Java Assignment Day 3

Case Study Title: Banking System Application Using OOPs Concepts

BankOperations (Interface)

```
package JavaAssigment_Day3;

public interface BankOperations {

    void deposit(double amount);

    void withdraw(double amount);

    void transfer(Account target, double amount);

    double checkBalance();

    void showTransactionHistory();

}}
```

Account (Abstract Class)

```
package JavaAssigment_Day3;

import java.util.*;

abstract class Account implements BankOperations {

    protected String accountNumber;

    protected double balance;

    protected List<String> transactionHistory = new ArrayList<>();


    public Account(String accountNumber, double initialBalance) {

        this.accountNumber = accountNumber;

        this.balance = initialBalance;

        addTransaction("Account [" + accountNumber + "] opened with initial balance: ₹" +
initialBalance);

    }


    public void transfer(Account target, double amount) {

        if (this.balance >= amount) {

            this.balance -= amount;

            target.balance += amount;

            addTransaction("Transferred ₹" + amount + " to Account " + target.accountNumber);

            target.addTransaction("Received from Account " + this.accountNumber + ": ₹" +
amount);

        } else {

            System.out.println("Insufficient funds to transfer.");
```

```
        }
    }

    public double checkBalance() {
        return balance;
    }

    public void showTransactionHistory() {
        System.out.println("Transaction History:");
        for (String entry : transactionHistory) {
            System.out.println(entry);
        }
    }

    protected void addTransaction(String info) {
        transactionHistory.add(info);
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public abstract void deposit(double amount);
    public abstract void withdraw(double amount);
}
```

SavingsAccount (extends Account, implements BankOperations)

```
package JavaAssigment_Day3;

public class SavingsAccount extends Account {
    private final double MIN_BALANCE = 1000.0;

    public SavingsAccount(String accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
    }
```

```
    public void deposit(double amount) {
        balance += amount;
        addTransaction("Deposited ₹" + amount);
    }

    public void withdraw(double amount) {
        if ((balance - amount) >= MIN_BALANCE) {
            balance -= amount;
            addTransaction("Withdrawn ₹" + amount);
        } else {
            System.out.println("Minimum balance requirement not met.");
        }
    }
}
```

CurrentAccount (extends Account, implements BankOperations)

```
package JavaAssigment_Day3;

public class CurrentAccount extends Account {
    private final double OVERDRAFT_LIMIT = 2000.0;

    public CurrentAccount(String accountNumber, double initialBalance) {
        super(accountNumber, initialBalance);
    }

    public void deposit(double amount) {
        balance += amount;
        addTransaction("Deposited ₹" + amount);
    }

    public void withdraw(double amount) {
        if ((balance - amount) >= -OVERDRAFT_LIMIT) {
            balance -= amount;
            addTransaction("Withdrawn ₹" + amount);
        } else {
            System.out.println("Overdraft limit exceeded.");
```

```
        }
    }
}
```

Customer

```
package JavaAssigment_Day3;


import java.util.ArrayList;
import java.util.List;


public class Customer {
    private String customerId;
    private String name;
    private List<Account> accounts = new ArrayList<>();

    public Customer(String customerId, String name) {
        this.customerId = customerId;
        this.name = name;
        System.out.println("Customer Created: " + name + " [Customer ID: " + customerId + "]");
    }

    public void addAccount(Account acc) {
        accounts.add(acc);
    }

    public List<Account> getAccounts() {
        return accounts;
    }

    public String getCustomerId() {
        return customerId;
    }

    public String getName() {
        return name;
    }
}
```

BankBranch

```
package JavaAssigment_Day3;


import java.util.ArrayList;

import java.util.List;


public class BankBranch {

    private String branchId;

    private String branchName;

    private List<Customer> customers = new ArrayList<>();


    public BankBranch(String branchId, String branchName) {

        this.branchId = branchId;

        this.branchName = branchName;

        System.out.println(" Branch Created: " + branchName + " [Branch ID: " + branchId +
"]");

    }


    public void addCustomer(Customer c) {

        customers.add(c);

        System.out.println(" Customer added to branch.");

    }


    public Customer findCustomerById(String id) {

        for (Customer c : customers) {

            if (c.getCustomerId().equals(id)) return c;

        }

        return null;

    }


    public void listAllCustomers() {

        for (Customer c : customers) {

            System.out.println(" " + c.getName());

        }

    }
```

```
}
```

Main

```
package JavaAssigment_Day3;

import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // 1. Create Branch

        System.out.print("Enter Branch ID: ");

        String branchId = scanner.nextLine();

        System.out.print("Enter Branch Name: ");

        String branchName = scanner.nextLine();

        BankBranch branch = new BankBranch(branchId, branchName);


        // 2. Create Customer

        System.out.print("Enter Customer ID: ");

        String customerId = scanner.nextLine();

        System.out.print("Enter Customer Name: ");

        String customerName = scanner.nextLine();

        Customer customer = new Customer(customerId, customerName);

        branch.addCustomer(customer);


        // 3. Create Accounts

        System.out.print("Enter Savings Account ID: ");

        String savingsId = scanner.nextLine();

        System.out.print("Enter Savings Initial Balance: ");

        double savingsInitBalance = scanner.nextDouble();

        SavingsAccount savings = new SavingsAccount(savingsId, savingsInitBalance);

        customer.addAccount(savings);


        System.out.print("Enter Current Account ID: ");

        scanner.nextLine();  // consume leftover newline

        String currentId = scanner.nextLine();

        System.out.print("Enter Current Initial Balance: ");

        double currentInitBalance = scanner.nextDouble();
```

```
        CurrentAccount current = new CurrentAccount(currentId, currentInitBalance);
        customer.addAccount(current);

        // 4. Deposit to Savings
        System.out.print("Enter amount to deposit in Savings: ");
        double depositAmt = scanner.nextDouble();
        savings.deposit(depositAmt);
        System.out.println(" Current Savings Balance: ₹" + savings.checkBalance());

        // 5. Withdraw from Current
        System.out.print("Enter amount to withdraw from Current: ");
        double withdrawAmt = scanner.nextDouble();
        current.withdraw(withdrawAmt);
        System.out.println(" Current Account Balance: ₹" + current.checkBalance());

        // 6. Transfer
        System.out.print("Enter amount to transfer from Savings to Current: ");
        double transferAmt = scanner.nextDouble();
        savings.transfer(current, transferAmt);
        System.out.println(" Savings Balance: ₹" + savings.checkBalance());
        System.out.println(" Current Balance: ₹" + current.checkBalance());

        // 7. Show Transactions
        System.out.println("\n Transaction History:");
        System.out.println("Account: " + savings.getAccountNumber());
        savings.showTransactionHistory();
        System.out.println("\nAccount: " + current.getAccountNumber());
        current.showTransactionHistory();

        scanner.close();
    }
}
```