МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського “ХАІ”

Кафедра 603

**Домашнє завдання**

з дисципліни “Програмування Інтернет”

ХАІ.603.631п.15В.050103.126337.ПЗ

Виконав студент гр.631п  Кошиль В. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (№ групи) (П.І.Б)

(підпис, дата)

Перевірив:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наукова ступінь, вчене звання)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, дата) (П.І.Б)

2015

**Введение**

Данная работа содержит методические рекомендации по выполнению домашнего задания по теме: «Объектно-ориентированное проектирование и программирование на языке Java». Выполнение домашнего задания включает проектирование и реализацию на языке программирования Java автономного программного приложения предназначенного для моделирования свойств и поведения объектов информационной системы заданной предметной области.

**Цель данной работы:** Закрепление теоретического материала и получение практических навыков при проектировании и реализации объектно-ориентированного приложения средствами языка Java.

**Вариант 10**

Разработать информационно-справочную систему расчета по коммунальным платежам.

**Порядок выполнения работы**

1. Для выполнения домашней работы было проанализировано задание и разработана концептуальная модель приложения. Модель: это будет консольное приложение (т.к. среда разработки Eclipse не обладает нормальным редактором форм) предназначенное для регистрации плательщиков, добавления им платежей, хранения их и их платежей в памяти. А так же редактирование и удаление плательщиков и их платежей. Программа будет работать с файлами (сохранение/загрузка списка плательщиков). Программа будет взаимодействовать с пользователем в режиме диалога. Программа предназначена для администраторов банков.
2. Входящие данные для добавления нового клиента в список: имя, фамилия отчество плательщика. Для добавления платежа клиенту: тип, размер, дата, дополнительная информация и id клиента.
3. Выходящие данные: подсказки и сообщения о результатах действий в консоли. Также информация о всех клиентах.
4. Требования к ПО:

* добавление нового клиента
* удаление клиента
* редактирование данных клиента
* добавление платежа клиенту
* удаление платежа
* редактирование платежа
* удаление всех платежей
* удаление всех клиентов
* просмотр информации обо всех клиентах
* сохранение базы данных
* загрузка базы данных

1. Согласно разработанной модели приложения были разработаны такие классы:

* Payment – сущность платежа. Содержит такие поля:

string Type – тип платежа

float Size – размер платежа

String Info – дополнительная инфа

Date Date – дата платежа

int ID – номер платежа

int PersonID – номер клиента

* Person – сущность клиента. Содержит список платежей и методы для работы с ними. Поля:

string Name – имя клиента

string Surname – фамилия клиента

string SecondName – отчество клиента

int ID – уникальный айди нашего сайта.

* Collection – коллекция клиентов, выполняет роль базы данных. Содержит методы для добавления, удаления, сохранения списка.
* ReadWrite – класс для реализации сохранения и загрузки из текстового файла.
* MainClass – основной класс приложения, который объединяет все классы и содержит вызовы.

**Классы**

Здесь приводятся дата сущности приложения:

Payment.java

**import** java.util.Date;

**public** **class** Payment {

**private** String Type;

**private** **float** Size;

**private** Date Date;

**private** String Info;

**public** **int** ID;

**public** **int** personID;

**public** String getType(){

**return** **this**.Type;

}

**public** **void** setType(String type){

**this**.Type = type;

}

**public** **float** getSize(){

**return** **this**.Size;

}

**public** **void** setSize(**float** size){

**this**.Size = size;

}

**public** Date getDate(){

**return** **this**.Date;

}

**public** **void** setDate(Date date){

**this**.Date = date;

}

**public** String getInfo(){

**return** **this**.Info;

}

**public** **void** setInfo(String info){

**this**.Info = info;

}

}

Persom.java

**import** java.util.Vector;

**import** java.util.Date;

**public** **class** Person{

**private** String FirstName;

**private** String SecondName;

**private** String Surname;

**private** Vector<Payment> Payments = **new** Vector<Payment>();

**public** **int** ID;

**public** String getFirstName(){

**return** **this**.FirstName;

}

**public** **void** setFirstName(String firstName){

**this**.FirstName = firstName;

}

**public** String getSecondName(){

**return** **this**.SecondName;

}

**public** **void** setSecondName(String secondName){

**this**.SecondName = secondName;

}

**public** String getSurname(){

**return** **this**.Surname;

}

**public** **void** setSurname(String surname){

**this**.Surname = surname;

}

**public** **int** getPaymentCount(){

**return** **this**.Payments.size();

}

**public** Vector<Payment> getPayments(){

**return** **this**.Payments;

}

**public** **void** setPayments(Vector<Payment> payments){

**this**.Payments = payments;

}

**public** **void** AddPayment(String type, **float** size, Date date, String info, **int** paymentId, **int** personId){

Payment pay = **new** Payment();

pay.ID = paymentId;

pay.personID = personId;

pay.setType(type);

pay.setSize(size);

pay.setDate(date);

pay.setInfo(info);

**this**.Payments.addElement(pay);

}

**public** **void** DeletePayment(**int** id){

**for**(Payment p : **this**.Payments){

**if**(p.ID == id){

**this**.Payments.remove(p);

}

}

}

**public** **void** EditPayment(String type, **float** size, Date date, String info, **int** paymentId){

**for**(Payment p : **this**.Payments){

**if**(p.ID == paymentId){

p.setType(type);

p.setSize(size);

p.setDate(date);

p.setInfo(info);

}

}

}

**public** **void** RemoveAllPayments(){

**this**.Payments.clear();

}

}

**Тестирование приложения**

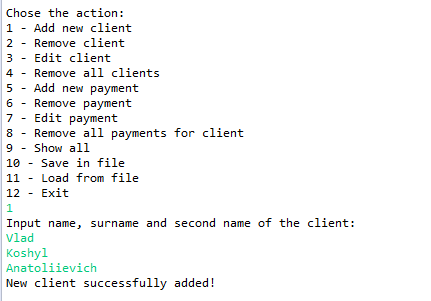


Рисунок 1 – Добавление нового клиента

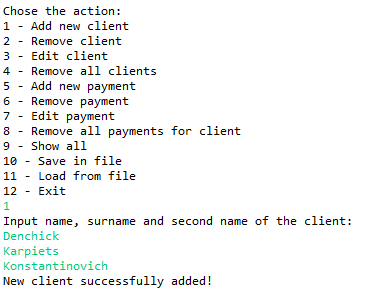


Рисунок 2 – Добавление другого клиента

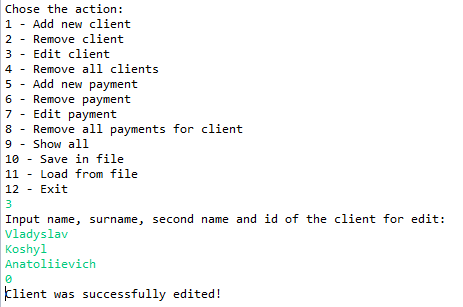


Рисунок 3 – Редактирование данных о клиенте

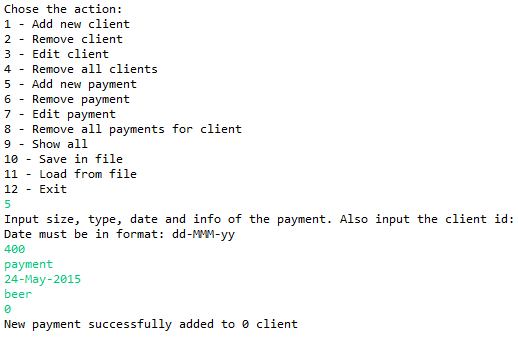


Рисунок 4 – Добавление платежа

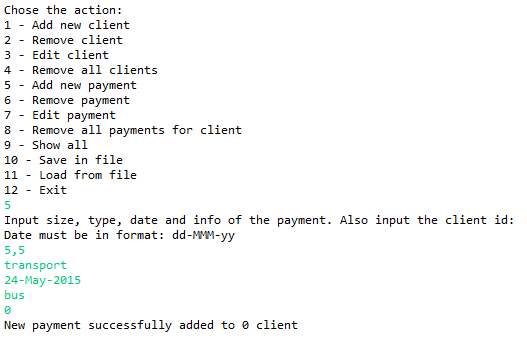


Рисунок 5 – Добавление еще одного платежа

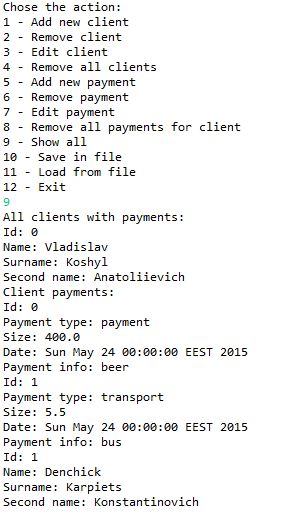


Рисунок 6 – Вывод содержимого коллекции

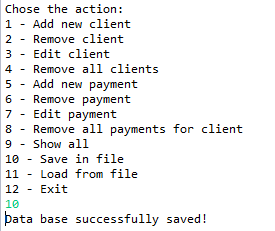


Рисунок 7 – Сохранение коллекции в текстовый файл

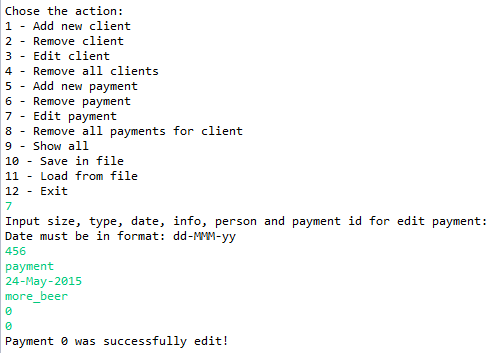


Рисунок 8 – Изменение платежа

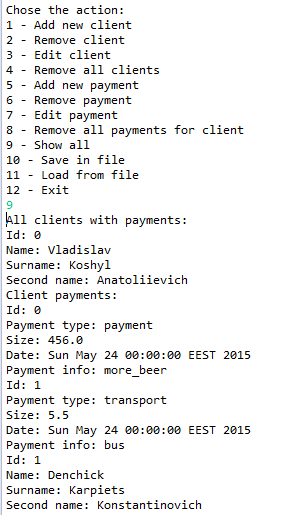


Рисунок 9 – Вывод измененных данных

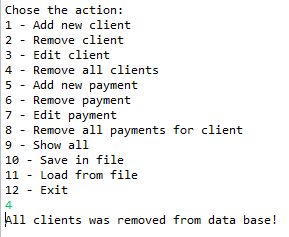


Рисунок 10 – Очистка коллекции

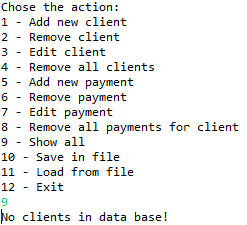


Рисунок 11 – Проверка коллекции

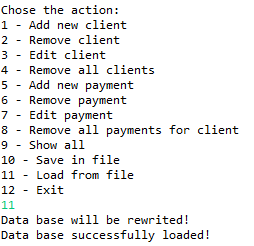


Рисунок 12 – Загрузка коллекции

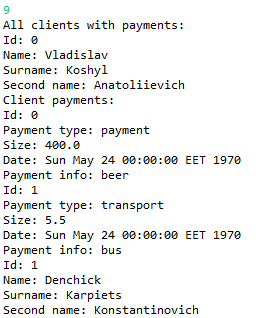


Рисунок 13 – Проверка загрузки

**Выводы**

Данная работа позволила закрепить теоретический материал и получить практические навыки при проектировании и реализации объектно-ориентированного приложения средствами языка Java. Было спроектировано

и реализовано приложение для управления платежами.

**Исходный код приложения**

Класс Payment.java

**import** java.util.Date;

**public** **class** Payment {

**private** String Type;

**private** **float** Size;

**private** Date Date;

**private** String Info;

**public** **int** ID;

**public** **int** personID;

**public** String getType(){

**return** **this**.Type;

}

**public** **void** setType(String type){

**this**.Type = type;

}

**public** **float** getSize(){

**return** **this**.Size;

}

**public** **void** setSize(**float** size){

**this**.Size = size;

}

**public** Date getDate(){

**return** **this**.Date;

}

**public** **void** setDate(Date date){

**this**.Date = date;

}

**public** String getInfo(){

**return** **this**.Info;

}

**public** **void** setInfo(String info){

**this**.Info = info;

}

}

Класс Person.java

**import java.util.Vector;**

**import java.util.Date;**

**public class Person{**

**private String FirstName;**

**private String SecondName;**

**private String Surname;**

**private Vector<Payment> Payments = new Vector<Payment>();**

**public int ID;**

**public String getFirstName(){**

**return this.FirstName;**

**}**

**public void setFirstName(String firstName){**

**this.FirstName = firstName;**

**}**

**public String getSecondName(){**

**return this.SecondName;**

**}**

**public void setSecondName(String secondName){**

**this.SecondName = secondName;**

**}**

**public String getSurname(){**

**return this.Surname;**

**}**

**public void setSurname(String surname){**

**this.Surname = surname;**

**}**

**public int getPaymentCount(){**

**return this.Payments.size();**

**}**

**public Vector<Payment> getPayments(){**

**return this.Payments;**

**}**

**public void setPayments(Vector<Payment> payments){**

**this.Payments = payments;**

**}**

**public void AddPayment(String type, float size, Date date, String info, int paymentId, int personId){**

**Payment pay = new Payment();**

**pay.ID = paymentId;**

**pay.personID = personId;**

**pay.setType(type);**

**pay.setSize(size);**

**pay.setDate(date);**

**pay.setInfo(info);**

**this.Payments.addElement(pay);**

**}**

**public void DeletePayment(int id){**

**for(Payment p : this.Payments){**

**if(p.ID == id){**

**this.Payments.remove(p);**

**}**

**}**

**}**

**public void EditPayment(String type, float size, Date date, String info, int paymentId){**

**for(Payment p : this.Payments){**

**if(p.ID == paymentId){**

**p.setType(type);**

**p.setSize(size);**

**p.setDate(date);**

**p.setInfo(info);**

**}**

**}**

**}**

**public void RemoveAllPayments(){**

**this.Payments.clear();**

**}**

**}**

Класс Collection.java

import java.io.FileNotFoundException;

import java.io.IOException;

import java.text.ParseException;

import java.util.Vector;

public class Collection {

private Vector<Person> Persons = new Vector<Person>();

public void SaveCollection(String path){

ReadWrite fileManager = new ReadWrite(path);

fileManager.SaveCollection(this.Persons);

}

public Vector<Person> LoadCollection(String path) throws FileNotFoundException, IOException, ParseException{

ReadWrite fileManager = new ReadWrite(path);

return fileManager.LoadCollection();

}

public void AddPerson(String firstName, String secondName, String surname, int personId){

Person p = new Person();

p.ID = personId;

p.setFirstName(firstName);

p.setSecondName(secondName);

p.setSurname(surname);

this.Persons.addElement(p);

}

public void Delete(Person person){

if(this.Persons.size() != 0){

this.Persons.remove(person);

}

}

public void Delete(int id){

for(Person p : this.Persons){

if(p.ID==id){

this.Persons.remove(p);

}

}

}

public Person GetPerson(int personId){

for(Person p : this.Persons){

if(p.ID == personId){

return p;

}

}

return null; //search nothing

}

public void EditPerson(String firstName, String secondName, String surname, int id){

for(Person p : this.Persons){

if(p.ID == id){

p.setFirstName(firstName);

p.setSecondName(secondName);

p.setSurname(surname);

}

}

}

public void ReplacePerson(Person person){

for(int i = 0; i < this.Persons.size(); i++){

if(this.Persons.elementAt(i).ID == person.ID){

this.Persons.set(i, person);

}

}

}

public void ClearCollection(){

this.Persons.clear();

}

public Vector<Person> getPersons() {

return this.Persons;

}

public void setPersons(Vector<Person> persons) {

this.Persons = persons;

}

public int Size(){

return this.Persons.size();

}

}

Класс ReadWrite.java

import java.io.\*;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Locale;

import java.util.Vector;

public class ReadWrite{

private String path;

public ReadWrite(String path){

this.path = path;

}

public void SaveCollection(Vector<Person> collection){

Writer writer = null;

try {

writer = new FileWriter(this.path);

for (int i = 0; i<collection.size(); i++) {

writer.write("Client");

writer.write("\n");

writer.write(Integer.toString(collection.elementAt(i).ID));

writer.write("\n");

writer.write(collection.elementAt(i).getFirstName());

writer.write("\n");

writer.write(collection.elementAt(i).getSurname());

writer.write("\n");

writer.write(collection.elementAt(i).getSecondName());

writer.write("\n");

Vector<Payment> payments = collection.elementAt(i).getPayments();

for(Payment p : payments){

writer.write("Payment");

writer.write("\n");

writer.write(Integer.toString(p.ID));

writer.write("\n");

writer.write(p.getType());

writer.write("\n");

writer.write(Float.toString(p.getSize()));

writer.write("\n");

writer.write(p.getDate().toString());

writer.write("\n");

writer.write(p.getInfo());

writer.write("\n");

writer.write(Integer.toString(p.personID));

writer.write("\n");

}

}

writer.flush();

}

catch (IOException e) {

System.err.println(e);

}

finally {

if (writer != null) {

try {

writer.close();

}

catch (IOException ex) {

}

}

}

}

public Vector<Person> LoadCollection() throws FileNotFoundException, IOException, ParseException{

Vector<Person> collection = new Vector<Person>();

File file = new File(this.path);

BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(file)));

int id=0;

String name="";

String surName="";

String secondName="";

String line = null;

while ((line = br.readLine()) != null){

if(line.equals("Client")){

id = Integer.parseInt(br.readLine());

name = line = br.readLine();

surName = line = br.readLine();

secondName = line = br.readLine();

Person client = new Person();

client.ID=id;

client.setFirstName(name);

client.setSecondName(secondName);

client.setSurname(surName);

collection.addElement(client);

}

if(line.equals("Payment")){

int pId = Integer.parseInt(br.readLine());

String type = br.readLine();

float size = Float.parseFloat(br.readLine());

String date = br.readLine();

DateFormat format = new SimpleDateFormat("EE MMM dd HH:mm:ss", Locale.ENGLISH);

Date realDate = format.parse(date);

String info = br.readLine();

int persId = Integer.parseInt(br.readLine());

for(Person p : collection){

if(p.ID == persId){

p.AddPayment(type, size, realDate, info, pId, persId);

}

}

}

}

br.close();

return collection;

}

}

Класс MainClass.java

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.text.DateFormat;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** java.util.Locale;

**import** java.util.Scanner;

**import** java.util.Vector;

**public** **class** MainClass {

**private** **static** Scanner *scan*;

**public** **static** **void** main(String[] args) **throws** ParseException, FileNotFoundException, IOException {

Collection dataBase = **new** Collection();

**int** choose = 0;

*scan* = **new** Scanner(System.***in***);

String path = "D:/SkyDrive/3 курс/!2 cemester/!COMPLETED/Java/HomeWork/data.txt";

//ReadWrite readerWriter = null;

//Thread fileThread = null;

**int** clientId = 0;

**int** paymentId = 0;

**while**(**true**){

System.***out***.println("\nChose the action:");

System.***out***.println("1 - Add new client");

System.***out***.println("2 - Remove client");

System.***out***.println("3 - Edit client");

System.***out***.println("4 - Remove all clients");

System.***out***.println("5 - Add new payment");

System.***out***.println("6 - Remove payment");

System.***out***.println("7 - Edit payment");

System.***out***.println("8 - Remove all payments for client");

System.***out***.println("9 - Show all");

System.***out***.println("10 - Save in file");

System.***out***.println("11 - Load from file");

System.***out***.println("12 - Exit");

choose = *scan*.nextInt();

**switch**(choose){

**case** 1:{

System.***out***.println("Input name, surname and second name of the client:");

String name = *scan*.next();

String surname = *scan*.next();

String secondName = *scan*.next();

dataBase.AddPerson(name, secondName, surname, clientId);

System.***out***.println("New client successfully added!");

clientId++;

**break**;

}

**case** 2:{

System.***out***.println("Input client id for removed from data base:");

**int** id = *scan*.nextInt();

//Person p = dataBase.GetPerson(id);

dataBase.Delete(id);

System.***out***.println("Client with id " +id+ " was removed from data base!");

**break**;

}

**case** 3:{

System.***out***.println("Input name, surname, second name and id of the client for edit:");

String newName = *scan*.next();

String newSurname = *scan*.next();

String newSecondName = *scan*.next();

**int** personId = *scan*.nextInt();

dataBase.EditPerson(newName, newSecondName, newSurname, personId);

//dataBase.SetPerson(editClient);

System.***out***.println("Client was successfully edited!");

**break**;

}

**case** 4:{

System.***out***.println("All clients was removed from data base!");

dataBase.ClearCollection();

**break**;

}

**case** 5:{

**if**(dataBase.Size() != 0){

System.***out***.println("Input size, type, date and info of the payment. Also input the client id:");

System.***out***.println("Date must be in format: dd-MMM-yy");

**float** size = *scan*.nextFloat();

String type = *scan*.next();

String date = *scan*.next();

DateFormat format = **new** SimpleDateFormat("dd-MMM-yy", Locale.***ENGLISH***);

Date realDate = format.parse(date);

String info = *scan*.next();

**int** personId = *scan*.nextInt();

Person person = dataBase.GetPerson(personId);

person.AddPayment(type, size, realDate, info, paymentId, personId);

dataBase.ReplacePerson(person);

paymentId++;

System.***out***.println("New payment successfully added to " + personId + " client");

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 6:{

**if**(dataBase.Size() != 0){

System.***out***.println("Input payment id and person id for deleting payment:");

**int** payId = *scan*.nextInt();

**int** personId = *scan*.nextInt();

Person person = dataBase.GetPerson(personId);

person.DeletePayment(payId);

dataBase.ReplacePerson(person);

System.***out***.println("Payment " + paymentId + " successfully deleted from person " + personId);

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 7:{

**if**(dataBase.Size() != 0){

System.***out***.println("Input size, type, date, info, person and payment id for edit payment:");

System.***out***.println("Date must be in format: dd-MMM-yy");

**float** newSize = *scan*.nextFloat();

String newType = *scan*.next();

String newDate = *scan*.next();

String newInfo = *scan*.next();

**int** newPersonId = *scan*.nextInt();

**int** newPaymentId = *scan*.nextInt();

DateFormat format = **new** SimpleDateFormat("dd-MMM-yy", Locale.***ENGLISH***);

Date realDate = format.parse(newDate);

Person person = dataBase.GetPerson(newPersonId);

person.EditPayment(newType, newSize, realDate, newInfo, newPaymentId);

dataBase.ReplacePerson(person);

System.***out***.println("Payment " + newPaymentId + " was successfully edit!");

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 8:{

**if**(dataBase.Size() != 0){

System.***out***.println("Input person id and all his payments will be removed");

**int** personId = *scan*.nextInt();

Person person = dataBase.GetPerson(personId);

person.RemoveAllPayments();

dataBase.ReplacePerson(person);

System.***out***.println("All payments for " + personId + " person was removed successfully!");

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 9:{

**if**(dataBase.Size() != 0){

System.***out***.println("All clients with payments:");

Vector<Person> persons = dataBase.getPersons();

**for**(Person p : persons){

System.***out***.println("Id: " + p.ID);

System.***out***.println("Name: " + p.getFirstName());

System.***out***.println("Surname: " + p.getSurname());

System.***out***.println("Second name: " + p.getSecondName());

Vector<Payment> payments = p.getPayments();

**if**(payments.size() != 0){

System.***out***.println("Client payments:");

**for**(Payment pt : payments){

System.***out***.println("Id: " + pt.ID);

System.***out***.println("Payment type: " + pt.getType());

System.***out***.println("Size: " + pt.getSize());

System.***out***.println("Date: " + pt.getDate());

System.***out***.println("Payment info: " + pt.getInfo());

}

}

}

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 10:{

**if**(dataBase.Size() != 0){

System.***out***.println("Data base successfully saved!");

dataBase.SaveCollection(path);

}

**else**{

System.***out***.println("No clients in data base!");

}

**break**;

}

**case** 11:{

**if**(dataBase.Size() != 0)

System.***out***.println("Data base will be rewrited!");

Vector<Person> pers = dataBase.LoadCollection(path);

dataBase.ClearCollection();

dataBase.setPersons(pers);

System.***out***.println("Data base successfully loaded!");

**break**;

}

**case** 12:

System.*exit*(0);

}

}

}

}