Day3 Java OOPs Assignment - Banking System

1. BankOperations Interface

```
package day3Assignment;

public interface BankOperations {
    void deposit(double amount);
    void withdraw(double amount);
    void transfer(Account target, double amount);
    double checkBalance();
    void showTransactionHistory();
}
```

2. Account Abstract Class

```
package day3Assignment;

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

public abstract class Account implements BankOperations {
    protected String accountNumber;
    protected double balance;
    protected List<String> transactionHistory;

    public Account(String accountNumber, double initialBalance) {
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
        this.transactionHistory = new ArrayList<>();
    }

    public void transfer(Account target, double amount) {
        if (this.balance >= amount) {
            this.withdraw(amount);
            target.deposit(amount);
            this.addTransaction("Transferred to Account " +
target.accountNumber + ": ₹" + amount);
            target.addTransaction("Received from Account " +
this.accountNumber + ": ₹" + amount);
        } else {
            System.out.println("Transfer failed: Insufficient
balance.");
        }
```

```
    }

    public double checkBalance() {
        return balance;
    }

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

    public void showTransactionHistory() {
        System.out.println("Account: " + accountNumber);
        for (String record : transactionHistory) {
            System.out.println(" - " + record);
        }
    }
}
```

3. SavingsAccount Class

```
package day3Assignment;

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

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

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

    @Override
    public void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;
            addTransaction("Withdrawn: ₹" + amount);
        } else {
            System.out.println("Withdrawal failed: Minimum balance ₹" +
MIN_BALANCE + " required.");
        }
    }
}
```

4. CurrentAccount Class

```
package day3Assignment;

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

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

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

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

5. Customer Class

```
package day3Assignment;

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

public class Customer {
    private String customerId;
    private String name;
    private List<Account> accounts;

    public Customer(String customerId, String name) {
        this.customerId = customerId;
        this.name = name;
        this.accounts = new ArrayList<>();
    }

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

```
    }

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

    public String getCustomerId() {
        return customerId;
    }

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

6. BankBranch Class

```
package day3Assignment;

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

public class BankBranch {
    private String branchId;
    private String branchName;
    private List<Customer> customers;

    public BankBranch(String branchId, String branchName) {
        this.branchId = branchId;
        this.branchName = branchName;
        this.customers = new ArrayList<>();
    }

    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() {
        System.out.println("Customers at Branch " + branchName + ":");
```

```
        for (Customer c : customers) {
            System.out.println("- " + c.getName() + " [" +
c.getCustomerId() + "]");
        }
    }
}
```