Министерство образования Республики Беларусь

Министерство науки и высшего образования Российской Федерации

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»»

**Лабораторная работа № 7. Удаленный вызов метода (RMI)**

Выполнил: студент гр.ИСИТ-191

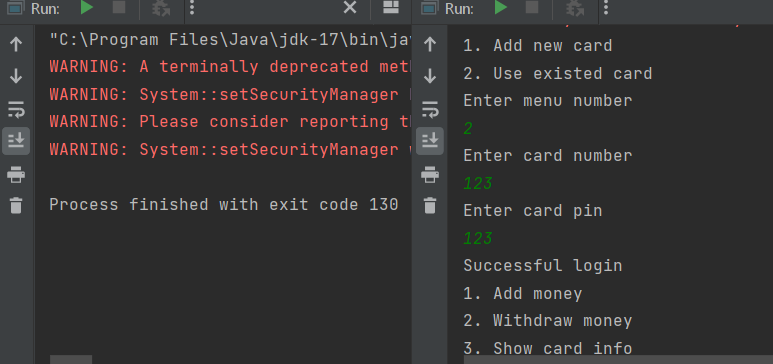
Харкевич А. В.

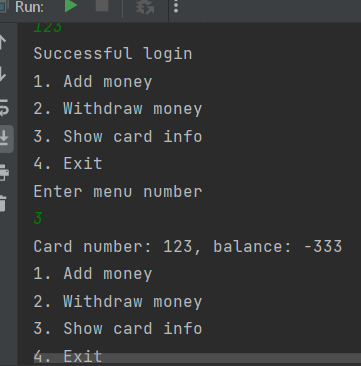
Проверил: Денисевич Д. А.

Могилев, 2021

**Цель работы:** реализовать приложение, используя технологию RMI.

**Задание для самостоятельной работы:** Создадать систему, функционально состоящую из двух компонентов – сервера (процессингового центра) и клиента (касса). Предполагается, что сервер в системе один (именно он обладает всей информацией о зарегистрированных картах и их балансах), а касс много и на них проходят операции регистрации новых карт, а также операции изменения баланса карт (соответственно – оплаты и занесения наличных).





**Client.java**

package lab7;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;

import java.rmi.RMISecurityManager;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class Client {

public static void main(String[] args) throws RemoteException, NotBoundException, MalformedURLException {

System.setProperty("java.security.policy", "D:\\Artem\\Desktop\\psp\\lab7\\lab7\\src\\.java.policy");

System.setSecurityManager(new RMISecurityManager());

Registry registry = LocateRegistry.getRegistry("127.0.0.1", 9095);

CardService cardService = (CardService) registry.lookup("CardServiceRemote");

Scanner sc = new Scanner(System.in);

String cardNumber = null;

String cardPin = null;

System.out.print("1. Add new card \n");

System.out.print("2. Use existed card \n");

System.out.print("Enter menu number\n");

int menu = sc.nextInt();

switch (menu) {

case 1: {

System.out.print("Enter card number\n");

sc.nextLine();

cardNumber = sc.nextLine();

System.out.print("Enter card pin\n");

cardPin = sc.nextLine();

if (cardService.createCard(cardNumber, cardPin)) {

System.out.print("Successful\n");

} else {

System.out.print("Card is existed\n");

return;

}

break;

}

case 2: {

System.out.print("Enter card number\n");

sc.nextLine();

cardNumber = sc.nextLine();

System.out.print("Enter card pin\n");

cardPin = sc.nextLine();

if (cardService.isCardCorrect(cardNumber, cardPin)) {

System.out.print("Successful login\n");

} else {

System.out.print("Invalid credentials\n");

return;

}

break;

}

}

Boolean isWork = true;

while (isWork) {

System.out.print("1. Add money\n");

System.out.print("2. Withdraw money\n");

System.out.print("3. Show card info \n");

System.out.print("4. Exit \n");

System.out.print("Enter menu number\n");

menu = sc.nextInt();

switch (menu) {

case 1: {

System.out.print("Enter money that you want to add\n");

cardService.addMoney(cardNumber, sc.nextInt());

break;

}

case 2: {

System.out.print("Enter money that you want to withdraw\n");

cardService.withdrawMoney(cardNumber, sc.nextInt());

break;

}

case 3: {

System.out.print(cardService.showInfo(cardNumber) + "\n");

break;

}

case 4: {

isWork = false;

break;

}

}

}

}

}

**Server.java**

package lab7;

import java.rmi.AlreadyBoundException;

import java.rmi.RMISecurityManager;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

public class Server {

public static void main(String[] args) throws RemoteException, AlreadyBoundException {

System.setProperty( "java.security.policy","D:\\Artem\\Desktop\\psp\\lab7\\lab7\\src\\.java.policy");

System.setSecurityManager(new RMISecurityManager());

CardService cardServiceObj = new CardServiceRemote();

CardService stub = (CardService) UnicastRemoteObject.exportObject(cardServiceObj, 0);

Registry registry = LocateRegistry.createRegistry(9095);

registry.bind("CardServiceRemote", stub);

}

}

**Card.java**

package lab7;

import java.io.Serializable;

public class Card implements Serializable {

private String number;

private String pinCode;

private int balance;

public Card(String number, String pinCode, int balance){

this.number = number;

this.pinCode = pinCode;

this.balance = balance;

}

public void setNumber(String number){

this.number = number;

}

public void setPinCode(String pinCode){

this.pinCode = pinCode;

}

public void setBalance(int balance){

this.balance = balance;

}

public String getNumber(){

return this.number;

}

public String getPinCode(){

return this.pinCode;

}

public int getBalance(){

return this.balance;

}

}

**CardService.java**

package lab7;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface CardService extends Remote {

public void addMoney(String cardNumber, int money) throws RemoteException;

public void withdrawMoney(String cardNumber, int money) throws RemoteException;

public String showInfo(String cardNumber) throws RemoteException;

public Boolean isCardCorrect(String cardNumber, String cardPin) throws RemoteException;

public Boolean createCard(String cardNumber, String cardPin) throws RemoteException;

}

**CardServiceRemote.java**

package lab7;

import java.rmi.RemoteException;

public class CardServiceRemote implements CardService{

@Override

public void addMoney(String cardNumber, int money) throws RemoteException {

Card card = getCard(cardNumber);

if(card == null){

return;

}

card.setBalance(card.getBalance() + money);

}

@Override

public void withdrawMoney(String cardNumber, int money) throws RemoteException {

Card card = getCard(cardNumber);

if(card == null){

return;

}

card.setBalance(card.getBalance() - money);

}

@Override

public String showInfo(String cardNumber) throws RemoteException {

Card card = getCard(cardNumber);

if(card == null){

return "Card is null\n";

}

return "Card number: " + card.getNumber() + ", balance: " + card.getBalance();

}

@Override

public Boolean isCardCorrect(String cardNumber, String cardPin) throws RemoteException {

for (Card card:Storage.cards) {

if (card.getNumber().equals(cardNumber) && card.getPinCode().equals(cardPin)){

return true;

}

}

return false;

}

@Override

public Boolean createCard(String cardNumber, String cardPin) throws RemoteException {

for (Card card: Storage.cards) {

if (card.getNumber().equals(cardNumber)) {

return false;

}

}

Storage.cards.add(new Card(cardNumber, cardPin, 0));

return true;

}

private Card getCard(String cardNumber){

for (Card card:Storage.cards) {

if(card.getNumber().equals(cardNumber)){

return card;

}

}

return null;

}

}

**Storage.java**

package lab7;

import java.util.ArrayList;

public class Storage {

public static ArrayList<Card> cards = new ArrayList<Card>() {

{

add(new Card("1111", "1111", 0));

add(new Card("2222", "1234", 200));

add(new Card("3333", "0000", 100));

}

};

}