A Minor Project Synopsis on

**BANK MANAGEMENT SYSTEM**

Submitted to Manipal University, Jaipur

Towards the partial fulfilment of the Award of the Degree of

**BACHELOR OF TECHNOLOGY**

In Computers Science and Engineering

2022-2023

By:

Name: Ojasvi Padhiar

Registration Number:229301684

Name: Vaibhav Sharma

Registration Number:229301728

A close up of a sign

Description automatically generated

Under the guidance of

Mr. Mayank Kumar Jain

**Department of Computer Science and Engineering**

**School of Computer Science and Engineering**

**Manipal University Jaipur**

**Jaipur, Rajasthan**

**INTRODUCTION**

Java is an object-oriented language that implements Object-Oriented Programming (OOP) concepts. OOP allows organizing code by creating objects that encapsulate data and behaviors. It promotes code reusability, modularity, and maintainability.

One of the fundamental principles of OOP is the concept of classes and objects. A class is a blueprint that defines the structure and behavior of an object, while an object is an instance of a class that represents a specific entity or concept.

Java supports encapsulation, inheritance, polymorphism, and abstraction. Encapsulation allows you to hide the internal details of an object and provide a public interface to interact with it. Inheritance enables you to create new classes based on existing classes, promoting code reuse and establishing a hierarchical relationship between classes. Polymorphism allows objects of different types to be treated as objects of a common superclass. Abstraction involves creating abstract classes and interfaces to define common characteristics and behaviors that can be shared among multiple classes.

Encapsulation is the process of bundling data and methods together within a class. It provides data hiding, allowing you to protect the internal state of an object from direct access and modification. By encapsulating data and exposing it through methods, you can enforce data integrity and maintain control over the object's behavior.

By leveraging these OOP concepts in Java, you can write more organized, modular, and extensible code. These principles help in building scalable and maintainable applications, making Java a powerful language for software development.

MOTIVATION

A Bank Management System is a project designed to address the challenges faced by banks and financial institutions. It aims to streamline operations, improve services, ensure data integrity, mitigate risks, provide comprehensive reports, and accommodate future growth. Implementing a Bank Management System can significantly enhance efficiency, accuracy, customer satisfaction, risk management, and scalability.

One of the most significant benefits of a Bank Management System is the streamlined workflow it provides. By automating various processes such as account opening, loan approvals, transaction monitoring, and customer onboarding, it reduces manual efforts, eliminates paperwork, and improves overall operational efficiency.

Another advantage of a Bank Management System is the enhanced customer service it offers. Banks can provide better customer service by offering features such as 24/7 access to accounts, personalized banking experiences, quick query resolution, and self-service options. This leads to improved customer satisfaction and loyalty, ultimately resulting in a positive reputation for the bank.

Objectives of the Project:

Here are some pros and cons of the basic bank management system described earlier:

Pros:

- The system provides a foundation for a comprehensive banking system that can be expanded upon and improved to meet specific requirements.

- It handles various account types, such as Savings Accounts, Current Accounts, Fixed Deposits, and Systematic Investment Plans, each with unique features and methods.

- The program includes basic error handling, manages multiple accounts, and allows users to display general bank information.

- Input is taken via the console, and menus are shown to take user input for different operations.

Cons:

- The system is basic and lacks advanced functionalities, such as transferring funds between accounts, checking transaction history, and authentication and security measures.

- The user interface is not very user-friendly, as it relies on console input and output.

- The error handling mechanisms are basic and may not be robust enough to handle potential exceptions adequately.

- The interest calculation methods are not very advanced and may not be tailored to specific account types.

Overall, the basic bank management system provides a good foundation for a comprehensive banking system, but there are several areas where it can be improved and expanded upon to meet specific requirements and provide a better user experience.

Please find below a summary of the pros and cons of each method used in the banking system.

General Cons for Methods:

1. Direct Console Output:
   1. The methods directly print messages to the console. This restricts flexibility to modify output mechanisms, such as writing to logs or GUI displays.
2. Limited Return Information:
   1. The methods perform actions but provide limited return information. For example, deposit and withdrawal methods only print messages, which might not be suitable for all use cases that require more detailed feedback.
3. Override for Specific Types:
   1. Specific display methods are overridden in subclasses, which may lead to maintenance issues when scaling or altering the system. Changes might require adjustments in multiple overridden methods, impacting code maintainability.

General Pros for Methods:

1. Clear Functionality:
   1. Each method serves a specific purpose, providing clarity about the actions they perform for each account type.
2. Validations:
   1. The deposit and withdrawal methods include basic validations to prevent invalid amounts or exceeding account balances, enhancing the system's reliability.
3. Structured Output:
   1. The display methods offer structured information tailored to each account type, providing a clear overview of account specifics.

To improve the methods, we can decouple the output from the methods, return status or error codes, and reconsider method overrides for better code maintenance and extensibility.

|  |  |  |
| --- | --- | --- |
| Methods | Pros | Cons |
| * `Account.deposit(double amount)` | * + Allows adding funds to the account.   + Validates and prevents negative deposits. | * Printing a message directly instead of returning a status, limits flexibility. |
| * Account.withdraw(double amount)` | * Enables withdrawing from the account. * Validates the withdrawal amount and balance. | * Printing a message directly instead of returning a status, which limits flexibility. |
| * Account.displayAccountInfo() | * Displays account details, including type, number, holder, and balance. | * Prints directly to the console, which restricts flexibility for output handling. |
| * SavingsAccount.displayAccountInfo() | * + Displays specialized account info, including interest rate. | * + Method overridden for specific type, which may cause maintenance issues in the future. |
| * CurrentAccount.displayAccountInfo() | * Displays specialized account info, including interest rate and maturity period. | * Method overridden for specific type, which may cause maintenance issues in the future. |
| * FixedDeposit.displayAccountInfo() | * Displays specialized account info, including interest rate and maturity period. | * + Method overridden for specific type, which may cause maintenance issues in the future. |
| * SystematicInvestmentPlan.display * AccountInfo() | * Displays specialized account info, including return rate and investment period. | * Displays specialized account info, including return rate and investment period. |

Methodology

The "Bank Management System" code is a Java-based application that models a basic banking system. It follows an object-oriented design with classes representing different aspects of the banking system. Here is a synopsis of the code:

1. Object-Oriented Design: The code follows an object-oriented design with classes representing different aspects of the banking system.
2. Account Class: The `Account` class serves as the base class for all account types. It includes fields for the account number, account holder's name, and balance. The `Account` class provides methods for depositing and withdrawing funds, checking balances, and displaying account information.
3. SavingsAccount and CurrentAccount: The `SavingsAccount` and `CurrentAccount` classes inherit from the `Account` class. They represent specific types of bank accounts. `SavingsAccount` includes an additional field for the interest rate, while `CurrentAccount` includes the overdraft limit. Both classes override the `withdraw` method to implement their specific rules.
4. BankManagementSystem Class: The main class contains the program's entry point in the `main` method. It drives the entire application.
5. Menu-Driven Interface: The code provides a user-friendly menu where users can choose various banking operations. These operations include creating savings or current accounts, depositing money, withdrawing money, displaying account information, and viewing bank-related information.
6. Dynamic Account Management: The code maintains an array of account objects, allowing the dynamic creation of accounts. Each account is assigned a unique account number, starting from 1001.
7. User Input Handling: The `Scanner` class is used to handle user input for various operations, making the application interactive.
8. Data Validation: The code includes data validation checks to ensure that operations like deposits and withdrawals are performed with appropriate input values and within the account's limits.
9. Bank Information Display: Users can access the bank's information, including the bank name (SBI) and the number of clients (accounts created).

Overall, the "Bank Management System" code demonstrates object-oriented programming principles, inheritance, user interaction, and simple account management for a bank. Users can create different types of accounts, manage funds, and retrieve account and bank-related information using a user-friendly menu-driven interface.

Facilities required for proposed work:

* Java
* OOPS concepts using Java
* VS code / IntelliJ / Eclipse

Bibliography

* H. Schildt, The Complete Reference Java, (10e), Oracle Press, 2018.
* Arnold K, & Gosling J, “The Java Programming Language”, 2002.
* https://www.geeksforgeeks.org/mini-banking-application-in-java/