

# JAVA DEVELOPER INTERNSHIP

## Task 5: Bank Account Simulation

**Soumen Das**

November 20, 2025

### OBJECTIVE

---

The objective of Task 5 was to simulate basic banking operations using Java Object-Oriented Programming (OOP) principles. The simulation required the implementation of a class-based program capable of handling:

- Account creation.
- Deposit and Withdrawal operations.
- Maintenance of account balance.
- Transaction history logging.

### TECHNICAL IMPLEMENTATION

---

The solution was implemented using a single Java class structure containing two main components:

1. **BankAccount Class:** Encapsulates the account details (Account Number, Holder Name, Balance) and business logic.
2. **BankSimulation Class:** Contains the main method to provide a user interface via the console.

#### 2.1 Key Features

- **Encapsulation:** All account fields are private to ensure data security.
- **Validation:** Withdrawal logic checks for sufficient funds before processing.
- **History Tracking:** An `ArrayList` is used to store a log of every transaction with a timestamp.

### SOURCE CODE

---

Below is the complete source code developed for this task.

```
1 import java.util.ArrayList;  
2 import java.util.List;  
3 import java.util.Scanner;  
4 import java.time.LocalDateTime;  
5 import java.time.format.DateTimeFormatter;
```

```

6
7 class BankAccount {
8     private String accountHolderName;
9     private String accountNumber;
10    private double balance;
11    private List<String> transactionHistory;
12
13    public BankAccount(String name, String accNum, double initialDeposit) {
14        this.accountHolderName = name;
15        this.accountNumber = accNum;
16        this.balance = initialDeposit;
17        this.transactionHistory = new ArrayList<>();
18        addTransaction("Init_Balance:_$" + initialDeposit);
19    }
20
21    public void deposit(double amount) {
22        if (amount > 0) {
23            balance += amount;
24            addTransaction("Deposited:_$" + amount);
25            System.out.println("Success:_Deposited_$" + amount);
26        }
27    }
28
29    public void withdraw(double amount) {
30        if (amount > 0 && amount <= balance) {
31            balance -= amount;
32            addTransaction("Withdrew:_$" + amount);
33            System.out.println("Success:_Withdrew_$" + amount);
34        } else {
35            System.out.println("Error:_Insufficient_funds.");
36        }
37    }
38
39    public void printStatement() {
40        System.out.println("\n---_Statement_---");
41        for (String record : transactionHistory) {
42            System.out.println(record);
43        }
44    }
45
46    private void addTransaction(String message) {
47        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd_HH:mm:ss");
48        LocalDateTime now = LocalDateTime.now();
49        transactionHistory.add("[ " + dtf.format(now) + " ]_" + message);
50    }
51 }
52
53 public class BankSimulation {
54     public static void main(String[] args) {
55         // Simulation logic
56         BankAccount myAccount = new BankAccount("Soumen_Das", "SBIN001", 1000.0)
57         ;
58         myAccount.deposit(500);
59         myAccount.withdraw(200);
60         myAccount.printStatement();
61     }
62 }

```

---

### Listing 1: BankSimulation.java

## CONCLUSION

---

The BankSimulation program successfully meets all requirements outlined in Task 5. It demonstrates the practical application of Classes, Objects, and Methods in Java to solve real-world scenarios.