Index

Abstract

In today's digital banking era, ATM (Automated Teller Machine) systems play a vital role in providing 24/7 banking services to users. This project presents a simplified and secure ATM machine simulation developed using Java. It demonstrates core functionalities such as **user authentication (PIN verification), balance inquiry, cash withdrawal, deposit, and transaction history**.

The system uses **OOP (Object-Oriented Programming)** concepts like **encapsulation**, **inheritance**, and **abstraction** to organize the code efficiently. The user interacts through a menu-driven interface that simulates real ATM operations.

Abstraction is used to hide complex internal processes such as account management and PIN validation, offering the user only the essential options needed to complete transactions. This enhances security and usability.

This project provides hands-on experience with Java programming and lays the foundation for understanding how banking systems work behind the scenes. It can be further extended with features like GUI (using Java Swing), card reader simulation, or database integration for real-time data handling.

Hindalco Summer Training Project

Project Report

2025

Hindalco Nexus

Connecting People, Teams & Solutions

Supervised To:
Mr. Digvijay Singh

Submitted By:
Shalini Singh

INTRODUCTION

In today's digital world, people expect fast, secure, and convenient access to their banking services. Traditional ATM systems often lack flexibility for updates, real-time tracking, and user-friendly interfaces for learning or simulation purposes.

This project introduces a simplified **ATM Machine System**, developed as a console-based or GUI-based application using Java. The system mimics real-world ATM functionalities such as PIN verification, balance inquiry, cash deposit, and withdrawal.

It serves as a learning model for understanding how core banking transactions work while implementing key concepts of object-oriented programming, file handling, and user input validation. This project not only demonstrates practical coding skills but also helps visualize how secure financial operations are carried out in a controlled, step-by-step process.

.

OBJECTIVE

The objective of this project is to design and implement a **Graphical ATM (Automated Teller Machine) System** using **Java Swing**, providing users with a secure, interactive, and user-friendly banking experience. The system simulates real-world ATM functionalities through an intuitive graphical interface. Key goals of the project include:

- **Secure User Authentication**:
  Implement a PIN-based login system to ensure that only authorized users can access their account.
- **Core Banking Operations**:
  Allow users to perform essential banking tasks such as:
  - **Balance Inquiry**
  - **Cash Withdrawal**
  - **Cash Deposit**
  - **PIN Change**
- **Graphical User Interface (GUI)**:
  Use **Java Swing** to create an interactive and visually engaging interface that mimics a real ATM machine, with buttons, input fields, and dialogs for better user interaction.
- **Real-Time Transaction Updates**:
  Immediately update the account balance after each transaction, ensuring accurate and consistent financial data.

- **Input Validation & Error Handling**:
  Handle incorrect PIN entries, invalid amounts, and insufficient balance conditions gracefully with proper error messages and dialog prompts.
- **Extensibility & Modularity**:
  Build the system in a modular way to allow future enhancements such as:
  - Mini statements
  - Multi-user login
  - Bank database integration
  - Card number simulation
- **Educational Simulation**:
  Serve as a learning model to understand the logic and flow of real ATM systems while applying Java programming and GUI design skills.

SIGNIFICANCE

The ATM Machine project holds significant educational and practical value by simulating core banking functionalities through a secure and user-friendly interface. It plays an important role in understanding the workings of digital banking systems and Java GUI development. The key contributions of this project include:

- **Understanding Real-World Banking Systems**:
  Simulates real-life ATM operations like cash withdrawal, deposit, balance inquiry, and PIN change, helping learners grasp the fundamentals of banking logic.
- **Promotes Secure Transaction Handling**:
  Demonstrates how user authentication and PIN-based access ensure security and prevent unauthorized use.
- **Hands-On GUI Development Using Java Swing**:
  Enhances skills in building interactive desktop applications, focusing on layout design, event handling, and user interaction.
- **Improves Error Handling & Input Validation**:
  Encourages building reliable systems by managing incorrect inputs, insufficient balance cases, and failed transactions effectively.
- **Modular and Scalable Structure**:
  Lays the foundation for future enhancements like:
  - Mini statements
  - Multi-user accounts
  - Database integration
  - Card simulation
- **Ideal for Learning and Demonstration**:
  Serves as a valuable beginner-level Java project to showcase object-oriented programming, GUI design, and problem-solving skills.

KEY FEATURES

- **Secure PIN-Based Authentication**:
  Users must enter a valid 4-digit PIN to access the ATM system, ensuring basic security and preventing unauthorized usage.
- **Graphical User Interface (GUI)**:
  Designed using **Java Swing** to simulate a real ATM interface with buttons, input fields, and interactive dialog boxes for better user experience.
- **Core Banking Functions**:
  The system supports essential ATM operations, including:
  - **Balance Inquiry**: View the current available balance.
  - **Cash Withdrawal**: Withdraw money with proper balance validation.
  - **Cash Deposit**: Deposit money and update balance in real-time.
  - **PIN Change**: Change the current PIN after successful authentication.
- **Real-Time Balance Updates**:
  Account balance is updated immediately after each transaction, reflecting accurate and up-to-date information.
- **Input Validation & Error Handling**:
  Handles invalid inputs like wrong PIN, insufficient balance, or negative deposit amounts with meaningful error messages.

- **Modular Code Structure**:
  The system is divided into well-organized classes and methods, making it easy to understand, maintain, and extend.
- **Simulated ATM Workflow**:
  The application follows a real-world ATM flow — login, menu navigation, transaction, and exit — to give users an authentic experience.
- **Future Scope for Enhancements**:
  The system is designed with flexibility to add more features in the future, such as:
  - Mini statement generation
  - Database integration for real user data
  - Card number and multi-account simulation
  - Receipt printing and transaction logs

ER Diagram for ATM Machine Project

🟦 Entities & Attributes

1. **User**
    - *User ID* (Primary Key)
    - Name
    - Card Number
    - PIN
    - Balance
2. **Transaction**
    - *Transaction ID* (Primary Key)
    - User ID (Foreign Key)
    - Transaction Type (Deposit / Withdraw / Balance Inquiry / PIN Change)
    - Amount
    - Date Time

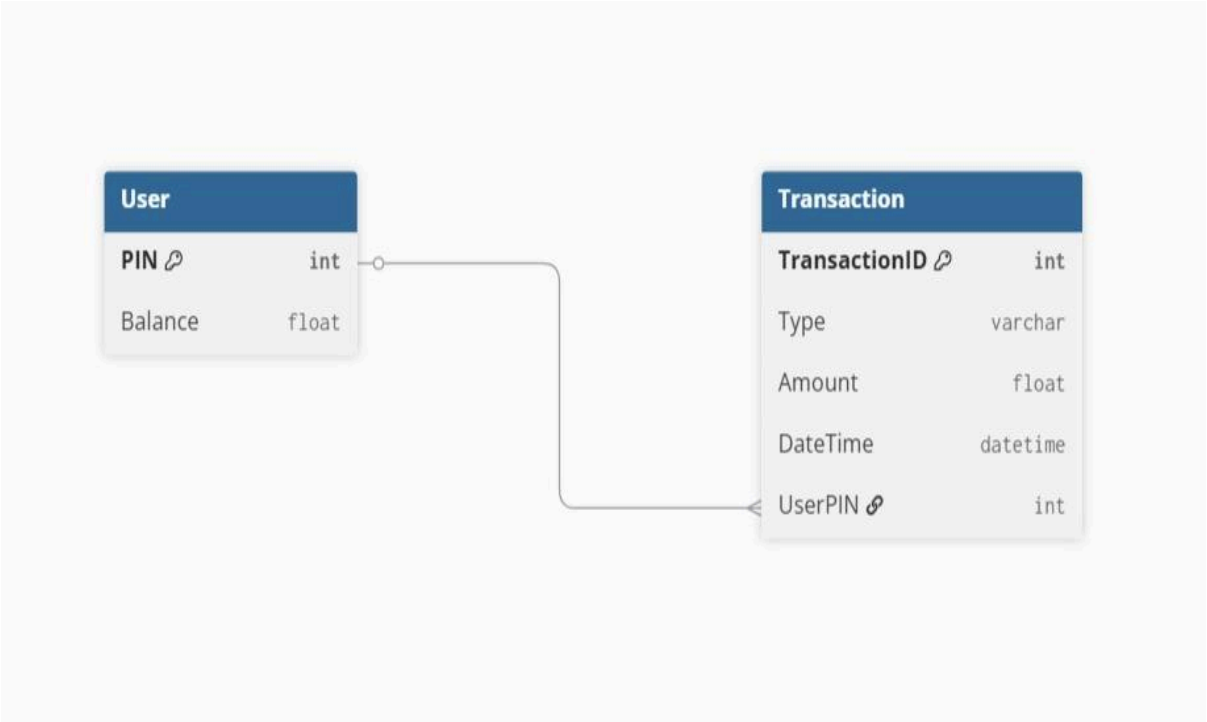
🔗 Relationships

- A **User** can perform **multiple Transactions**
  → **One-to-Many** relationship between User and Transaction

---

🔧

ER Diagram Representation (Text-Based View)



BENEFITS

1. 24/7 Banking Convenience

Allows users to access banking services such as cash withdrawal, deposit, and balance inquiry anytime, reducing the need to visit a bank branch.

2. Enhanced User Experience with Digital Interface

A user-friendly interface simplifies banking operations, making it easy for customers to perform transactions quickly and securely.

3. Secure and Reliable Transactions

Implements PIN-based authentication and transaction logs to ensure secure operations and protect user data from unauthorized access.

4. Time and Resource Efficiency

Reduces the workload on bank staff by automating routine transactions, thereby saving time and improving service efficiency.

5. Realistic Simulation for Learning and Demonstration

Provides a real-world simulation of ATM operations, making it ideal for educational projects and training purposes.

6. Expandable Functionalities

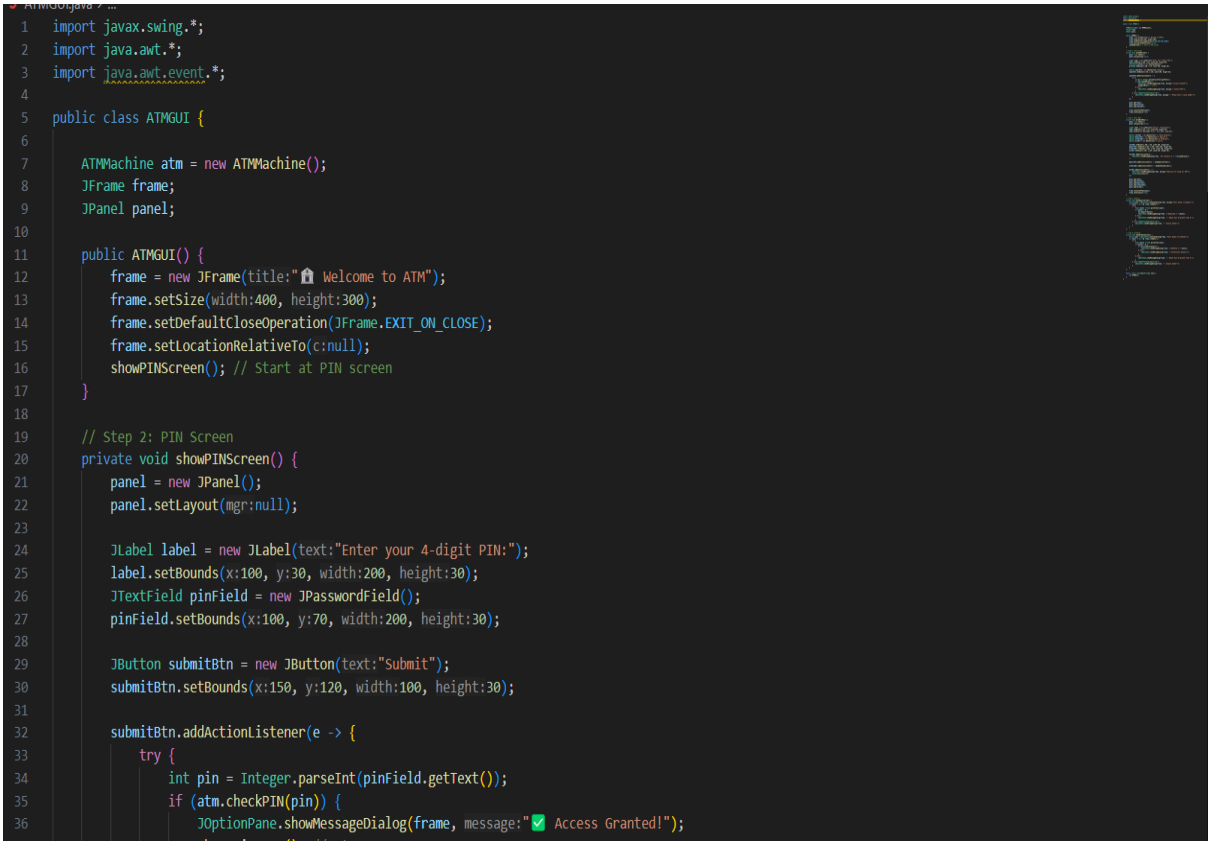
The system can be extended with features like mini-statements, fund transfers, or multi-language support, offering scalability for future improvements.

IMPLEMENTATIONS

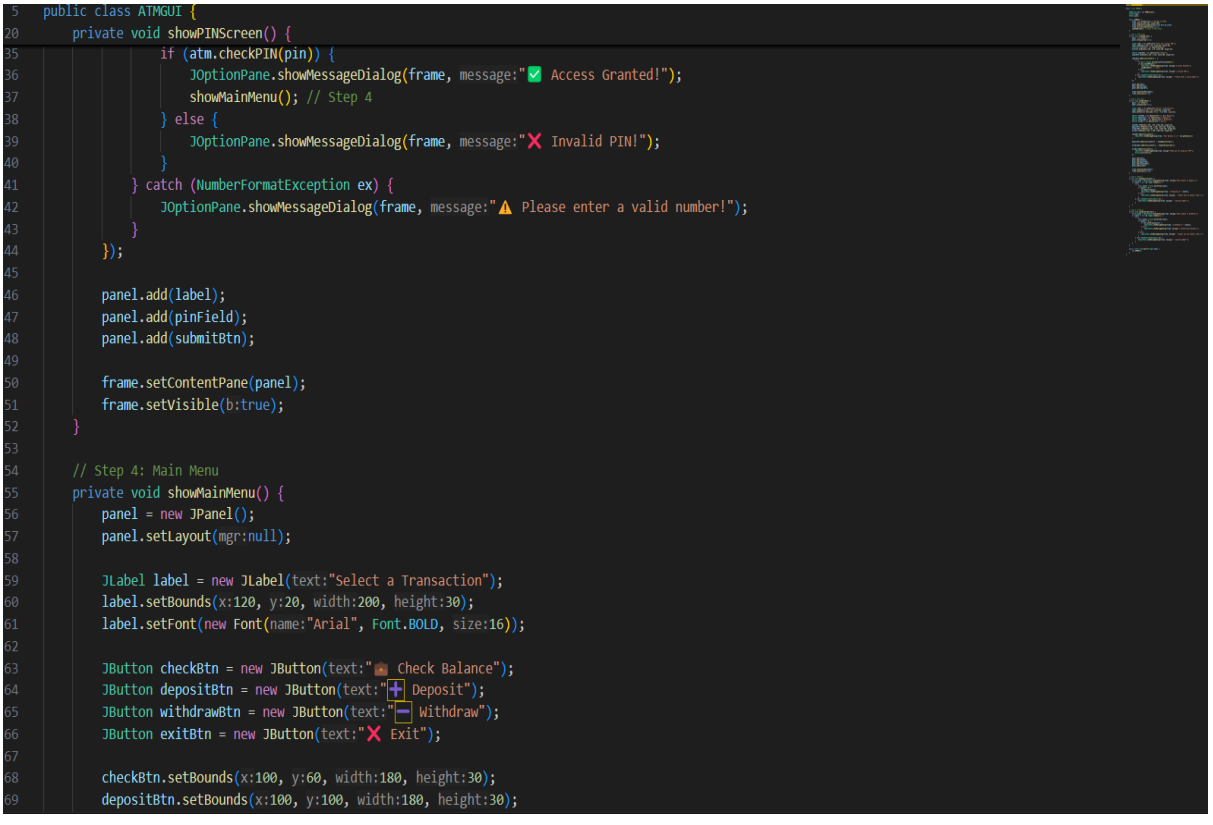
- **Java-Based Application Architecture**:
  Developed the ATM system using Java, focusing on object-oriented principles to structure features like account management, transactions, and authentication.
- **Graphical User Interface with Java Swing**:
  Designed an interactive GUI for user operations such as Withdraw, Deposit, Check Balance, and Exit, providing a realistic ATM experience.
- **File Handling for Data Storage**:
  Implemented file-based data storage to manage user account details, balances, and transaction history securely without needing a database.
- **PIN Authentication System**:
  Integrated a secure login mechanism using a 4-digit PIN to restrict unauthorized access and simulate real-world ATM verification.
- **Transaction Processing Logic**:
  Handled core banking operations such as withdrawal with balance validation, deposits, and updating account balances after each transaction.
- **User Feedback and Error Handling**:
  Displayed clear messages for every transaction and incorporated error handling for invalid inputs, incorrect PINs, and insufficient balance.
- **Scalable Design for Future Enhancements**:
  Structured the application to support future features like mini-statement, fund transfer, receipt printing, or multi-user support.

CHALLENGES

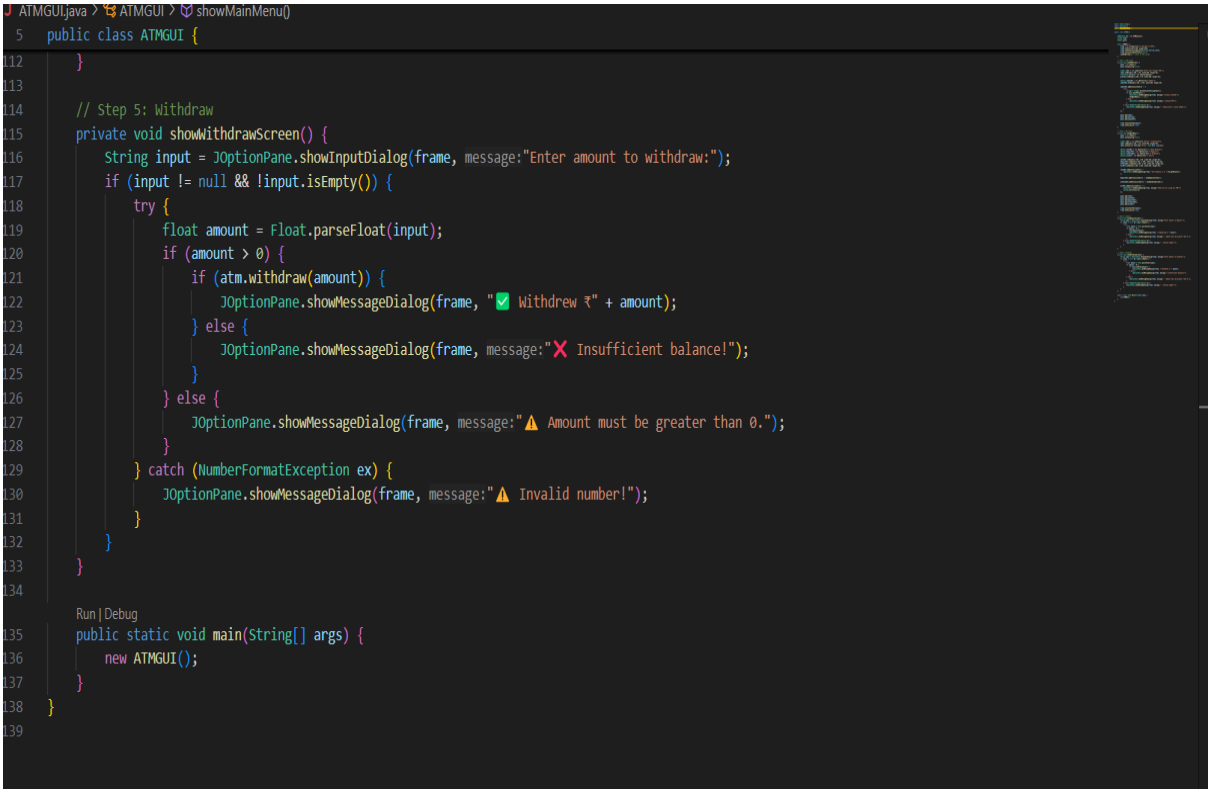
- **PIN-Based Authentication Handling**:
  Ensuring secure and accurate user login through 4-digit PIN verification while preventing unauthorized access.
- **Transaction Accuracy and Balance Validation**:
  Implementing precise logic to handle deposits, withdrawals, and balance checks while validating limits and preventing overdrafts.
- **File Handling for Persistent Data Storage**:
  Managing user account details and transaction history using Java File I/O without data corruption or loss.
- **GUI Input Management with Swing**:
  Handling dynamic input fields based on user actions (Withdraw, Deposit, etc.) while maintaining a smooth and interactive interface.
- **User Feedback and Error Handling**:
  Providing clear and informative messages for invalid inputs, insufficient balance, and incorrect PIN attempts to improve usability.
- **Extensibility for Future Features**:
  Designing a modular system that allows easy addition of future functionalities like mini-statement, fund transfer, or receipt printing.



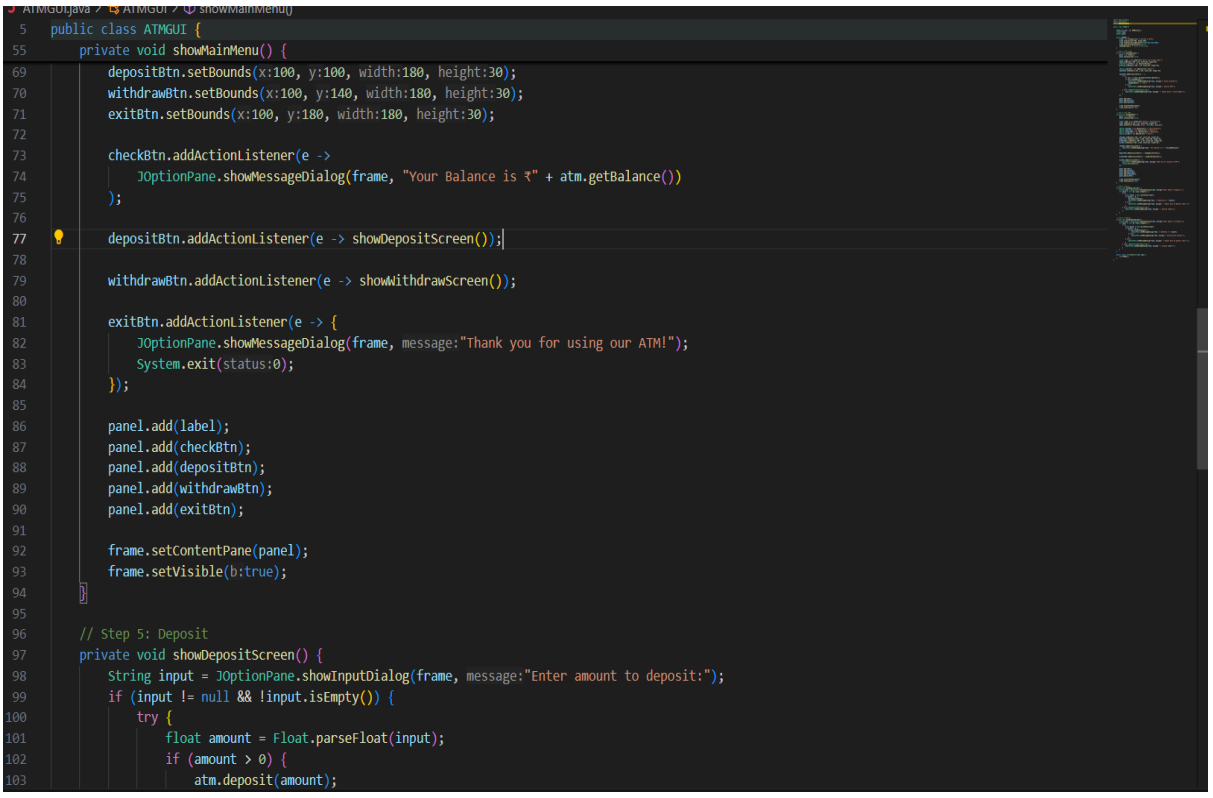
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ATMGUI {

    ATMMachine atm = new ATMMachine();
    JFrame frame;
    JPanel panel;

    public ATMGUI() {
        frame = new JFrame(title:"🏦 Welcome to ATM");
        frame.setSize(width:400, height:300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(c:null);
        showPINScreen(); // Start at PIN screen
    }

    // Step 2: PIN Screen
    private void showPINScreen() {
        panel = new JPanel();
        panel.setLayout(mgr:null);

        JLabel label = new JLabel(text:"Enter your 4-digit PIN:");
        label.setBounds(x:100, y:30, width:200, height:30);
        JTextField pinField = new JPasswordField();
        pinField.setBounds(x:100, y:70, width:200, height:30);

        JButton submitBtn = new JButton(text:"Submit");
        submitBtn.setBounds(x:150, y:120, width:100, height:30);

        submitBtn.addActionListener(e -> {
            try {
                int pin = Integer.parseInt(pinField.getText());
                if (atm.checkPIN(pin)) {
                    JOptionPane.showMessageDialog(frame, message:"✅ Access Granted!");
```



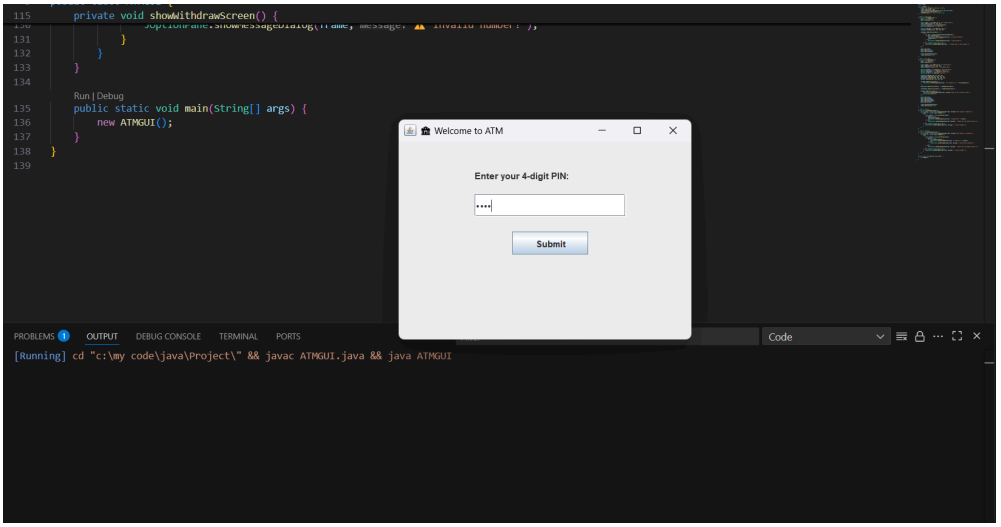
```java
public class ATMGUI {
    private void showPINScreen() {
                if (atm.checkPIN(pin)) {
                    JOptionPane.showMessageDialog(frame, message:"✅ Access Granted!");
                    showMainMenu(); // Step 4
                } else {
                    JOptionPane.showMessageDialog(frame, message:"❌ Invalid PIN!");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, message:"⚠️ Please enter a valid number!");
            }
        });

        panel.add(label);
        panel.add(pinField);
        panel.add(submitBtn);

        frame.setContentPane(panel);
        frame.setVisible(b:true);
    }

    // Step 4: Main Menu
    private void showMainMenu() {
        panel = new JPanel();
        panel.setLayout(mgr:null);

        JLabel label = new JLabel(text:"Select a Transaction");
        label.setBounds(x:120, y:20, width:200, height:30);
        label.setFont(new Font(name:"Arial", Font.BOLD, size:16));

        JButton checkBtn = new JButton(text:"💼 Check Balance");
        JButton depositBtn = new JButton(text:"➕ Deposit");
        JButton withdrawBtn = new JButton(text:"➖ Withdraw");
        JButton exitBtn = new JButton(text:"❌ Exit");

        checkBtn.setBounds(x:100, y:60, width:180, height:30);
        depositBtn.setBounds(x:100, y:100, width:180, height:30);
```

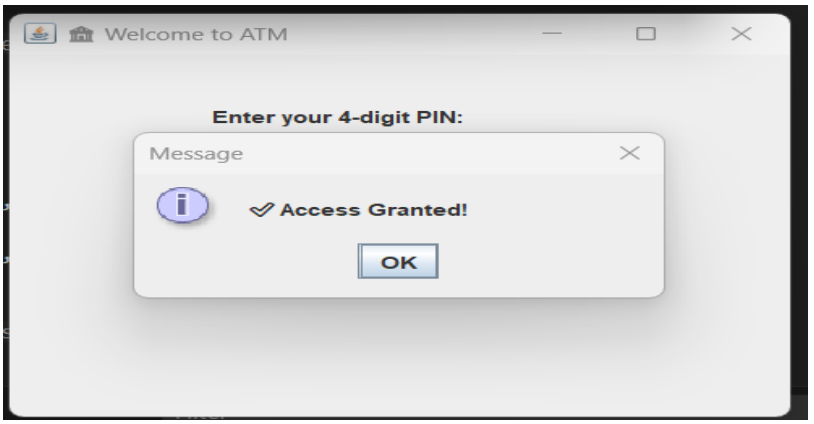


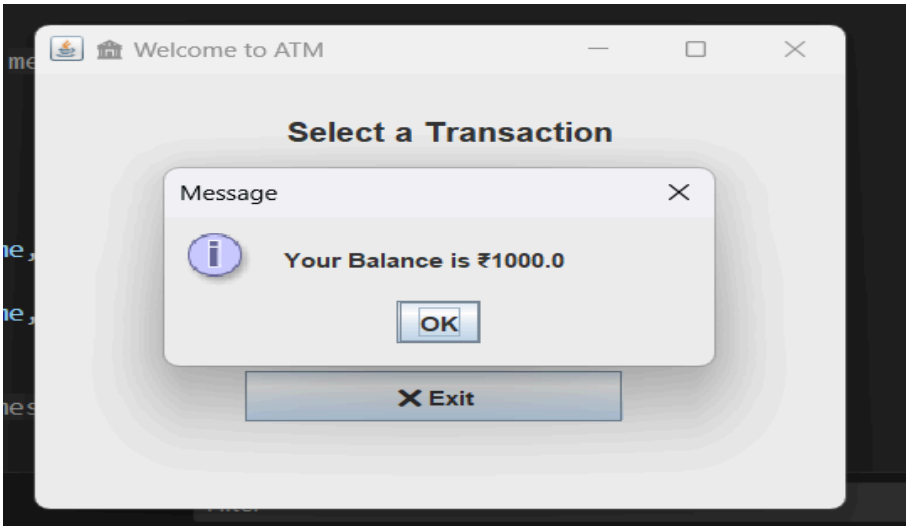
```
ATMGUI.java > ATMGUI > showMainMenu()
5   public class ATMGUI {
112     }
113
114     // Step 5: Withdraw
115     private void showWithdrawScreen() {
116         String input = JOptionPane.showInputDialog(frame, message:"Enter amount to withdraw:");
117         if (input != null && !input.isEmpty()) {
118             try {
119                 float amount = Float.parseFloat(input);
120                 if (amount > 0) {
121                     if (atm.withdraw(amount)) {
122                         JOptionPane.showMessageDialog(frame, "✅ Withdrew ₹" + amount);
123                     } else {
124                         JOptionPane.showMessageDialog(frame, message:"❌ Insufficient balance!");
125                     }
126                 } else {
127                     JOptionPane.showMessageDialog(frame, message:"⚠️ Amount must be greater than 0.");
128                 }
129             } catch (NumberFormatException ex) {
130                 JOptionPane.showMessageDialog(frame, message:"⚠️ Invalid number!");
131             }
132         }
133     }
134
        Run | Debug
135     public static void main(String[] args) {
136         new ATMGUI();
137     }
138 }
139
```

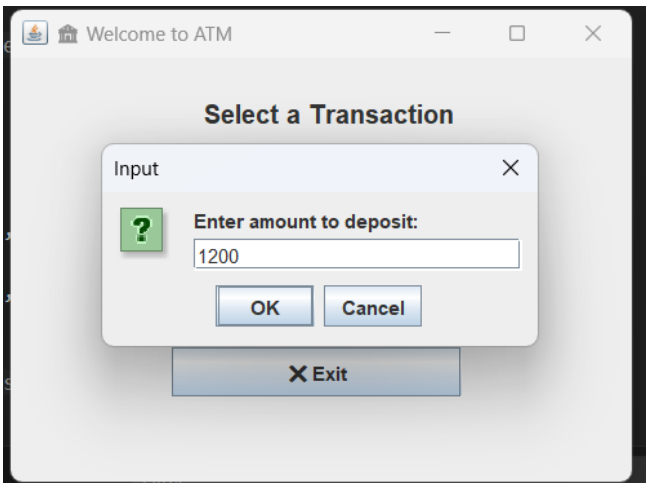


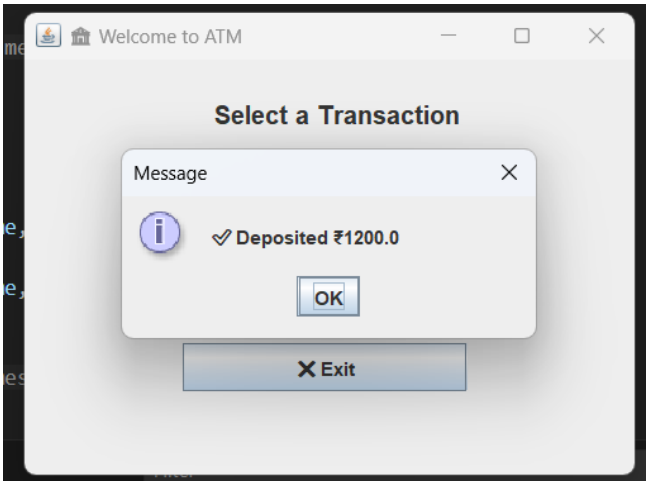
```
ATMGUI.java > ATMGUI > showMainMenu()
5   public class ATMGUI {
55      private void showMainMenu() {
69          depositBtn.setBounds(x:100, y:100, width:180, height:30);
70          withdrawBtn.setBounds(x:100, y:140, width:180, height:30);
71          exitBtn.setBounds(x:100, y:180, width:180, height:30);
72
73          checkBtn.addActionListener(e ->
74              JOptionPane.showMessageDialog(frame, "Your Balance is ₹" + atm.getBalance())
75          );
76
77          depositBtn.addActionListener(e -> showDepositScreen());
78
79          withdrawBtn.addActionListener(e -> showWithdrawScreen());
80
81          exitBtn.addActionListener(e -> {
82              JOptionPane.showMessageDialog(frame, message:"Thank you for using our ATM!");
83              System.exit(status:0);
84          });
85
86          panel.add(label);
87          panel.add(checkBtn);
88          panel.add(depositBtn);
89          panel.add(withdrawBtn);
90          panel.add(exitBtn);
91
92          frame.setContentPane(panel);
93          frame.setVisible(b:true);
94      }
95
96      // Step 5: Deposit
97      private void showDepositScreen() {
98          String input = JOptionPane.showInputDialog(frame, message:"Enter amount to deposit:");
99          if (input != null && !input.isEmpty()) {
100             try {
101                 float amount = Float.parseFloat(input);
102                 if (amount > 0) {
103                     atm.deposit(amount);
```

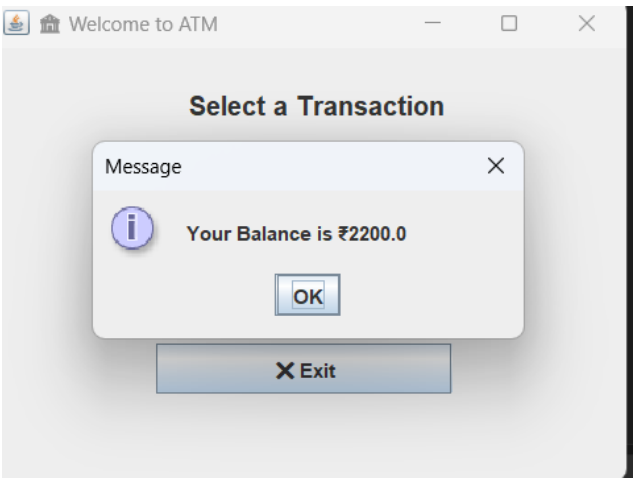

private void showWithdrawScreen() {
}
}
}
Run | Debug
public static void main(String[] args) {
new ATMGUI();
}
}
Welcome to ATM
Enter your 4-digit PIN:
Submit
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Code
[Running] cd "c:\my code\java\Project\" && javac ATMGUI.java && java ATMGUI

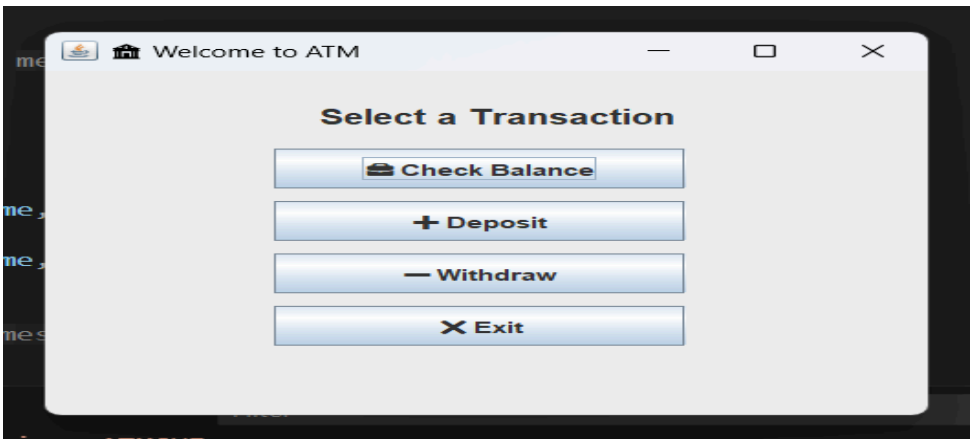

Welcome to ATM
Enter your 4-digit PIN:
Message
Access Granted!
OK

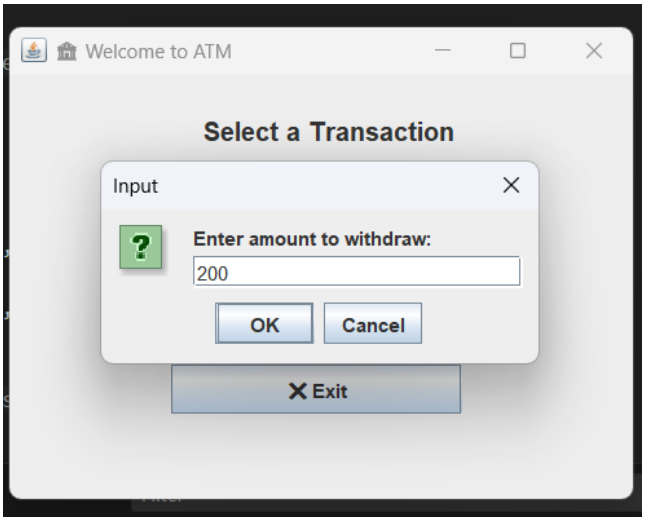

Welcome to ATM
Select a Transaction
Message
Your Balance is ₹1000.0
OK
Exit

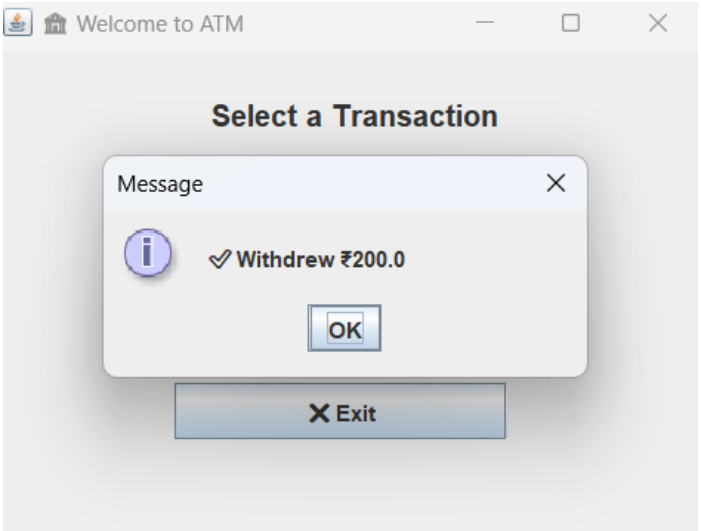

Welcome to ATM
Select a Transaction
Input
Enter amount to deposit:
1200
OK
Cancel
Exit

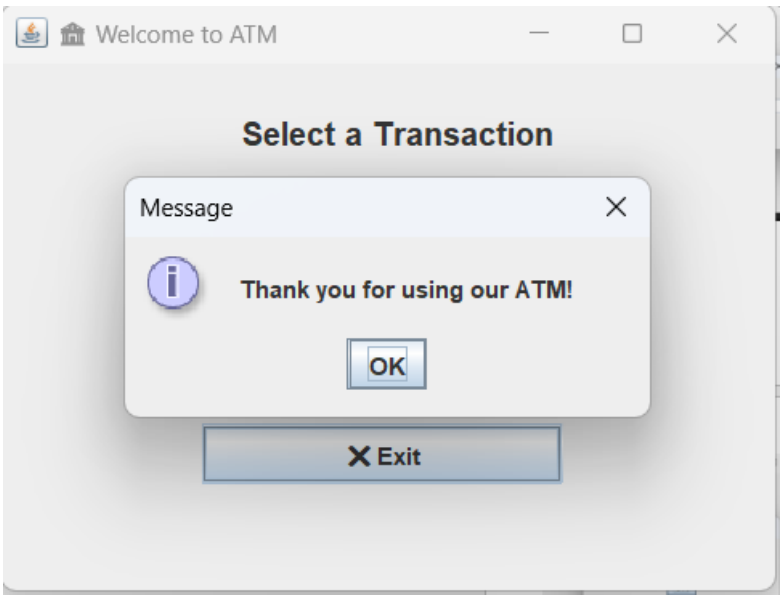

Welcome to ATM
Select a Transaction
Message
Deposited ₹1200.0
OK
Exit

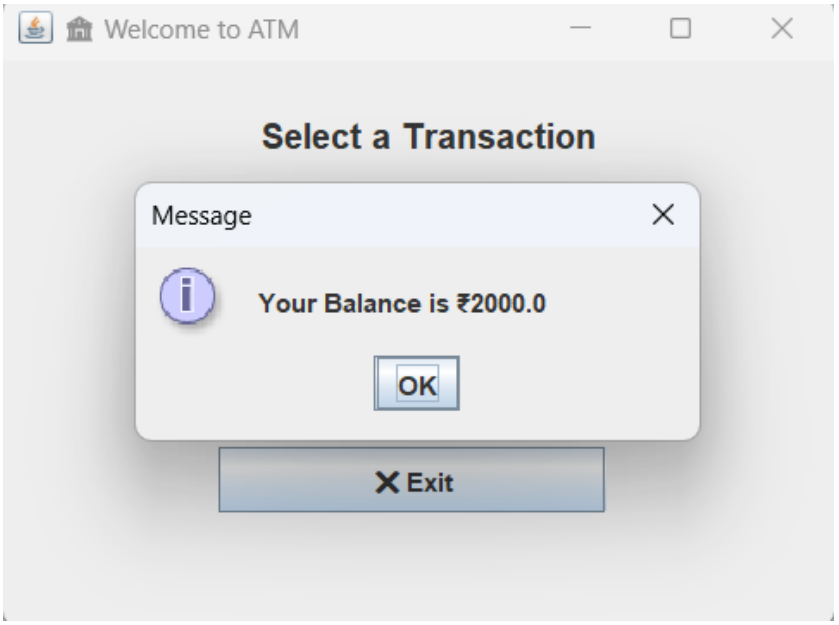

Welcome to ATM
Select a Transaction
Message
Your Balance is ₹2200.0
OK
Exit


Welcome to ATM
Select a Transaction
Check Balance
Deposit
Withdraw
Exit


Welcome to ATM
Select a Transaction
Input
Enter amount to withdraw:
200
OK
Cancel
Exit


Welcome to ATM
Select a Transaction
Message
Withdrew ₹200.0
OK
Exit


Welcome to ATM
Select a Transaction
Message
Thank you for using our ATM!
OK
Exit


Welcome to ATM
Select a Transaction
Message
Your Balance is ₹2000.0
OK
Exit