SIMPLE BANKING APPLICATION

A PROJECT REPORT

Submitted by

GANGAISELVAN C(2303811710421302)

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 **"SIMPLE BANKING APPLICATION"** is the bonafide work of **GANGAISELVAN C (2303811710421302)** 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

[illegible]

SIGNATURE

Dr.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
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 03/12/2024

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

[illegible]

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Ms. R.ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“SIMPLE BANKING APPLICATION”** 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

[illegible]

GANGAISELVAN

Place: Samayapuram

Date: 03/12/2024

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 **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,),** 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

A **simple banking application** designed to streamline basic banking operations for users and administrators. The application offers an intuitive interface and fundamental functionalities such as account creation, balance inquiries, fund transfers, and transaction history. It caters to both individual users and financial institutions seeking a lightweight, scalable solution. The application is built using modern programming frameworks to ensure security, reliability, and performance. Core features include user authentication, secure data storage, and role-based access control. Additionally, the system employs basic encryption mechanisms to safeguard sensitive information.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
A Simple Banking Java application is a basic program built in Java to simulate core banking operations. It provides fundamental functionalities such as creating bank accounts, checking account balances, transferring funds, and viewing transaction history. The application typically includes a user interface (CLI or GUI) for interaction, authentication for security, and data storage (either in-memory or in a simple database).	**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 1

INTRODUCTION

1.1Objective

The objective of this project is to develop a Simple Banking Application that allows users to perform basic banking operations such as depositing funds, withdrawing funds, checking account balances, and viewing transaction histories. The application is designed to provide a user-friendly interface for financial transactions, ensuring data security and accuracy. It serves as a foundational system to demonstrate the integration of Java programming concepts in solving real-world problems.

1.2Overview

The Simple Banking Application is a console-based program that mimics basic functionalities of a banking system.

Key Features:

- Account Creation: Users can set up a new account with a name and account number.
- Deposit and Withdrawal: Transactions can be made with input validation for proper handling of amounts.
- Balance Inquiry: The current balance is always accessible to users.
- Transaction History: A detailed list of all previous transactions is maintained.

Exit Option: Users can safely terminate the session.

1.3 Java Programming Concepts

1. Object-Oriented Programming (OOP)

The Account class represents a bank account, bundling related data (like account holder name and balance) and actions (like depositing or withdrawing money).

2. Control Structures

A while loop is used to repeatedly show the menu until the user exits.

3. Dynamic Data Storage

Keeps track of the transaction history, allowing storage and retrieval of transaction details easily

CHAPTER 2
PROJECT METHODOLOGY

2.1Proposed Work

The proposed work involves the development of a Simple Banking Application that simulates basic banking functionalities for users. The application is designed to handle core banking tasks such as depositing funds, withdrawing funds, viewing account balance, and transaction history.

The main features of the application include:

1. **Account Creation**: Users can set up a new account by providing their name and account number. Once created, users will have access to the basic banking functions.

2. **Deposit and Withdrawal Functionality**:

   - Users can deposit money into their account.

   - Users can withdraw funds from their account, with appropriate checks for **sufficient balance**.

3. **Balance Inquiry**: Users can check their current account balance at any time.

4. **Transaction History**: The application will maintain a list of transactions, enabling users to view all deposits and withdrawals made to their account.

5. **User-friendly Interface**: A console-based menu-driven interface

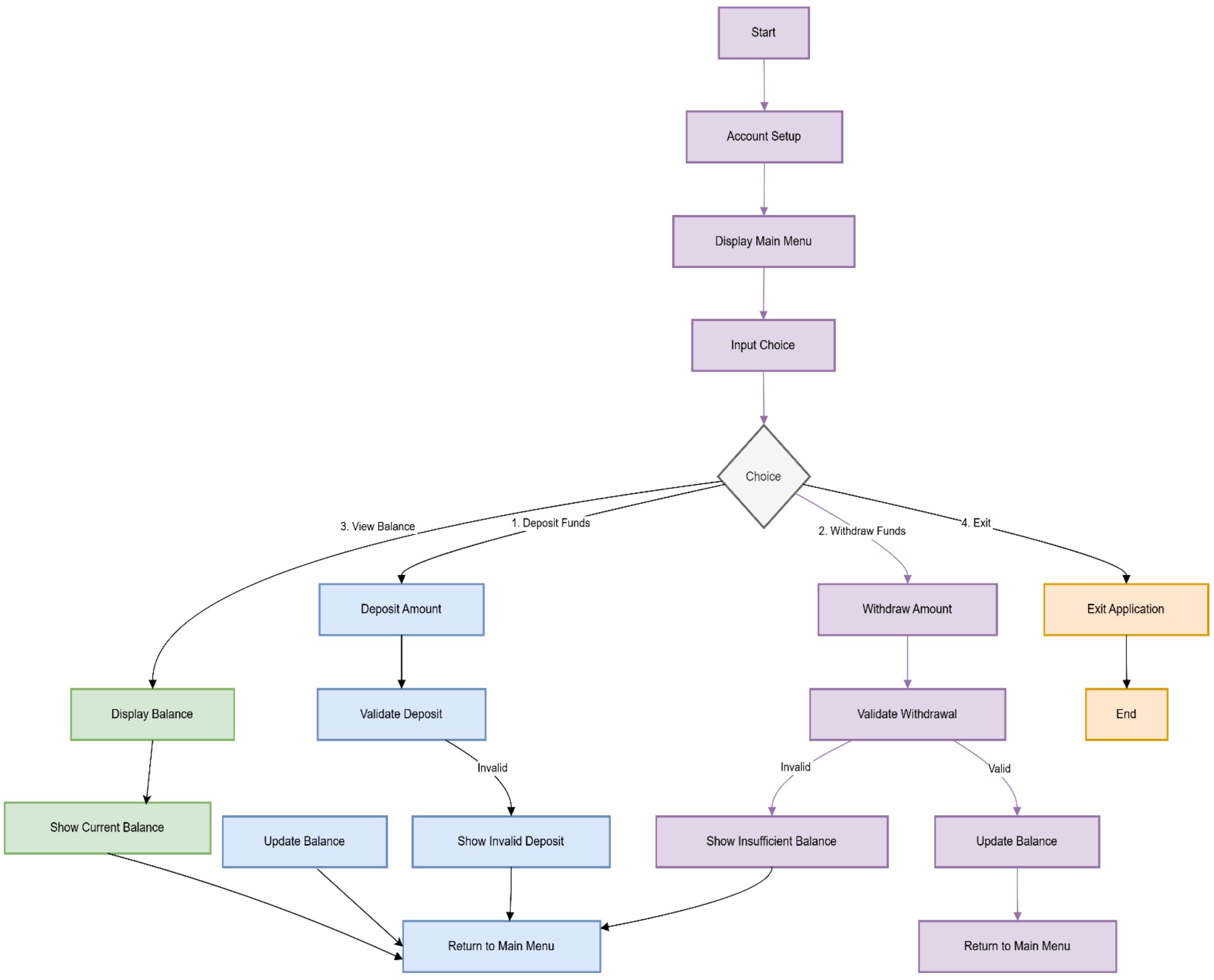
allows users to navigate through different banking operations with ease.

Key Goals of the Project:

- To provide a secure, modular banking application where all operations (deposit, withdrawal, balance checking, transaction history) are handled by separate, reusable modules.

- The use of **Java classes, methods**, and **collections** for managing banking operations and user.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Navigation Module:

Purpose: Provides users with an interface to navigate between different functionalities like deposits, withdrawals, reservations, or viewing details.

Features:

- Displays menu options.
- Handles user input to select an action.

3.2 Customer Details Module:

Purpose: Collects and manages customer information such as name, contact details, or account number.

Features:

- Stores customer data securely.
- Retrieves data when required for bookings or transactions.

3.3 Hotel Booking Module:

Purpose: Enables users to make hotel reservations by selecting available options and providing required details.

Features:

- Displays available hotels and rooms.
- Handles booking requests and updates availability status.

3.4 Reservation Module:

Purpose: Manages all reservations and transactions, including customer payments or cancellations.

- Features:

  Tracks bookings and payment statuses.

  Updates records after each transaction.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Simple Banking Application project demonstrates the core functionality and basic principles of a banking system implemented in Java. The application provides a simple user interface where users can perform essential operations, such as depositing and withdrawing funds, checking their account balance, and viewing transaction history. The main goal of the project is to help users understand basic banking operations and gain hands-on experience with key Java programming concepts such as object-oriented programming (OOP), input validation, and data management. Through this project, users can see how class structures and methods are used to encapsulate account details, handle transactions, and maintain a history of activities. While this application is simple, it lays the groundwork for more advanced banking systems that could include features like multiple user accounts, secure login systems, and interest rate calculations

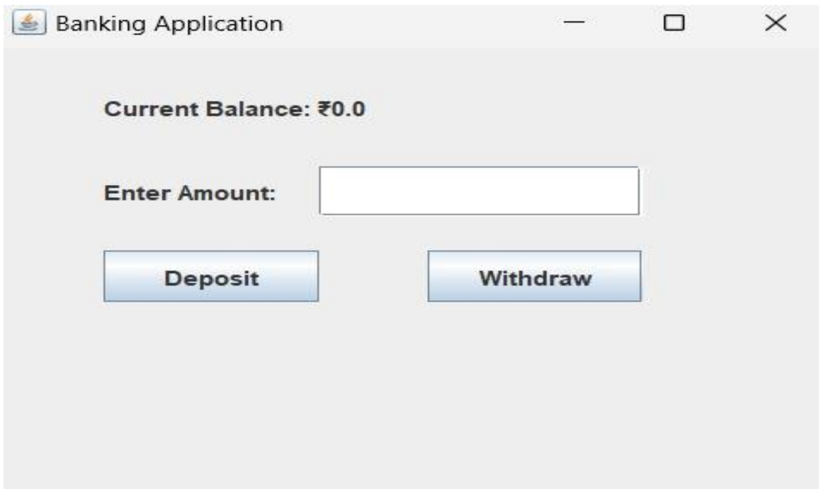
4.2 FUTURE SCOPE

A simple banking application has significant future scope as it can evolve into a robust financial tool by incorporating features like personalized financial insights, automated budgeting, and multi-currency support. Enhanced security measures, such as biometric authentication and AI-powered fraud detection, can improve user trust, while advanced technologies like blockchain and AI chatbots can streamline operations. Integration with investment platforms, insurance services, and tax reporting tools can make the app a one-stop financial solution. Additionally, scalability through cloud-based architecture and compatibility with multiple devices ensures growth potential, like green banking and financial literacy tools can enhance its appeal to a broader audience.

APPENDIX A
(SOURCE CODE)

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

public class BankingApp {
    private double balance = 0;

    public BankingApp() {
        JFrame frame = new JFrame("Banking Application");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);

        JLabel balanceLabel = new JLabel("Current Balance: ₹" + balance);
        balanceLabel.setBounds(50, 20, 300, 30);
        frame.add(balanceLabel);

        JLabel amountLabel = new JLabel("Enter Amount:");
        amountLabel.setBounds(50, 70, 100, 30);
        frame.add(amountLabel);

        JTextField amountField = new JTextField();
        amountField.setBounds(150, 70, 150, 30);
        frame.add(amountField);

        JButton depositButton = new JButton("Deposit");
        depositButton.setBounds(50, 120, 100, 30);
        frame.add(depositButton);

        JButton withdrawButton = new JButton("Withdraw");
        withdrawButton.setBounds(200, 120, 100, 30);
        frame.add(withdrawButton);
        depositButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    double amount = Double.parseDouble(amountField.getText());
                    if (amount > 0) {
                        balance += amount;
                        balanceLabel.setText("Current Balance: ₹" + balance);
                        JOptionPane.showMessageDialog(frame, "₹" + amount + " deposited 
successfully!");
                    } else {
                        JOptionPane.showMessageDialog(frame, "Amount must be 
positive!");
                    }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Invalid input! Please enter a 
valid amount.");
                }
            }
        });

        withdrawButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    double amount = Double.parseDouble(amountField.getText());
                    if (amount > 0 && amount <= balance) {
                        balance -= amount;
                        balanceLabel.setText("Current Balance: ₹" + balance);
                        JOptionPane.showMessageDialog(frame, "₹" + amount + " withdrawn 
successfully!");
                    } else if (amount > balance) {
                        JOptionPane.showMessageDialog(frame, "Insufficient balance!");
                    } else {
                        JOptionPane.showMessageDialog(frame, "Amount must be 
positive!");
                    }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Invalid input! Please enter a 
valid amount.");
                }
            }
        });

        frame.setVisible(true);
    }

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

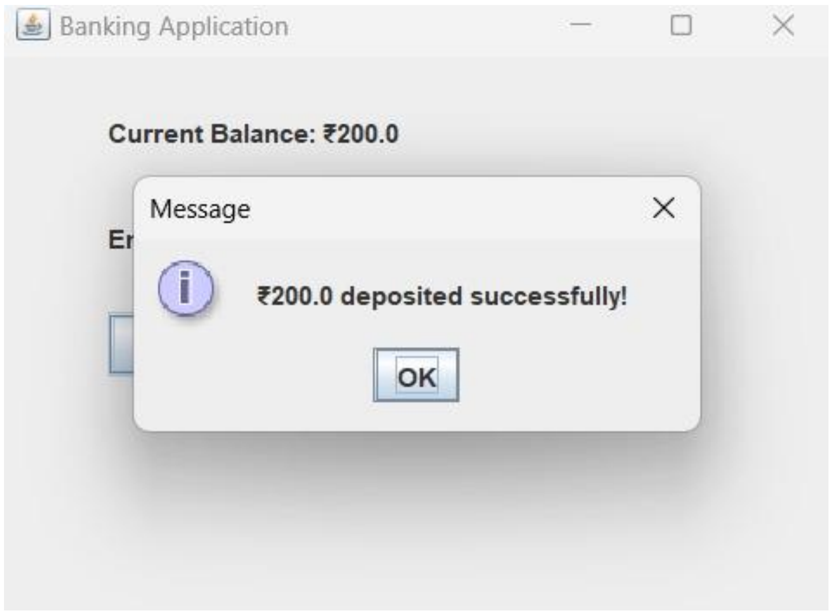
APPENDIX B
(SCREENSHOTS)


Banking Application

Current Balance: ₹0.0

Enter Amount:

DEPOSIT

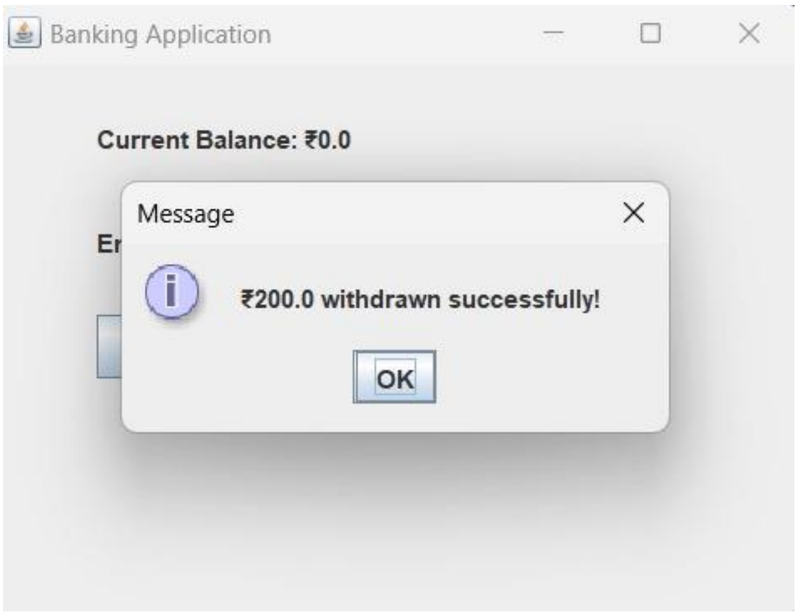

Banking Application

Current Balance: ₹200.0

Message

₹200.0 deposited successfully!

WITHDRAW



Current Balance: ₹0.0

₹200.0 withdrawn successfully!

REFERENCES

Oracle Java Documentation

- https://docs.oracle.com/javase/tutorial/
  A comprehensive resource for learning Java programming concepts, from basic syntax to advanced object-oriented techniques.
-

  □ **"Introduction to Java Programming" by Daniel Liang**

- A detailed textbook that provides step-by-step guidance on Java programming, focusing on essential topics like classes, methods, arrays, and object-oriented principles.
-

  □ **"Java: How to Program" by Paul Deitel and Harvey Deitel**

- A thorough book that covers Java fundamentals with practical examples, ideal for beginners and intermediate programmers.
-

  □ **Java Code Geeks - Banking System in Java**

- https://www.javacodegeeks.com
  A site with tutorials, examples, and code snippets for Java developers, including articles on building simple banking systems.