Project Report

Simple Banking Application

Project report by:
Swaraj Sonawane
Gopal Patil
Kuntal Dive

1. Introduction

The aim of the Simple Banking Application project is to learn practical skills in Java programming through real-life scenarios for those who don’t have good prior knowledge. Through this project, we can engage in fundamental banking tasks such as depositing money, withdrawing funds, and checking account balances. By implementing strong fundamentals techniques through which we will create a user interface which will include programming concepts like text manipulation, iterative actions, code modularization for reusability, and conditional decision-making. Additionally, individuals with less/low coding experience can acquire skills while building this project.

2. Project Overview

The project essentially operates on a Java program that functions as a simplified version of a bank. Within this, there's a class named SimpleBankingApplication, which contains all the tools needed to perform banking tasks like depositing money into your account, withdrawing funds, and checking your account balance. While using the program, you are offered various options on the screen that give you choices for what you want to do with your account. You can pick from these options to manage your money however you like.

3. Implementation

Initialization: Firstly, the program will ask the user for an input for the given possible options. Users are required to invest their balances accurately to ensure accurate results and efficient use of program resources. Once the first balance is paid, users can seamlessly proceed with their wanted banking activity or simulation.

Menu- By making a simple user interface, users are shown with a menu of banking functions. This menu acts as a search tool, allowing users to select their preferred action. With clear options, users can easily navigate the program to make deposits, withdrawals, balance inquiries, and transfers. This intuitive interface enhances the communication between the program and the user with the system.

Deposit: To deposit money, the user simply enters the desired amount in a designated area. This simplified design allows for faster and more efficient communication. Once the amount is registered, users can confirm the deposit, ensuring its authenticity and security. This user-friendly approach makes saving money a seamless experience in the banking system.

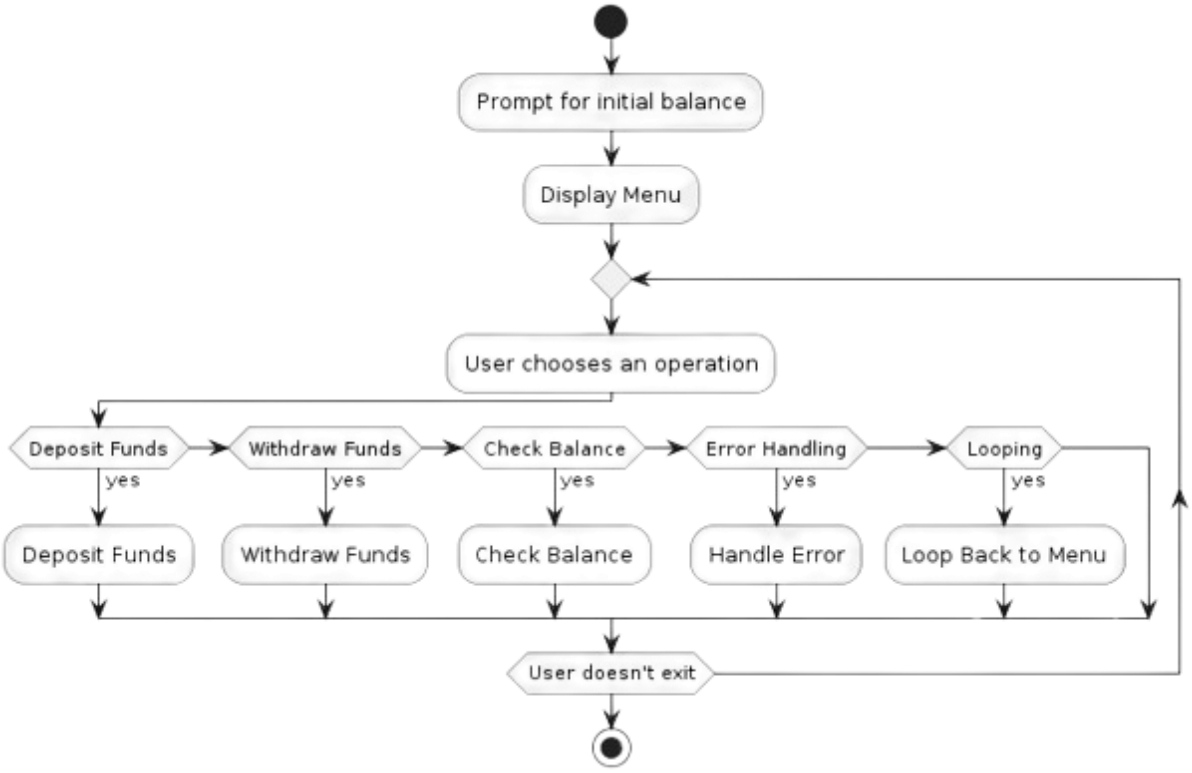
Withdrawal: Users can withdraw funds from their accounts, if sufficient funds are available. This creates financial stability and prevents overdrafts. Once a checkout is requested, users can proceed with the transaction, ensuring a smooth and secure transaction. By following this process, users can better manage their finances within the banking system.

Check Balance: As the title says, basically a user have the ability to check their account balance at any time. This gives them the ability to gain real-time insight into their financial situation. By providing this feature, users can manage their transactions and manage their money with confidence. Simple and convenient.

Error Handling: The system is designed to handle invalid deposits and immediately notify users of insufficient funds at the time of withdrawal. This approach improves communication efficiency and prevents communication errors. By alerting users to such issues, the program contributes to consistency and transparency, improving the overall user experience in the banking system.

Looping: A loop is used to continuously display the menu options until the user chooses to exit. This ensures that users have plenty of time to navigate through available banking functions at their own pace. By providing a consistent menu interface, the program facilitates a simple and easy user experience, increasing user satisfaction and ease of use.

4. Flowchart



5. Future Enhancements

User Authentication: Using user authentication increases account security by requiring users to verify themselves before logging into an account. This approach helps prevent access and protects sensitive financial information. By adding the features of authentication such as passwords, biometrics, or two-factor authentication, the program ensures that only authorized users can access their accounts, enforcing security strength of all systems.

Transaction History: Users improve their banking experience by having access to their transaction history. This feature provides users with a complete record of past transactions, for them to track financial activity. With easy access to transaction history within the program, the user can make proper financial decisions and not have to worry about the transactions he/she has been making.

Multiple Accounts: Allowing users to create and manage multiple accounts within the system increases their banking flexibility. This feature empowers users to better organize their finances, separating funds for

different purposes or organizations. Offering the ability to create and manage multiple accounts, the program delivers advanced flexibility and control that meets a variety of financial needs and preferences.

GUI Integration: Develop a graphical user interface for improved user experience.

6. Project Shortcomings

While the Simple Banking Application serves as a beneficial learning tool, it also possesses certain limitations:

- Lack of security features
- Incomplete input validation
- Assumption of a single user environment
- Absence of data persistence mechanism
- Potential scalability issues
- Limited error handling
- Insufficient documentation

7. Conclusion

The Simple Banking Application project provides an easy introduction to Java programming. By making a simple program that includes easy, yet important/core concepts of java programming which individuals with not much coding experience can also understand and analyse. This project will help those who are excited to learn the language but don’t know how to get it up and running.