more options, like letting you pay bills and change how it looks. Overall, it's a handy tool for managing your money.