

ATM SIMULATION SYSTEM



A PROJECT REPORT

Submitted by

YAZHINI P (2303811710422185)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**ATM SIMULATION SYSTEM**” is the bonafide work of **YAZHINI P (2303811710422185)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr.M.A.MALARMANNAN A, M.E.,
ASSISTANT PROFESSOR

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 06/12/2024

CGB1201-JAVA PROGRAMMING
Mr. M. S. SIVANATHAN, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

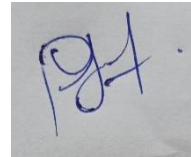
CGB1201-JAVA PROGRAMMING
Mr.R. KARTHIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**ATM SIMULATION SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



YAZHINI P

Place: Samayapuram

Date: 06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project focuses on simulating a basic Automated Teller Machine (ATM) system using Java to replicate essential banking operations. The system allows users to securely perform transactions such as withdrawing money, depositing funds, and checking account balances. By utilizing user authentication with an account number and PIN, the system ensures that only authorized users can access their accounts.

A menu-driven approach provides a simple and intuitive interface, enabling users to perform transactions seamlessly. Core functionalities are implemented using modular programming concepts, ensuring clarity and scalability. Real-time updates to account balances, along with meaningful feedback like warnings for insufficient funds and transaction confirmations, enhance user experience and reliability.

While the current implementation is console-based, the project serves as a foundation for future development. Enhancements such as integrating a graphical user interface (GUI), connecting to a database for persistent data storage, adding advanced security features like multi-factor authentication, and supporting additional banking operations can significantly improve its functionality. This project demonstrates the practical application of Java programming concepts, making it a valuable educational tool for understanding real-world banking systems.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
This project presents a console-based ATM simulation that enables users to perform essential banking transactions, including withdrawals, deposits, and balance inquiries. The system ensures secure access through authentication using an account number and PIN. It incorporates features like real-time balance updates, error handling for insufficient funds, and transaction confirmations. Designed with a modular code structure, the project offers adaptability for future enhancements. Serving as a practical educational tool, this simulation provides a foundational understanding of ATM operations using basic programming constructs and logical flow.	PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
2	PROJECT METHODOLOGY	3
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	MODULE DESCRIPTION	4
	3.1 User Authentication	4
	3.2 ATM Menu	4
	3.3 Withdraw	4
	3.4 Deposit	4
	3.5 Check Balance	4
	3.6 Exit	5
4	CONCLUSION & FUTURE SCOPE	6
	4.1 Conclusion	6
	4.2 Future Scope	6
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	12
	REFERENCE	16

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this project is to simulate a basic ATM system where users can securely perform essential banking operations, including withdrawals, deposits, and balance inquiries. The system aims to provide secure access through user authentication using an account number and PIN. It ensures real-time updates of account balances and provides meaningful feedback for transactions, such as insufficient balance warnings or transaction confirmations. Additionally, the project is designed to showcase the application of fundamental programming concepts, with a focus on modularity and scalability. It serves as both a functional ATM simulation and an educational tool for understanding core banking operations and basic software development practices.

1.2 Overview

This project focuses on simulating a basic ATM system to perform fundamental banking tasks such as withdrawals, deposits, and balance inquiries. The application is designed as a console-based program, emphasizing simplicity and ease of use. It ensures secure access through user authentication, requiring an account number and PIN for operation. The program handles real-time updates to account balances and provides feedback for each transaction, including warnings for insufficient funds. With its modular structure, the project serves as a foundation for future enhancements, such as integrating a database or developing a graphical user interface (GUI). Additionally, it is an educational tool, demonstrating core programming concepts and logical flow for understanding real-world ATM operations.

1.3 Java Programming Concepts

1.Object-Oriented Programming (OOP)

- **Encapsulation:** Grouping related data (e.g., account details, balance) and operations (e.g., withdraw, deposit) into a single entity like a BankAccount class. Example: The account number, PIN, and balance can be encapsulated within a class, and access can be controlled using methods (getters and setters).
- **Abstraction:** Simplifying complex functionalities, such as user authentication or balance checking, into distinct methods. Example: A authenticate() method to validate

user credentials, hiding the implementation details.

- **Inheritance:** If the project expands to include different types of accounts (e.g., Savings, Checking), a base Account class could be created, and specialized account types could inherit from it.
- **Polymorphism:** Implementing methods like performTransaction() that could behave differently depending on the type of transaction (withdrawal, deposit, or inquiry). This could be achieved through method overriding in subclasses.
- **Classes and Objects:**

Classes: BankAccount represents a user's account with attributes like account number, PIN, and balance, and methods for transactions. ATM represents the system handling user interaction and managing transactions.

Objects: A specific account (e.g., account1) is an object of the BankAccount class. The ATM instance (e.g., atm) is an object of the ATM class to interact with users.

2. Variables and Data Types

- **Integer Variables:** Store essential information like accountNumber, pin, balance, withdraw, and deposit. These ensure precise arithmetic calculations for transactions.
- **Scope Management:** Transaction-specific variables (withdraw and deposit) are declared inside the loop to optimize memory usage.

3. Control Structures

- **If-Else:** Validates the user's account number and PIN to ensure secure access. This mechanism prevents unauthorized users from accessing the system.
- **Switch-Case:** Handles menu operations such as Withdraw, Deposit, Check Balance, and Exit in an organized and readable way.
- **While Loop:** Repeatedly displays the ATM menu until the user chooses to exit, enabling multiple transactions in one session.

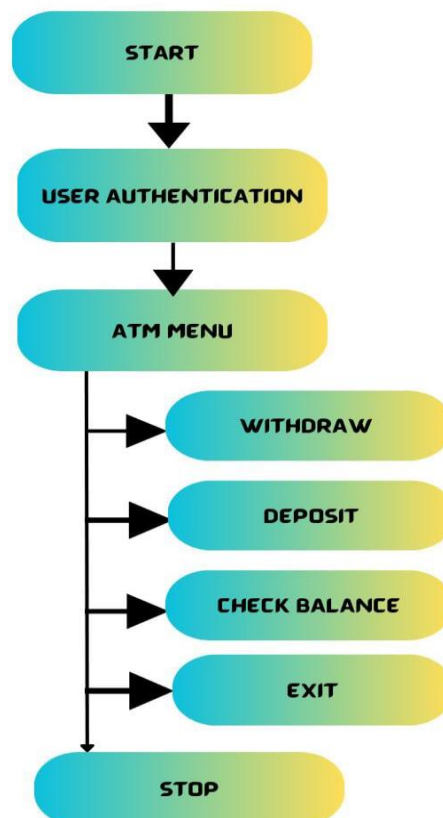
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work aims to enhance the current ATM simulation by incorporating advanced features and improving usability. One major enhancement is transitioning from a console-based interface to a Graphical User Interface (GUI) using JavaFX or Swing, making the system more user-friendly and visually appealing. Integrating the system with a database, such as MySQL, will allow persistent storage of account details, balances, and transaction history. Security features, including multi-factor authentication and encryption of sensitive data, will be added to ensure safer transactions. The system will also include additional functionalities, such as displaying transaction history and implementing other banking operations like fund transfers and bill payments. Improved error handling and input validation will ensure robust and reliable performance. Furthermore, a mobile app version of the system will be developed to allow users to perform transactions conveniently. These enhancements will transform the basic ATM simulation into a practical and secure system suitable for real-world applications.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Authentication

The system requires users to input their account number and PIN to proceed. These credentials are matched against predefined data to validate authenticity. If the entered credentials do not match, access is immediately denied, and the program terminates to protect user data. This mechanism ensures that unauthorized users cannot access the system, adding a critical layer of security.

3.2 ATM Menu:

Once authenticated, users are presented with a clear and straightforward menu displaying transaction options: Withdraw, Deposit, Check Balance, and Exit. The menu is designed for user convenience, allowing seamless navigation. A loop ensures users can perform multiple transactions without restarting the program, maintaining continuity and efficiency during their session.

3.3 Withdraw:

When withdrawing funds, the program verifies that the requested amount does not exceed the available balance. If sufficient, the amount is deducted, and a confirmation message is shown to the user. In case of insufficient funds, an error message informs the user about the issue, avoiding negative balances. This feature ensures financial accuracy while safeguarding the user's funds.

3.4 Deposit:

For deposits, users enter the desired amount to be added to their account. The program updates the balance instantly, reflecting the deposited amount. A confirmation message reassures the user of the transaction's success. This functionality is designed to ensure quick and accurate balance updates while maintaining user trust in the system.

3.5 Check Balance:

At any point during the session, users can select this option to view their current account balance. The balance shown reflects real-time changes, incorporating the effects of withdrawals and deposits made during the session. This feature offers users a transparent and up-to-date view of their financial status, promoting confidence in the system.

3.6 Exit:

The Exit option enables users to terminate the session securely. Upon choosing this option, the program confirms the end of the session and closes gracefully. This ensures a clean logout and prevents accidental misuse of the system after the session ends.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The ATM Simulation System successfully models a basic Automated Teller Machine system, offering users secure functionalities such as withdrawals, deposits, and balance inquiries. By employing core Java concepts like control structures, loops, input handling, and modular design, it replicates real-world banking operations efficiently. The system ensures security through user authentication and robust transaction handling, including sufficient balance checks and user feedback. While currently a console-based application, it serves as an educational tool for understanding structured programming. Future enhancements, including GUI integration, database connectivity, and advanced error handling, could enhance usability and functionality, making it a scalable foundation for more sophisticated systems. Furthermore, its modular design lays the groundwork for future enhancements, such as integrating advanced features like graphical user interfaces (GUIs), real-time database connectivity for dynamic account management, and explicit exception handling to manage invalid inputs gracefully. The project not only highlights the practical application of programming skills but also demonstrates the importance of building reliable, user-friendly, and scalable systems for real-world use cases.

4.2 FUTURE SCOPE

The future scope of this project involves several enhancements to improve functionality, usability, and scalability. Transitioning from a console-based application to a graphical user interface (GUI) will make interactions more intuitive and visually appealing for users. Integrating a database will enable dynamic account management, real-time updates, and secure storage of user data. To enhance security, encryption can be applied to sensitive information, such as account numbers and PINs, along with multi-factor authentication to prevent unauthorized access. Additional features, such as fund transfers, mini-statements, and loan application options, will make the system more comprehensive and practical for real-world use. Robust error handling mechanisms can be introduced to manage invalid inputs and runtime issues effectively, ensuring a seamless user experience. Multi-language support can expand accessibility, catering to users from diverse linguistic backgrounds. Finally, real-time connectivity with banking networks will allow live transaction updates system.

APPENDIX A

(SOURCE CODE)

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

public class ATMExampleAWT extends Frame implements ActionListener {
    // Initialize account details
    private int accountNumber = 123456;
    private int pin = 1234;
    private int balance = 100000;
    // UI components
    Label label1, label2, label3, messageLabel;
    TextField accountField, pinField, inputField;
    Button loginButton, withdrawButton, depositButton, checkBalanceButton,
    exitButton, confirmButton;
    Panel loginPanel, menuPanel, actionPanel;
    private String currentAction = ""; // To track if we are withdrawing or
    depositing

    public ATMExampleAWT() {
        // Set up the frame
        setTitle("ATM Example");
        setSize(400, 300);
        setLayout(new CardLayout());
        // Login panel
        loginPanel = new Panel(new GridLayout(4, 2));
        label1 = new Label("Account Number:");
```

```

label2 = new Label("PIN:");
accountField = new TextField();
pinField = new TextField();
pinField.setEchoChar('*');
loginButton = new Button("Login");
loginButton.addActionListener(this);
messageLabel = new Label("", Label.CENTER);
loginPanel.add(label1);
loginPanel.add(accountField);
loginPanel.add(label2);
loginPanel.add(pinField);
loginPanel.add(new Label(""));
loginPanel.add(loginButton);
loginPanel.add(messageLabel);
// Menu panel
menuPanel = new Panel(new GridLayout(5, 1));
withdrawButton = new Button("Withdraw");
depositButton = new Button("Deposit");
checkBalanceButton = new Button("Check Balance");
exitButton = new Button("Exit");
withdrawButton.addActionListener(this);
depositButton.addActionListener(this);
checkBalanceButton.addActionListener(this);
exitButton.addActionListener(this);
menuPanel.add(withdrawButton);
menuPanel.add(depositButton);
menuPanel.add(checkBalanceButton);
menuPanel.add(exitButton);

```

```

// Action panel for deposit/withdraw inputs
actionPanel = new Panel(new FlowLayout());
label3 = new Label("Enter Amount:");
inputField = new TextField(10);
confirmButton = new Button("Confirm");
confirmButton.addActionListener(this);
actionPanel.add(label3);
actionPanel.add(inputField);
actionPanel.add(confirmButton);
actionPanel.setVisible(false);

// Add panels to the frame
add(loginPanel);
add(menuPanel);
add(actionPanel);
menuPanel.setVisible(false);
setVisible(true);

// Handle window closing
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});
}

@Override
public void actionPerformed(ActionEvent e) {
    String action = e.getActionCommand();
    if (action.equals("Login")) {
        int enteredAccountNumber = Integer.parseInt(accountField.getText());
        int enteredPin = Integer.parseInt(pinField.getText());

```

```

    if (enteredAccountNumber == accountNumber && enteredPin == pin) {
        loginPanel.setVisible(false);
        menuPanel.setVisible(true);
    } else {
        messageLabel.setText("Invalid Account Number or PIN");
    }
} else if (action.equals("Withdraw")) {
    currentAction = "Withdraw";
    actionPanel.setVisible(true);
    menuPanel.setVisible(false);
    label3.setText("Enter Amount to Withdraw:");
    inputField.setText("");
} else if (action.equals("Deposit")) {
    currentAction = "Deposit";
    actionPanel.setVisible(true);
    menuPanel.setVisible(false);
    label3.setText("Enter Amount to Deposit:");
    inputField.setText("");
} else if (action.equals("Check Balance")) {
    showMessage("Balance: " + balance);
} else if (action.equals("Exit")) {
    System.exit(0);
} else if (action.equals("Confirm")) {
    int amount = Integer.parseInt(inputField.getText());
    if (currentAction.equals("Withdraw")) {
        if (balance >= amount) {
            balance -= amount;
            showMessage("Withdrawal successful. Collect your cash.");
        } else {

```

```

        showMessage("Insufficient Balance");
    }
    } else if (currentAction.equals("Deposit")) {
        balance += amount;
        showMessage("Deposit successful.");
    }
    // Reset to menu
    actionPanel.setVisible(false);
    menuPanel.setVisible(true);
}
}

```

```

private void showMessage(String message) {
    Dialog dialog = new Dialog(this, "Message", true);
    dialog.setLayout(new FlowLayout());
    Label messageLabel = new Label(message);
    Button okButton = new Button("OK");
    okButton.addActionListener(e -> dialog.setVisible(false));
    dialog.add(messageLabel);
    dialog.add(okButton);
    dialog.setSize(300, 100);
    dialog.setVisible(true);
}

```

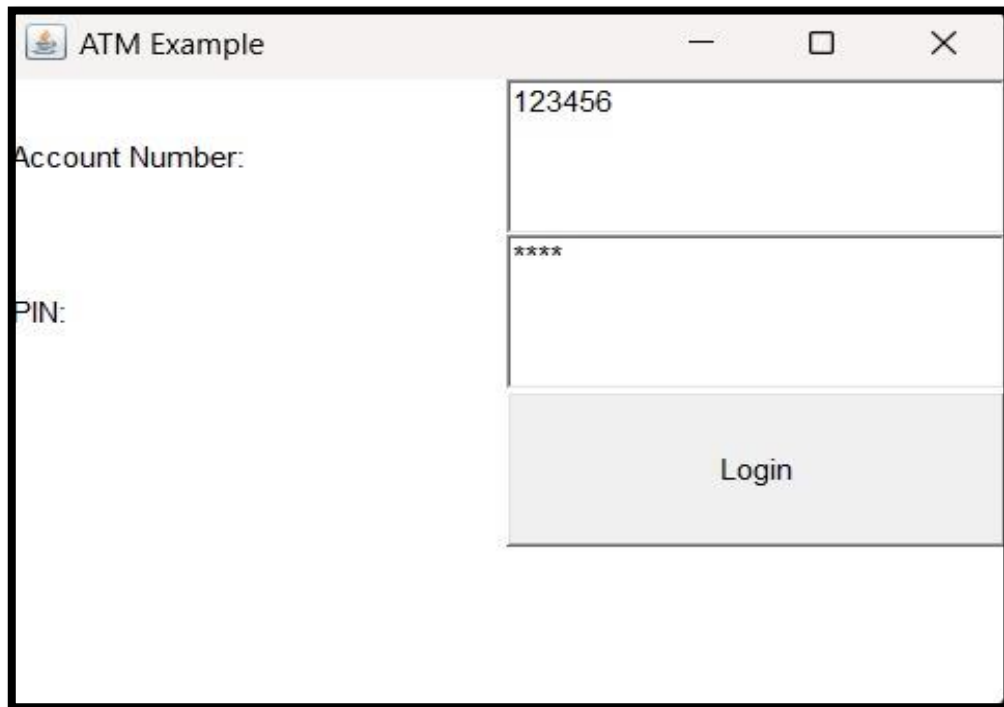
```

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

```

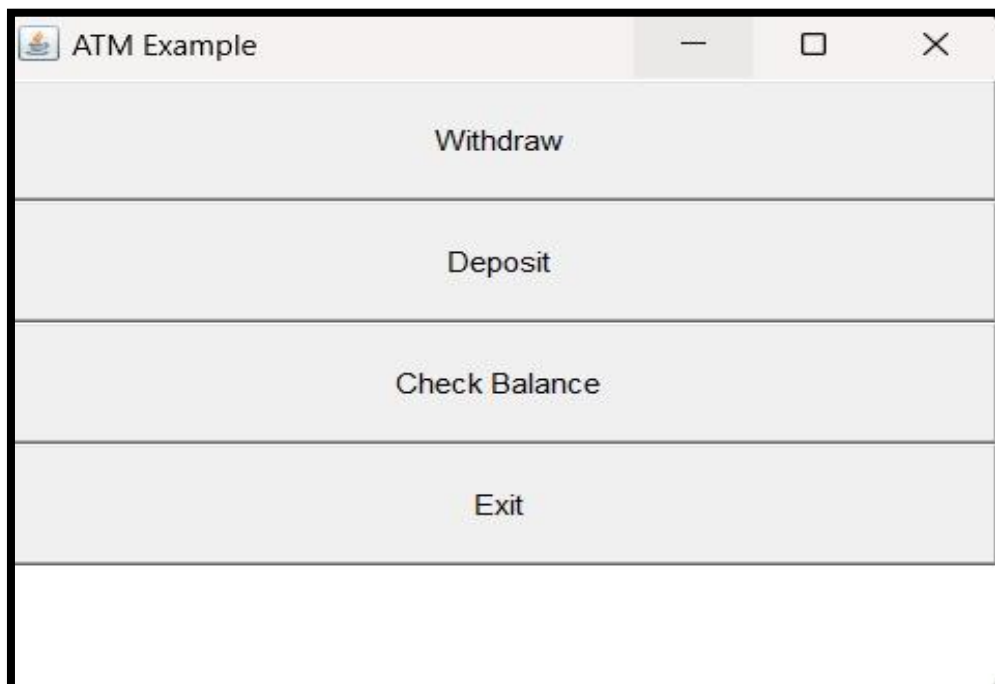
APPENDIX B (SCREENSHOTS)

1. User Authentication



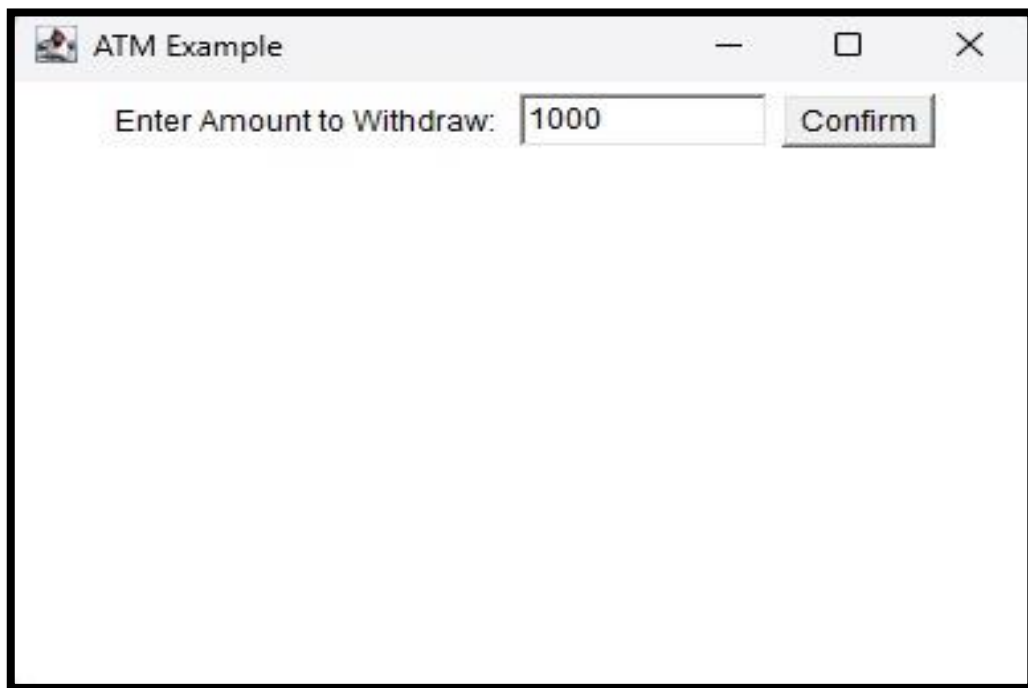
The screenshot shows a window titled "ATM Example" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form for user authentication. On the left side, there are two labels: "Account Number:" and "PIN:". To the right of these labels are two input fields. The first input field contains the text "123456". The second input field contains five asterisks "*****". Below these input fields is a large, light gray button labeled "Login".

2. ATM Menu

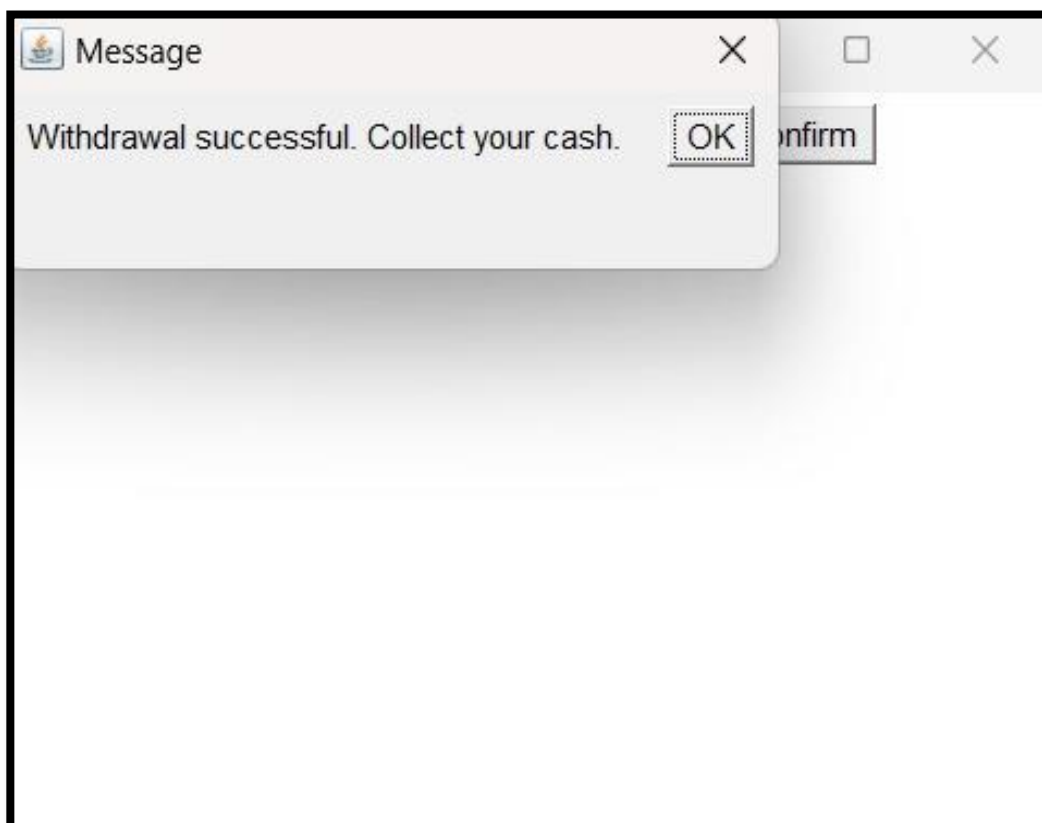


The screenshot shows a window titled "ATM Example" with a standard Windows title bar. The window displays a menu with four options, each in a separate light gray rectangular button. The options are: "Withdraw", "Deposit", "Check Balance", and "Exit". The buttons are stacked vertically and centered within the window.

3. Withdraw

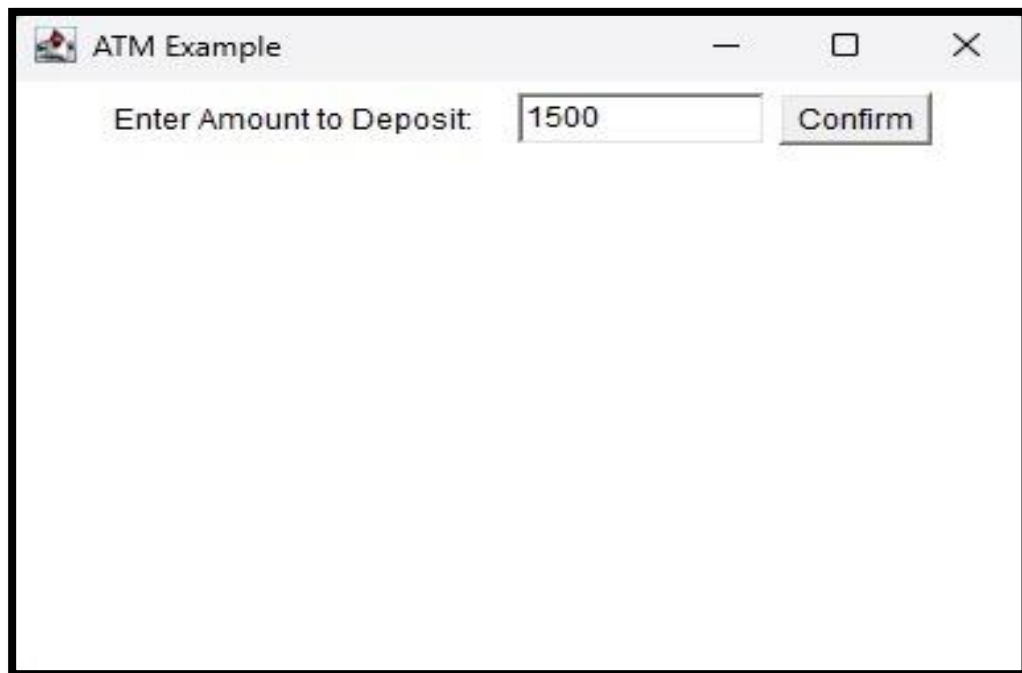


A screenshot of a Java Swing window titled "ATM Example". The window has a standard title bar with a minimize button, a maximize button, and a close button. Inside the window, there is a text label "Enter Amount to Withdraw:" followed by a text input field containing the value "1000". To the right of the input field is a button labeled "Confirm".

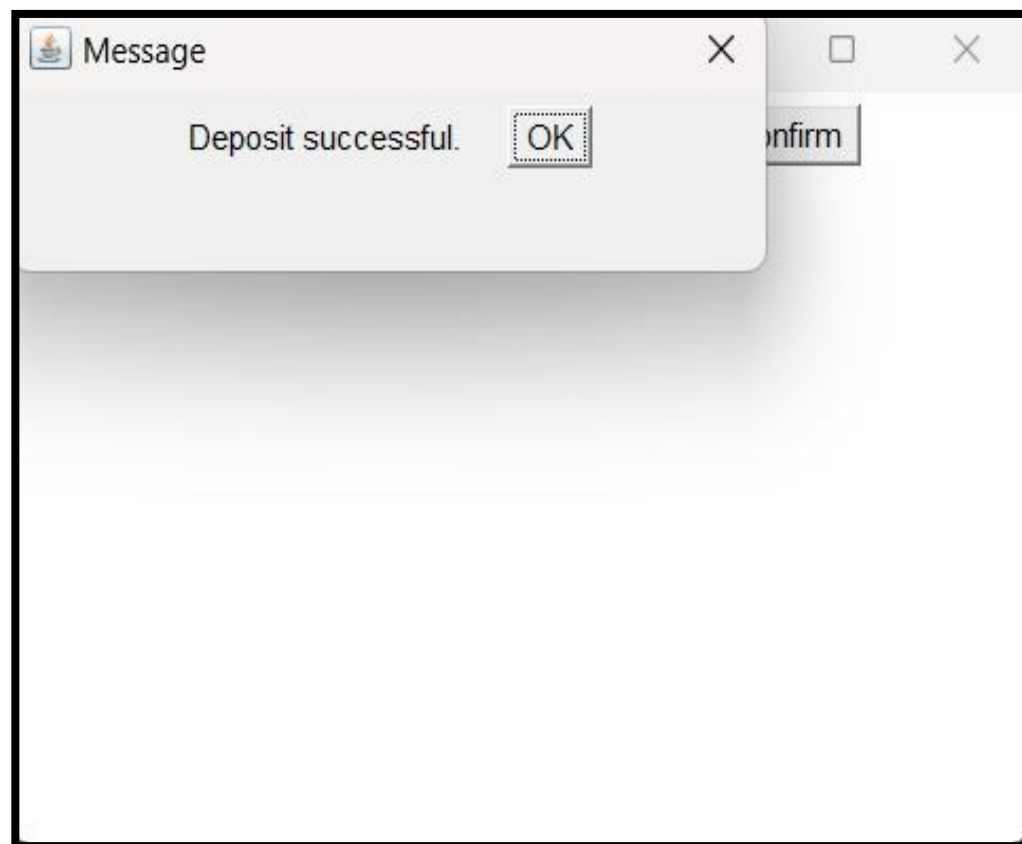


A screenshot showing a "Message" dialog box overlaid on the "ATM Example" window. The dialog box has a title bar with a close button. The message text inside the dialog is "Withdrawal successful. Collect your cash." To the right of the message are two buttons: "OK" and "Confirm". The "ATM Example" window is partially visible behind the dialog box.

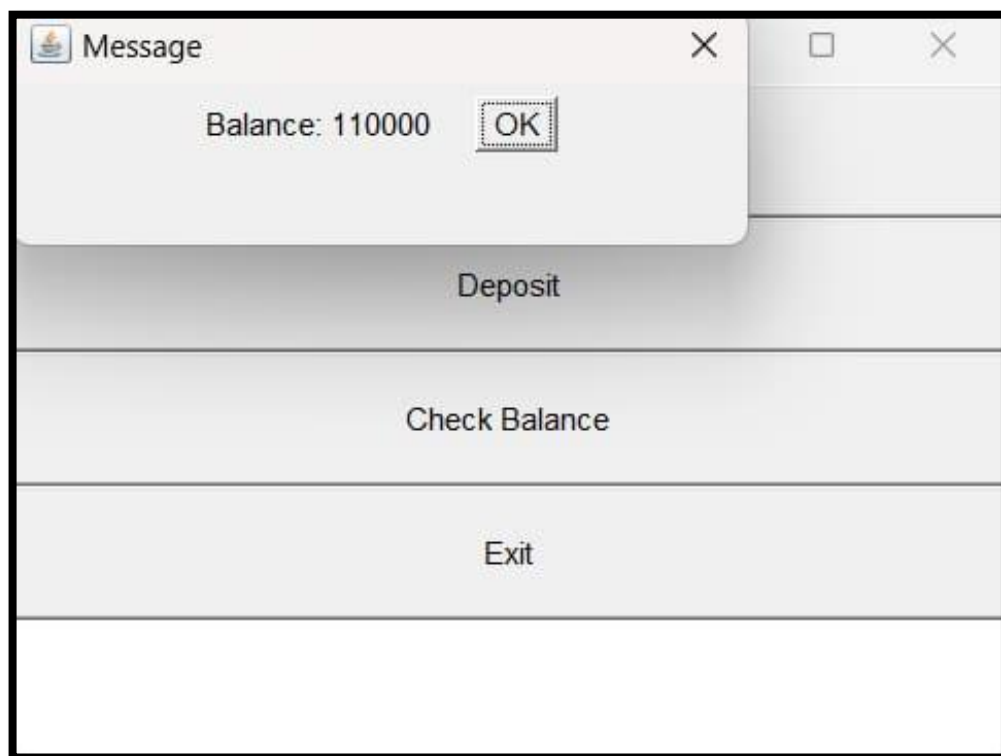
4. Deposit:



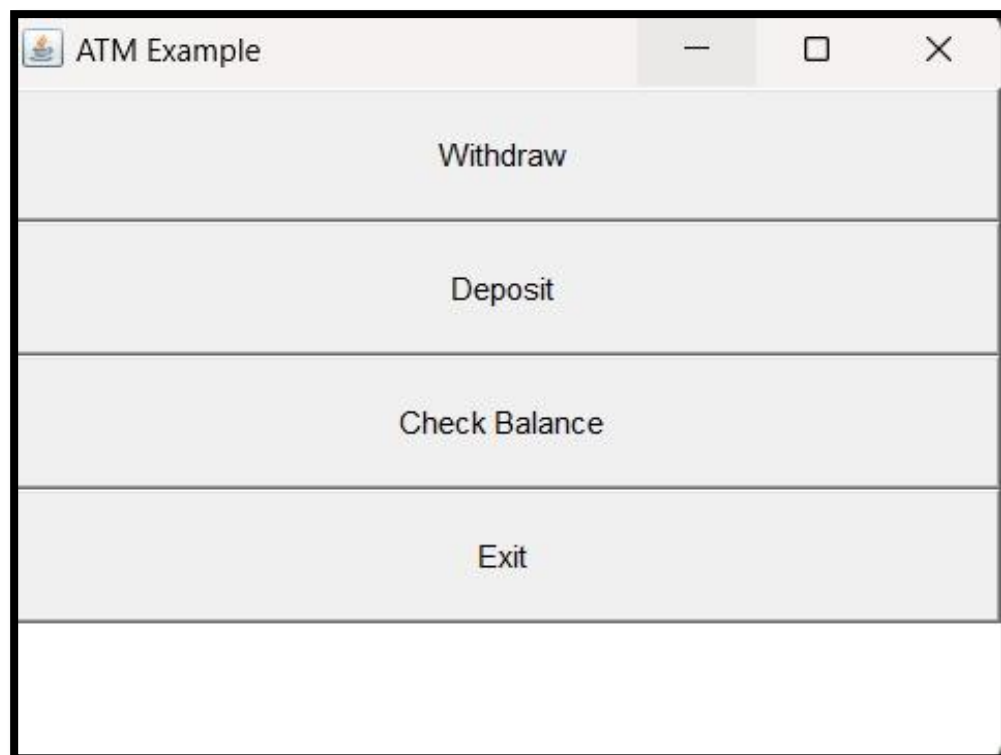
A screenshot of a Java Swing window titled "ATM Example". The window has a standard title bar with a minimize button, a maximize button, and a close button. Inside the window, there is a text label "Enter Amount to Deposit:" followed by a text input field containing the value "1500". To the right of the input field is a button labeled "Confirm".



5. Check Balance:



6. Exit:



REFERENCES

1. Aripionammal, S. and Natarajan, S. (1994) 'Transport Phenomena of Sm Sel – X Asx', *Pramana – Journal of Physics*, Vol.42, No.1, pp.421–425.
2. Shin, K.G. and McKay, N.D. (1984) 'Open Loop Minimum Time Control of Mechanical Manipulations and its Applications', *Proc. Amer. Contr. Conf.*, San Diego, CA, pp.1231–1236.
3. Oracle Corporation. (2023) 'Java Documentation: Scanner Class', Available at: <https://docs.oracle.com/javase/> (Accessed: 30 November 2024).