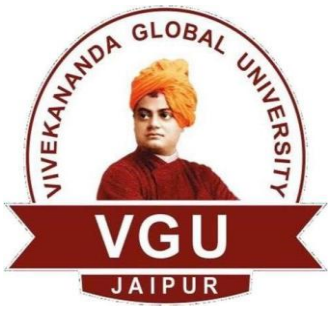

VIVEKANANDA GLOBAL UNIVERSITY

Master of Computer Applications

Object Oriented Programming Using JAVA (PGCSA111)

Build a Basic Bank Account System with Deposit and Withdrawal

Project Report

PROJECT GUIDE:
Mr. Narayan Vyas

SUBMITTED BY:

GAURAV SINGH	24CSA3BC036
GAURAV SHARMA	24CSA3BC004
FAIYYAZ KHAN	24CSA3BC105
HARSH KUMAR	24CSA3BC064

ACKNOWLEDEGEMENT

I have taken this opportunity to express my gratitude and humble regards to the Vivekananda Global University to provide an opportunity to present a project on the "Build a Basic Bank Account System with Deposit and Withdrawal".

I would also be thankful to my project guide Mr. Narayan Vyas to help me in the completion of my project and the documentation. I have taken efforts in this project, but the success of this project would not be possible without their support and encouragement.

I would like to thanks our dean sir "Dr. R C Tripathi" to help us in providing all the necessary books and other stuffs as and when required. I show my gratitude to the authors whose books has been proved as the guide in the completion of my project I am also thankful to my classmates and friends who have encouraged me in the course of completion of the project.

Thanks

Place: Jaipur
Date: 29-03-2025

DECLARATION

We hereby declare that this Project Report titled “Build a Basic Bank Account System with Deposit and Withdrawal” submitted by us and approved by our project guide, to the Vivekananda Global University, Jaipur is a Bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Project Guide: Mr. Narayan Vyas

Student Name:	GAURAV SINGH	24CSA3BC036
	GAURAV SHARMA	24CSA3BC004
	FAIYYAZ KHAN	24CSA3BC105
	HARSH KUMAR	24CSA3BC064

Table of Contents

Introduction

In the digital age, managing personal finances efficiently has become an essential aspect of everyday life. With the rise of online banking and digital transactions, understanding the fundamentals of a banking system is important. One of the most basic, yet critical, components of a banking system is a bank account. A bank account serves as a secure repository for holding money, where users can perform essential transactions such as deposits, withdrawals, and balance inquiries.

The purpose of this project is to create a **basic bank account system** that simulates essential banking functionalities such as **depositing** money, **withdrawing** money, and **checking the balance** of an account. The system will be developed using Java, a popular object-oriented programming language that is widely used for building applications that require a high degree of modularity and maintainability.

This system will:

- **Allow the user to deposit funds** into their bank account.
- **Enable withdrawals** of funds, provided there is enough balance in the account.
- **Display the account balance**, allowing users to monitor their available funds.

While this system is designed to simulate a basic bank account, it provides the foundation for understanding how more complex banking applications function. It focuses on the core concepts of object-oriented programming (OOP), such as creating classes and methods, managing user input, and performing basic error handling.

In this report, we will discuss the objectives of the system, its design and implementation, and how it simulates the key functionalities of a typical bank account. Furthermore, we will explore potential future enhancements and additional features that could be incorporated to extend the system's functionality. Through this project, the reader will gain insights into how simple banking operations can be modeled and implemented using programming techniques.

2. Objective

2.1. Primary Goals

The primary goals of the project were:

- To develop a simple system for managing basic banking operations.
- To ensure that deposits and withdrawals are processed accurately and efficiently.
- To implement basic error handling for input validation and ensure proper balance management.

2.2. System Features

Key features of the bank account system include:

- **Account Creation**: Users can create a new account with an initial deposit.
- **Deposit**: Users can deposit funds into their accounts.
- **Withdrawal**: Users can withdraw money, provided their account balance is sufficient.
- **Balance Checking**: Users can view the current balance of their accounts.

3. System Requirements

3.1. Software Requirements

- **Programming Language**: Python (or Java/C++ could be used as alternatives).
- **Operating System**: Windows, Linux, or macOS.
- **IDE**: Any text editor or Integrated Development Environment (IDE) like Visual Studio Code or PyCharm.

3.2. Hardware Requirements

- **Processor**: Any modern processor capable of running the chosen programming language.
- **RAM**: Minimum of 2 GB.
- **Storage**: Sufficient space for the source code and testing environment (around 100 MB).

3.3. Tools Used

- **Python** for implementation.
- **Version Control**: Git for source code management.
- **Testing Framework**: Python's built-in unit test framework for testing.

4. System Design

4.1. Architecture Overview

The system is structured with two primary classes:

- **Bank Account**: Represents an individual bank account with properties like account number, holder name, and balance.
- **Bank Account System**: Manages a collection of Bank Account objects, allowing users to perform operations like deposits and withdrawals.

4.2. Bank Account Class Design

The Bank Account class contains:

- **Attributes**: Account number, account holder name, and balance.
- **Methods**: Methods for depositing money, withdrawing funds, and checking the balance.

4.3. Bank Account System Class Design

The Bank Account System class manages the overall operation of multiple Bank Account objects:

- **Attributes**: A dictionary or list to hold multiple Bank Account objects.
- **Methods**: Methods to create accounts, process deposits and withdrawals, and handle basic user interaction.

4.4. Flow of Execution

- The user creates an account via Bank Account System.
- The user performs actions such as depositing or withdrawing funds.
- The system checks for sufficient funds, updates the balance, and displays the updated balance.

5. Implementation

5.1. BankAccount Class Implementation

```
import java.util.Scanner;

public class BankAccount {
   private String accountHolder;
   private double balance;

   // Constructor
   public BankAccount(String name, double initialBalance) {
      accountHolder = name;
      balance = initialBalance;
   }

   // Deposit method
   public void deposit(double amount) {
      if (amount > 0) {
         balance += amount;
         System.out.println("Deposited: ₹" + amount);
      } else {
         System.out.println("Invalid deposit amount.");
      }
   }

   // Withdraw method
```

```
public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
        System.out.println("Withdrawn: ₹" + amount);
    } else {
        System.out.println("Insufficient balance or invalid amount.");
    }
}

// Display balance
public void displayBalance() {
    System.out.println("Current Balance: ₹" + balance);
}

// Main method
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Welcome to Simple Bank System");

    System.out.print("Enter your name: ");
    String name = scanner.nextLine();

    System.out.print("Enter initial deposit amount: ₹");
    double initialAmount = scanner.nextDouble();
```

```
BankAccount account = new BankAccount(name, initialAmount);
account.displayBalance();

boolean exit = false;
while (!exit) {
   System.out.println("\nChoose an operation:");
   System.out.println("1. Deposit");
   System.out.println("2. Withdraw");
   System.out.println("3. Show Balance");
   System.out.println("4. Exit");
   System.out.print("Enter choice: ");
   int choice = scanner.nextInt();

   switch (choice) {
      case 1:
         System.out.print("Enter amount to deposit: ₹");
         double depositAmount = scanner.nextDouble();
         account.deposit(depositAmount);
         break;
      case 2:
         System.out.print("Enter amount to withdraw: ₹");
         double withdrawAmount = scanner.nextDouble();
         account.withdraw(withdrawAmount);
         break;
      case 3:
```

```
                account.displayBalance();
                break;
            case 4:
                exit = true;
                System.out.println("Thank you for using the bank system!");
                break;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
    scanner.close();
  }
}
```

Output:

Welcome to Simple Bank System

Enter your name: Gaurav Singh

Enter initial deposit amount: ₹1000

Current Balance: ₹1000.0

Choose an operation:

1. Deposit

2. Withdraw

3. Show Balance

4. Exit

Enter choice: 1

Enter amount to deposit: ₹500

Deposited: ₹500.0

Choose an operation:

1. Deposit

2. Withdraw

3. Show Balance

4. Exit

Enter choice: 3

Current Balance: ₹1500.0

Choose an operation:

1. Deposit

2. Withdraw

3. Show Balance

4. Exit

Enter choice: 2

Enter amount to withdraw: ₹200

Withdrawn: ₹200.0

Choose an operation:

1. Deposit

2. Withdraw

3. Show Balance

4. Exit

Enter choice: 3

Current Balance: ₹1300.0

Choose an operation:

1. Deposit

2. Withdraw

3. Show Balance

4. Exit

Enter choice: 4

Thank you for using the bank system!

6.Explanation of the Code

6.1. Bank Account Class:

- **Attributes: account holder, account Number, and balance.**
- **Methods:**
  - **Deposit (double amount): Adds money to the balance.**
  - **Withdraw (double amount): Subtracts money from the balance (checks for sufficient funds).**
  - **Display Balance (): Displays the current balance of the account.**

6.2. Bank Account System Class (Main Program):

- **Initializes a new Bank Account with details (name, account number, and initial deposit).**
- **Provides a menu for users to:**
  - **Deposit money.**
  - **Withdraw money.**
  - **Check balance.**
  - **Exit the program.**

7. Challenges and Solutions

7.1. Input Validation Challenges

Handling invalid inputs, like negative numbers for deposits or withdrawals, was challenging.

7.2. Error Handling Challenges

Ensuring that users could not withdraw more money than they had was a primary challenge.

7.3. Solution to Issues

- Input validation was implemented by checking if amounts were positive before performing operations.
- Error handling for insufficient funds was implemented within the withdraw method.

8. Future Enhancements

8.1. Graphical User Interface (GUI)

Developing a GUI to make the system more user-friendly, such as using Tkinter in Python.

8.2. Transaction History Feature

Implementing a feature to log all transactions and display them to users.

8.3. User Authentication and Security

Adding user authentication (e.g., password protection) for secure access to accounts.

8.4. Multi-Account and Multi-User Support

Expanding the system to handle multiple users, each with multiple accounts.

9. Conclusion

9.1. Summary

This report described the development of a basic bank account system that supports deposits and withdrawals. The system is simple, but it covers the core features necessary for basic account management.

9.2. Key Takeaways

The project successfully demonstrated how to implement essential banking operations with simple object-oriented programming principles. Future enhancements could include adding advanced features like security, transaction history, and multi-user support.

10. References

1. **Java Documentation**: https://docs.oracle.com/javase/8/docs/api/
2. **Object-Oriented Programming (OOP) Concepts**: https://en.wikipedia.org/wiki/Object-oriented_programming
3. **Scanner Class in Java**: https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html