Day 3 Java Assignments

Banking System Application Using OOPs Concepts

BankOperation.java

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

Account.java

```
package Assignment;
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 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 balance for transfer.");
        }
    }
    public double checkBalance() {
        return balance;
    }
    protected 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);
        }
    }
}
```

SavingsAccount.java

```
package Assignment;
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);
        System.out.println("Deposited ₹" + amount + " to Savings Account [" + accountNumber
+ "]");
    }
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;
            addTransaction("Withdrawn: ₹" + amount);
            System.out.println("Withdrawn ₹" + amount + " from Savings Account [" +
accountNumber + "]");
        } else {
            System.out.println("Minimum balance requirement not met!");
        }
    }
}
```

CurrentAccount.java

```
package Assignment;
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);
        System.out.println("Deposited ₹" + amount + " to Current Account [" + accountNumber
+ "]");
    }
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= -OVERDRAFT_LIMIT) {
            balance -= amount;
            addTransaction("Withdrawn: ₹" + amount);
            System.out.println("Withdrawn ₹" + amount + " from Current Account [" +
accountNumber + "]");
        } else {
            System.out.println("Overdraft limit exceeded!");
        }
    }
}
```

Customer.java

```
package Assignment;
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;
    }
}
```

BankBranch.java

```
package Assignment;
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<>();
        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() {
        System.out.println("Customers at Branch " + branchName);
        for (Customer c : customers) {
            System.out.println(" - " + c.getName() + " [ID: " + c.getCustomerId() + "]");
        }
    }
}
```

Main.java

```
package Assignment;
public class Main {
    public static void main(String[] args) {
        BankBranch branch = new BankBranch("B001", "Main Branch");

        Customer c1 = new Customer("C001", "Alice");
        branch.addCustomer(c1);

        SavingsAccount sa = new SavingsAccount("S001", 50000.0);
        CurrentAccount ca = new CurrentAccount("C001", 20000.0);
```

```
        c1.addAccount(sa);
        c1.addAccount(ca);

        sa.deposit(20000);
        ca.withdraw(25000);
        sa.transfer(ca, 10000);

        System.out.println("\n Final Balances:");
        System.out.println("Savings: ₹" + sa.checkBalance());
        System.out.println("Current: ₹" + ca.checkBalance());

        System.out.println("\n📋 Full Transaction History:");
        sa.showTransactionHistory();
        ca.showTransactionHistory();
    }
}
```