**Министр науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 1

Выполнил студент группы № M32071

Реброва Татьяна Ивановна

Проверил:

Бутенко Олег Романович

Санкт-Петербург  
2022

Credit.java  
  
package bankAccounts;  
  
import entities.BankAccount;  
  
public class Credit extends BankAccount{  
 private double owesMoney;  
 public Credit(double money, double limit, double percent, double owesMoney) {  
 super(money, limit, percent);  
 this.owesMoney = owesMoney;  
 }  
  
 public double getOwesMoney() {  
 return owesMoney;  
 }  
  
 public void setOwesMoney(double owesMoney) {  
 this.owesMoney = owesMoney;  
 }  
  
 @Override  
 public void countPercent(int amountMonth) {  
 owesMoney += ((owesMoney / 100) \* getPercent()) \* amountMonth;  
 }  
}  
  
  
  
Debit.java  
  
package bankAccounts;  
  
import entities.BankAccount;  
  
public class Debit extends BankAccount {  
 public Debit(double money, double limit, double percent) {  
 super(money, limit, percent);  
 }  
}  
  
  
  
Deposit.java  
  
package bankAccounts;  
  
import entities.BankAccount;  
import entities.Transaction;  
import entities.Client;  
import entities.Bank;  
import tools.BankException;  
  
public class Deposit extends BankAccount{  
 public Deposit(double money, double limit, double percent) {  
 super(money, limit, percent);  
 }  
  
 @Override  
 public Transaction makeTransaction(Client client, BankAccount bankAccount1, BankAccount bankAccount2, double money) throws BankException {  
 throw new BankException("Sorry, transaction are not available for deposit accounts");  
 }  
  
 @Override  
 public Transaction withdrawCash(Bank bank, Client client, BankAccount bankAccount, double money) throws BankException {  
 throw new BankException("Sorry, you cannot withdraw money from the deposit account");  
 }  
}  
  
  
  
Bank.java  
  
package entities;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.UUID;  
  
public class Bank {  
 private UUID id;  
 private String name;  
 private double percentDebit;  
 private double percentCredit;  
 private double percentDeposit;  
 private double limitDebit;  
 private double limitCredit;  
 private double limitDeposit;  
 private List<Client> clients;  
 private List<BankAccount> bankAccounts;  
  
 public Bank(String name, double percentDebit,  
 double percentCredit, double percentDeposit,  
 double limitDebit, double limitCredit, double limitDeposit) {  
 this.id = UUID.randomUUID();  
 this.name = name;  
 this.percentDebit = percentDebit;  
 this.percentCredit = percentCredit;  
 this.percentDeposit = percentDeposit;  
 this.limitDebit = limitDebit;  
 this.limitCredit = limitCredit;  
 this.limitDeposit = limitDeposit;  
 this.bankAccounts = new ArrayList<BankAccount>();  
 this.clients = new ArrayList<Client>();  
 }  
  
 public UUID getId() {  
 return id;  
 }  
  
 public void setId(UUID id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getPercentDebit() {  
 return percentDebit;  
 }  
  
 public void setPercentDebit(double percentDebit) {  
 this.percentDebit = percentDebit;  
 }  
  
 public double getPercentCredit() {  
 return percentCredit;  
 }  
  
 public void setPercentCredit(double percentCredit) {  
 this.percentCredit = percentCredit;  
 }  
  
 public double getPercentDeposit() {  
 return percentDeposit;  
 }  
  
 public void setPercentDeposit(double percentDeposit) {  
 this.percentDeposit = percentDeposit;  
 }  
  
 public double getLimitDebit() {  
 return limitDebit;  
 }  
  
 public void setLimitDebit(double limitDebit) {  
 this.limitDebit = limitDebit;  
 }  
  
 public double getLimitCredit() {  
 return limitCredit;  
 }  
  
 public void setLimitCredit(double limitCredit) {  
 this.limitCredit = limitCredit;  
 }  
  
 public double getLimitDeposit() {  
 return limitDeposit;  
 }  
  
 public void setLimitDeposit(double limitDeposit) {  
 this.limitDeposit = limitDeposit;  
 }  
  
 public List<Client> getClients() {  
 return clients;  
 }  
  
 public void setClients(List<Client> clients) {  
 this.clients = clients;  
 }  
  
 public List<BankAccount> getBankAccounts() {  
 return bankAccounts;  
 }  
  
 public void setBankAccounts(List<BankAccount> bankAccounts) {  
 this.bankAccounts = bankAccounts;  
 }  
  
 public void countPercent(int amountMonth) {  
 for (BankAccount bankAccount : bankAccounts) {  
 bankAccount.countPercent(amountMonth);  
 }  
 }  
}  
  
  
  
  
  
BankAccount.java  
  
package entities;  
  
import tools.BankException;  
import bankAccounts.Debit;  
import java.util.Objects;  
import java.util.UUID;  
  
public abstract class BankAccount {  
 private UUID id;  
 private double money;  
 private double limit;  
 private double percent;  
 public BankAccount(double money, double limit, double percent) {  
 this.id = UUID.randomUUID();  
 this.money = money;  
 this.limit = limit;  
 this.percent = percent;  
 }  
  
 public UUID getId() {  
 return id;  
 }  
  
 public void setId(UUID id) {  
 this.id = id;  
 }  
  
 public double getMoney() {  
 return money;  
 }  
  
 public void setMoney(double money) {  
 this.money = money;  
 }  
  
 public double getLimit() {  
 return limit;  
 }  
  
 public void setLimit(double limit) {  
 this.limit = limit;  
 }  
  
 public double getPercent() {  
 return percent;  
 }  
  
 public void setPercent(double percent) {  
 this.percent = percent;  
 }  
  
 public Transaction makeTransaction(Client client,  
 BankAccount bankAccount1,  
 BankAccount bankAccount2,  
 double money) throws BankException {  
 if (!Objects.equals(client.getPassport(), "") && bankAccount1.getLimit() < money) {  
 throw new BankException("Sorry, you are not fully registered client");  
 }  
  
 if (bankAccount1 instanceof Debit && bankAccount1.getMoney() < 0) {  
 throw new BankException("Sorry, insufficient funds");  
 }  
  
 Transaction transaction = new Transaction(bankAccount1, bankAccount2, money);  
 bankAccount1.setMoney(bankAccount1.getMoney() - money);  
 bankAccount2.setMoney(bankAccount2.getMoney() + money);  
 return transaction;  
 }  
  
 public Transaction withdrawCash(Bank bank, Client client, BankAccount bankAccount, double money) throws BankException {  
 Transaction transaction = new Transaction(bankAccount, null, money);  
 if (bank.getClients().stream().filter(anyClient -> anyClient == client).findFirst() == null) {  
 throw new BankException("Sorry, client not found this bank");  
 }  
  
 if (client.getListAccounts().stream().filter(anyBankAccount -> anyBankAccount == bankAccount).findFirst() == null) {  
 throw new BankException("Sorry, client don't have this bank account");  
 }  
  
 bankAccount.setMoney(bankAccount.getMoney() - money);  
 return transaction;  
 }  
  
 public void topUpCash(Bank bank, Client client, BankAccount bankAccount, double money) throws BankException {  
 if (bank.getClients().stream().filter(anyClient -> anyClient == client) == null) {  
 throw new BankException("Sorry, client not found this bank");  
 }  
  
 if (client.getListAccounts().stream().filter(anyBankAccount -> anyBankAccount == bankAccount) == null) {  
 throw new BankException("Sorry, client don't have this bank account");  
 }  
  
 bankAccount.setMoney(bankAccount.getMoney() + money);  
 }  
  
 public void countPercent(int amountMonth)  
 {  
 money += ((money / 100) \* percent) \* amountMonth;  
 }  
}  
  
  
CentralBank.java  
  
package entities;  
  
import tools.BankException;  
import bankAccounts.Credit;  
import bankAccounts.Debit;  
import bankAccounts.Deposit;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
import java.util.UUID;  
  
public class CentralBank {  
 public final int countMonth = 30;  
 private String name;  
 private List<Bank> banks;  
 private List<Transaction> transactions;  
  
 public CentralBank(String name){  
 this.name = name;  
 this.banks = new ArrayList<Bank>();  
 this.transactions = new ArrayList<Transaction>();  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public List<Bank> getBanks() {  
 return banks;  
 }  
  
 public void setBanks(List<Bank> banks) {  
 this.banks = banks;  
 }  
  
 public List<Transaction> getTransactions() {  
 return transactions;  
 }  
  
 public void setTransactions(List<Transaction> transactions) {  
 this.transactions = transactions;  
 }  
  
 public Bank addBank(String name, double percentDebit,  
 double percentCredit, double percentDeposit,  
 double limitDebit, double limitCredit, double limitDeposit) {  
 Bank bank = new Bank(name, percentDebit, percentCredit, percentDeposit, limitDebit, limitCredit, limitDeposit);  
 banks.add(bank);  
 return bank;  
 }  
  
 public void addTransaction(Transaction transaction) {  
 transactions.add(transaction);  
 }  
  
 public void cancelTransaction(UUID id) throws BankException {  
 Transaction transaction = transactions.stream()  
 .filter(transaction1 -> transaction1.getId() == id).findFirst().orElse(null);  
 if(transaction == null) {  
 throw new BankException("Sorry, no such transaction id or transaction has already been completed");  
 }  
 transaction.getBankAccount1().setMoney(transaction.getBankAccount1().getMoney() + transaction.getMoney());  
 if(transaction.getBankAccount2() != null) {  
 transaction.getBankAccount2().setMoney(transaction.getBankAccount2().getMoney() - transaction.getMoney());  
 }  
  
 transactions.remove(transaction);  
 }  
  
 public Client createClient(String firstName, String lastName) {  
 Client client = new Client(firstName, lastName);  
 return client;  
 }  
  
 public Client addClientAddress(Client client, String address) {  
 ClientBuilder clientBuilder = new ClientBuilder();  
 clientBuilder.addFirstName(client.getFirstName());  
 clientBuilder.addLastName(client.getLastName());  
 clientBuilder.addAddress(address);  
 clientBuilder.addPassport(client.getPassport());  
 Client newClient = clientBuilder.toBuild();  
 return newClient;  
 }  
  
 public Client addClientPassport(Client client, String passport) {  
 ClientBuilder clientBuilder = new ClientBuilder();  
 clientBuilder.addFirstName(client.getFirstName());  
 clientBuilder.addLastName(client.getLastName());  
 clientBuilder.addAddress(client.getAddress());  
 clientBuilder.addPassport(passport);  
 return clientBuilder.toBuild();  
 }  
  
 public void addClientBank(Client client, Bank bank) {  
 bank.getClients().add(client);  
 }  
  
 public Debit createDebitAccountForClient(Bank bank, Client client, double money) {  
 double limit;  
 if (!Objects.equals(client.getPassport(), null)) {  
 limit = Integer.MAX\_VALUE;  
 }  
 else {  
 limit = bank.getLimitDebit();  
 }  
  
 Debit debit = new Debit(money, limit, bank.getPercentDebit());  
 bank.getBankAccounts().add(debit);  
 return debit;  
 }  
  
 public Deposit createDepositAccountForClient(Bank bank, Client client, double money) {  
 double limit;  
 if(!Objects.equals(client.getPassport(), null)) {  
 limit = Integer.MAX\_VALUE;  
 }  
 else {  
 limit = bank.getLimitDeposit();  
 }  
  
 Deposit deposit = new Deposit(money, limit, bank.getPercentDeposit());  
 bank.getBankAccounts().add(deposit);  
 client.addInListAccount(deposit);  
 return deposit;  
 }  
  
 public Credit createCreditAccountForClient(Bank bank, Client client, double money) {  
 double limit;  
 if (!Objects.equals(client.getPassport(), null)) {  
 limit = Integer.MAX\_VALUE;  
 } else {  
 limit = bank.getLimitCredit();  
 }  
  
 Credit credit = new Credit(money, limit, bank.getPercentCredit(), money);  
 bank.getBankAccounts().add(credit);  
 client.addInListAccount(credit);  
 return credit;  
 }  
  
 public List<Bank> allCountPercent(int days) {  
 int amountMonth = days / countMonth;  
 for (Bank bank : banks) {  
 bank.countPercent(amountMonth);  
 }  
  
 return banks;  
 }  
  
 public void changePercentDebitAccount(Bank bank, double newPercent) {  
 bank.setPercentDebit(newPercent);  
 for (BankAccount bankAccount : bank.getBankAccounts()) {  
 if(bankAccount instanceof Debit) {  
 bankAccount.setPercent(newPercent);  
 }  
 }  
 }  
  
 public void changePercentCreditAccount(Bank bank, double newPercent) {  
 bank.setPercentCredit(newPercent);  
 for (BankAccount bankAccount : bank.getBankAccounts()) {  
 if(bankAccount instanceof Credit) {  
 bankAccount.setPercent(newPercent);  
 }  
 }  
 }  
  
 public void changePercentDepositAccount(Bank bank, double newPercent) {  
 bank.setPercentDeposit(newPercent);  
 for (BankAccount bankAccount : bank.getBankAccounts()) {  
 if(bankAccount instanceof Deposit) {  
 bankAccount.setPercent(newPercent);  
 }  
 }  
 }  
}  
  
  
Client.java  
  
package entities;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Client {  
 private String firstName;  
 private String lastName;  
 private String address;  
 private String passport;  
 private List<BankAccount> listAccounts;  
 public Client(String firstName, String lastName)  
 {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.address = null;  
 this.passport = null;  
 this.listAccounts = new ArrayList<BankAccount>();  
 }  
  
 public ClientBuilder toBuilder(ClientBuilder clientBuilder)  
 {  
 clientBuilder.addFirstName(firstName);  
 clientBuilder.addLastName(lastName);  
 clientBuilder.addAddress(address);  
 clientBuilder.addPassport(passport);  
 return clientBuilder;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getPassport() {  
 return passport;  
 }  
  
 public void setPassport(String passport) {  
 this.passport = passport;  
 }  
  
 public List<BankAccount> getListAccounts() {  
 return listAccounts;  
 }  
  
 public void setListAccounts(List<BankAccount> listAccounts) {  
 this.listAccounts = listAccounts;  
 }  
  
 public List<BankAccount> addInListAccount(BankAccount bankAccount) {  
 listAccounts.add(bankAccount);  
 return listAccounts;  
 }  
}  
  
  
ClientBuilder.java  
  
package entities;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ClientBuilder {  
 private String firstName;  
 private String lastName;  
 private String address;  
 private String passport;  
 private List<BankAccount> listAccounts;  
 public ClientBuilder addFirstName(String firstName)  
 {  
 this.firstName = firstName;  
 return this;  
 }  
  
 public ClientBuilder addLastName(String lastName)  
 {  
 this.lastName = lastName;  
 return this;  
 }  
  
 public ClientBuilder addAddress(String address)  
 {  
 this.address = address;  
 return this;  
 }  
  
 public ClientBuilder addPassport(String passport)  
 {  
 this.passport = passport;  
 return this;  
 }  
  
 public ClientBuilder addListAccounts(List<BankAccount> bankAccounts)  
 {  
 this.listAccounts = bankAccounts;  
 return this;  
 }  
  
 public Client toBuild()  
 {  
 Client finalClient = new Client(firstName, lastName);  
 return finalClient;  
 }  
}  
  
  
  
Transaction.java  
  
package entities;  
  
import java.util.UUID;  
  
public class Transaction {  
 private UUID id;  
 private BankAccount bankAccount1;  
 private BankAccount bankAccount2;  
 private double money;  
  
 public Transaction(BankAccount bankaccount1, BankAccount bankAccount2, double money){  
 this.id = UUID.randomUUID();  
 this.bankAccount1 = bankaccount1;  
 this.bankAccount2 = bankAccount2;  
 this.money = money;  
 }  
  
 public UUID getId() {  
 return id;  
 }  
  
 public void setId(UUID id) {  
 this.id = id;  
 }  
  
 public BankAccount getBankAccount1() {  
 return bankAccount1;  
 }  
  
 public void setBankAccount1(BankAccount bankAccount1) {  
 this.bankAccount1 = bankAccount1;  
 }  
  
 public BankAccount getBankAccount2() {  
 return bankAccount2;  
 }  
  
 public void setBankAccount2(BankAccount bankAccount2) {  
 this.bankAccount2 = bankAccount2;  
 }  
  
 public double getMoney() {  
 return money;  
 }  
  
 public void setMoney(double money) {  
 this.money = money;  
 }  
}  
  
  
  
BankTest.java  
  
package tests;  
  
import entities.CentralBank;  
import entities.Bank;  
import entities.Client;  
import entities.BankAccount;  
import entities.Transaction;  
import bankAccounts.Credit;  
import tools.BankException;  
import org.junit.Assert;  
import org.junit.Before;  
import org.junit.Test;  
  
public class BankTest {  
 private CentralBank \_centralBank;  
 final double delta = 0.000001;  
  
 @Before  
 public void setUp() {  
 \_centralBank = new CentralBank("Moscow Central Bank");  
 }  
  
 @Test  
 public void makeTransaction() throws BankException {  
 double money = 10000;  
 double money2 = 500;  
 Bank sberbank = \_centralBank.addBank("Sberbank", 0, 2, 3, 5000, 15000, 15000);  
 Bank tinkoff = \_centralBank.addBank("Tinkoff", 1, 1, 4, 6000, 20000, 20000);  
 Client client1 = \_centralBank.createClient("Tanya", "Rebrova");  
 Client client2 = \_centralBank.createClient("Petr", "Petrovich");  
 \_centralBank.addClientBank(client1, sberbank);  
 \_centralBank.addClientBank(client2, tinkoff);  
 BankAccount account1 = \_centralBank.createDebitAccountForClient(sberbank, client1, money);  
 BankAccount account2 = \_centralBank.createCreditAccountForClient(tinkoff, client2, money2);  
 Transaction transaction = account1.makeTransaction(client1, account1, account2, 1000);  
 Assert.assertEquals(9000, account1.getMoney(), delta);  
 Assert.assertEquals(1500, account2.getMoney(), delta);  
 }  
  
 @Test  
 public void cancelTransaction() throws BankException {  
 double money = 10000;  
 double money2 = 500;  
 Bank sberbank = \_centralBank.addBank("Sberbank", 0, 2, 3, 5000, 15000, 15000);  
 Bank tinkoff = \_centralBank.addBank("Tinkoff", 1, 1, 4, 6000, 20000, 20000);  
 Client client1 = \_centralBank.createClient("Tanya", "Rebrova");  
 Client client2 = \_centralBank.createClient("Petr", "Petrovich");  
 \_centralBank.addClientBank(client1, sberbank);  
 \_centralBank.addClientBank(client2, tinkoff);  
 BankAccount account1 = \_centralBank.createDebitAccountForClient(sberbank, client1, money);  
 BankAccount account2 = \_centralBank.createCreditAccountForClient(tinkoff, client2, money2);  
 Transaction transaction = account1.makeTransaction(client1, account1, account2, 1000);  
 \_centralBank.addTransaction(transaction);  
 \_centralBank.cancelTransaction(transaction.getId());  
 Assert.assertEquals(10000, account1.getMoney(), delta);  
 }  
  
 @Test  
 public void toSeeHowMuchMoneyInMonth() throws BankException {  
 double money = 10000;  
 double money2 = 500;  
 Bank sberbank = \_centralBank.addBank("Sberbank", 0, 2, 3, 5000, 15000, 15000);  
 Bank tinkoff = \_centralBank.addBank("Tinkoff", 1, 1, 4, 6000, 20000, 20000);  
 Client client1 = \_centralBank.createClient("Tanya", "Rebrova");  
 Client client2 = \_centralBank.createClient("Petr", "Petrovich");  
 \_centralBank.addClientBank(client1, sberbank);  
 \_centralBank.addClientBank(client2, tinkoff);  
 BankAccount account1 = \_centralBank.createDebitAccountForClient(sberbank, client1, money);  
 BankAccount account2 = \_centralBank.createCreditAccountForClient(tinkoff, client2, money2);  
 \_centralBank.allCountPercent(124);  
 Assert.assertEquals(520, ((Credit) account2).getOwesMoney(), delta);  
 Assert.assertEquals(10000, account1.getMoney(), delta);  
 }  
}  
  
  
  
  
BankException.java  
  
package tools;  
public class BankException extends Exception {  
 public BankException() {  
 }  
  
 public BankException(String message){  
 super(message);  
 }  
  
 public BankException(String message, Exception innerException){  
 super(message, innerException);  
 }  
}  
  
  
Program.java  
  
package tools;  
  
import entities.\*;  
  
import java.util.Objects;  
import java.util.Scanner;  
import static java.lang.System.out;  
  
public class Program {  
 public static Scanner scanner = new Scanner(System.in);  
 private final static CentralBank centralBank = new CentralBank("Moscow Central Bank");  
  
 public static void main(String[] args) throws BankException {  
 Client oldClient = centralBank.createClient("Ivan", "Ivanov");  
 Bank sberbank = centralBank.addBank("Sberbank", 0, 2, 3, 5000, 15000, 15000);  
 Bank tinkoff = centralBank.addBank("Tinkoff", 1, 1, 4, 6000, 20000, 20000);  
 centralBank.addClientBank(oldClient, tinkoff);  
 BankAccount bankAccount = centralBank.createCreditAccountForClient(tinkoff, oldClient, 10000);  
  
 out.println("Write your first name");  
 String firstName = scanner.next();  
 out.println("Write your last name");  
 String lastName = scanner.next();  
 Client client = centralBank.createClient(firstName, lastName);  
 out.println("Do you want to register your address? y/n");  
 String chooseAddress = scanner.next();  
 if (Objects.equals(chooseAddress, "y")) {  
 out.println("Enter your address");  
 String address = scanner.next();  
 centralBank.addClientAddress(client, address);  
 }  
  
 out.println("Do you want to register your passport? y/n");  
 String choosePassport = scanner.next();  
 if (Objects.equals(choosePassport, "y")) {  
 out.println("Enter your passport");  
 String passport = scanner.next();  
 centralBank.addClientAddress(client, passport);  
 }  
  
 out.println("Select the bank where you want to register: ");  
 int i = 0;  
 for (Bank allbank : centralBank.getBanks()) {  
 out.println((i+1) + allbank.getName());  
 }  
 String enteredBank = scanner.next();  
 Bank bank = centralBank.getBanks().stream().filter(banks -> banks.getName().equals(enteredBank)).findAny().orElse(null);  
 centralBank.addClientBank(client, bank);  
 out.println("Select the account where you want to register: " +  
 "1) Debit" +  
 "2) Deposit" +  
 "3) Credit" +  
 "Write one number");  
 BankAccount card = null;  
 String result = scanner.next();  
 int resultFor;  
 if (Objects.equals(result, "1")) {  
 out.println("Enter how much do you want to put in your debit account?");  
 resultFor = scanner.nextInt();  
 int money = resultFor;  
 card = centralBank.createDebitAccountForClient(bank, client, money);  
 }  
 else if (Objects.equals(result, "2"))  
 {  
 out.println("Enter how much do you want to put in your deposit account?");  
 resultFor = scanner.nextInt();  
 int money = resultFor;  
 card = centralBank.createDepositAccountForClient(bank, client, money);  
 }  
 else if (Objects.equals(result, "3"))  
 {  
 out.println("Enter how much do you want to recieve for your credit account?");  
 resultFor = scanner.nextInt();  
 int money = resultFor;  
 card = centralBank.createCreditAccountForClient(bank, client, money);  
 }  
  
 out.println("Do you want to transaction to someone?(y/n)");  
 String res = scanner.next();  
 if (Objects.equals(res, "y"))  
 {  
 out.println("How much money do you want to transaction?");  
 double moneyTrans = scanner.nextDouble();  
 Transaction transaction = card.makeTransaction(client, card, bankAccount, moneyTrans);  
 }  
 else if (Objects.equals(res, "n"))  
 {  
 out.println("Thanks for you choice. Bye Bye!!!");  
 }  
  
 out.println("Money in your card");  
 out.println(card.getMoney());  
 out.println("Thanks for you choice. Bye Bye!!!");  
 }  
}