**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)**

import java.util.\*

public 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 void transfer(Account target, double amount) {

if (amount > 0 && this.balance >= amount) {

this.withdraw(amount);

target.deposit(amount);

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

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

}

}

public double checkBalance() {

return balance;

}

protected void addTransaction(String info) {

transactionHistory.add(info);

}

public void showTransactionHistory() {

System.out.println("Account: " + accountNumber);

for (String t : transactionHistory) {

System.out.println("- " + t);

}

}

}

**3.SavingsAccount (extends Account, implements BankOperations)**

public 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addTransaction("Withdrawn: ₹" + amount);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 must be ₹" + MIN\_BALANCE);

}

}

}

**4.CurrentAccount (extends Account, implements BankOperations)**

public 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addTransaction("Withdrawn: ₹" + amount);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("Cannot withdraw: Overdraft limit exceeded");

}

}

}

**6.Customer**

import java.util.\*;

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;

}

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**

import java.util.\*;

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("Customer: " + c.getName() + " [ID: " + c.getCustomerId() + "]");

}

}

}

**7.MAIN**

package Day3Assignment;

public class Main {

public static void main(String[] args) {

BankBranch branch = new BankBranch("B001", "Main Branch");

Customer c1 = new Customer("C001", "Alice");

System.***out***.println("Customer Created: " + c1.getName() + " [Customer ID: " + c1.getCustomerId() + "]");

SavingsAccount sa = new SavingsAccount("S001", 5000);

CurrentAccount ca = new CurrentAccount("C001", 2000);

c1.addAccount(sa);

c1.addAccount(ca);

branch.addCustomer(c1);

sa.deposit(2000);

System.out.println("Current Balance: ₹" + sa.checkBalance());

ca.withdraw(2500);

System.out.println("Current Balance: ₹" + ca.checkBalance());

sa.transfer(ca, 1000);

System.out.println("Savings Balance: ₹" + sa.checkBalance());

System.out.println("Current Balance: ₹" + ca.checkBalance());

System.out.println("\nTransaction History:");

sa.showTransactionHistory();

ca.showTransactionHistory();

}

}