DAY 3 Assessment

```
import java.util.*;

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

// Abstract class
abstract class Account implements BankOperations {
    protected String accountNumber;
    protected double balance;
    protected List<String> transactionHistory = new ArrayList<>();

    public Account(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

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

    public void transfer(Account target, double amount) {
        if (this.balance >= amount) {
            this.withdraw(amount);
            target.deposit(amount);
            addTransaction("Transferred to Account " + target.accountNumber + ": " + amount);
```

```
            target.addTransaction("Received from Account " + this.accountNumber + ": " + amount);
        } else {
            System.out.println(" Insufficient funds for transfer.");
        }
    }

    public double checkBalance() {
        return balance;
    }

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

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

    public String getAccountNumber() {
        return accountNumber;
    }
}

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

    public SavingsAccount(String accountNumber, double balance) {
```

```
        super(accountNumber, balance);
    }

    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            addTransaction("Withdrawn: " + amount);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(" Cannot withdraw. Minimum balance requirement not met.");
        }
    }
}

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

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

    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 class
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;
    }

    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 class
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() + " [ID: " + c.getCustomerId() + "]");
        }
    }
}

// Main Class
public class BankingSystem {
    public static void main(String[] args) {
        // Step 1: Create branch
        BankBranch branch = new BankBranch("B001", "Main Branch");

        // Step 2: Create customer
        Customer c1 = new Customer("C001", "Alice");
        System.out.println(" Customer Created: " + c1.getName() + " [Customer ID: " +
c1.getCustomerId() + "]");
        branch.addCustomer(c1);

        // Step 3: Create accounts
        SavingsAccount sa = new SavingsAccount("S001",
        5000.0);        CurrentAccount ca = new
        CurrentAccount("C001", 2000.0); c1.addAccount(sa);
        c1.addAccount(ca);
        System.out.println(" Savings Account [S001] opened with initial balance: ₹5000.0");
        System.out.println(" Current Account [C001] opened with initial balance: ₹2000.0 and overdraft
limit ₹2000.0");

        // Step 4: Deposit to savings
        sa.deposit(2000.0);
```

```
        System.out.println(" Deposited 2000.0 to Savings Account [S001]");
        System.out.println(" Current Balance: " + sa.checkBalance());

        // Step 5: Withdraw from current using overdraft
        ca.withdraw(2500.0);
        System.out.println(" Withdrawn 2500.0 from Current Account [C001]");
        System.out.println(" Current Balance: " + ca.checkBalance());

        // Step 6: Transfer from savings to current
        sa.transfer(ca, 1000.0);
        System.out.println("Transferred 1000.0 from Savings Account [S001] to Current Account [C001]");
        System.out.println(" Savings Balance: " + sa.checkBalance());
        System.out.println(" Current Balance: " + ca.checkBalance());

        // Step 7: Show transaction history
        sa.showTransactionHistory();
        ca.showTransactionHistory();
    }
}
```