**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине: «Визуальные средства разработки программных приложений»

на тему: «Потоки ввода-вывода.»

Выполнил: студент гр. ИТИ-31

Ковшаров Г. Ю.  
Принял: преподаватель

Ястребов А. А.

Гомель 2024

**Цель работы**: изучить методы работы с потоками ввода и вывода.

**Задание:**

Выполнить задания, сохраняя объекты приложения в одном или нескольких файлах с применением механизма сериализации. Объекты могут содержать поля, помеченные как static, а также transient.

14. **Система Заказ гостиницы**. **Клиент** оставляет **Заявку** на **Номер**, ука-

зав количество мест в номере, класс апартаментов и время пребывания.

**Администратор** рассматривает **Заявку**, подтверждает или отклоняет

её. Результат просматривает **Клиент**. В случае подтверждения **Заявки**

**Клиент** оплачивает услуги.

**Ход работы:**

Были сделаны классы и методы, работающие с этими классами для сериализации и десиарелизации объектов приложения, для системы заказов гостиницы. Результат работы программы представлен на рисунке 1.

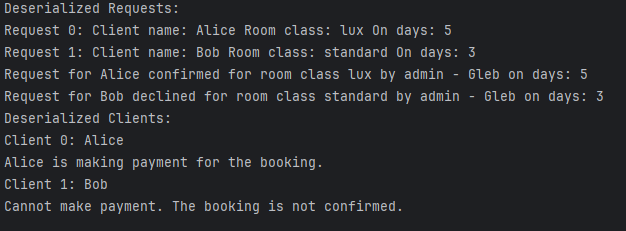


Рисунок 1 – Результат работы программы

Текст разработанных классов на языке *Java* представлен в приложении А.

**Выводы:** в ходе проделанной работы получены навыки работы с потоками ввода и вывода. Также было создано приложение реализующее сериализацию и десериализацию объектов приложения.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст классов**

**Administrator.java:**

import entities.Administrator;

import entities.BookingRequest;

import entities.Client;

import java.io.\*;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

var admin = new Administrator("Gleb");

var firstClient = new Client("Alice", "password123");

var secondClient = new Client("Bob", "securepass");

var firstBookingRequest = firstClient.createBookingRequest(3, "lux", 5);

var secondBookingRequest = secondClient.createBookingRequest(2, "standard", 3);

try {

var fileOutputStreamForBookingRequest = new FileOutputStream("booking\_requests.ser");

var bookingRequests = new ArrayList<BookingRequest>();

bookingRequests.add(firstBookingRequest);

bookingRequests.add(secondBookingRequest);

writeObjectsToFile(fileOutputStreamForBookingRequest, bookingRequests);

fileOutputStreamForBookingRequest.close();

var savedBookingRequests = getBookingRequests();

System.out.println("Deserialized Requests:");

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

var currentBookingRequest = savedBookingRequests.get(i);

System.out.println("Request " + i + ": Client name: " + currentBookingRequest.getClient().getName()

+ " Room class: " + currentBookingRequest.getRoomClass() + " On days: " + currentBookingRequest.getDays());

}

var processedRequests = admin.processRequests(savedBookingRequests);

var fileOutputStreamForClients = new FileOutputStream("clients.ser");

var clients = new ArrayList<Client>();

clients.add(firstClient);

clients.add(secondClient);

writeObjectsToFile(fileOutputStreamForClients, clients);

fileOutputStreamForClients.close();

var savedClients = getClients();

System.out.println("Deserialized Clients:");

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

var currentClient = savedClients.get(i);

System.out.println("Client " + i + ": " + currentClient.getName());

currentClient.makePayment(processedRequests.get(i));

}

} catch (IOException | ClassNotFoundException e) {

System.out.println(e.getMessage());

}

}

private static ArrayList<BookingRequest> getBookingRequests() throws IOException, ClassNotFoundException {

var fileInputStreamForBookingRequest = new FileInputStream("booking\_requests.ser");

var objectInputStreamForBookingRequest = new ObjectInputStream(fileInputStreamForBookingRequest);

var firstSavedBookingRequest = (BookingRequest) objectInputStreamForBookingRequest.readObject();

var secondSavedBookingRequest = (BookingRequest) objectInputStreamForBookingRequest.readObject();

objectInputStreamForBookingRequest.close();

fileInputStreamForBookingRequest.close();

var savedBookingRequests = new ArrayList<BookingRequest>();

savedBookingRequests.add(firstSavedBookingRequest);

savedBookingRequests.add(secondSavedBookingRequest);

return savedBookingRequests;

}

private static ArrayList<Client> getClients() throws IOException, ClassNotFoundException {

var fileInputStreamForClients = new FileInputStream("clients.ser");

var objectInputStreamForClients = new ObjectInputStream(fileInputStreamForClients);

var firstSavedClient = (Client) objectInputStreamForClients.readObject();

var secondSavedClient = (Client) objectInputStreamForClients.readObject();

objectInputStreamForClients.close();

fileInputStreamForClients.close();

var savedClients = new ArrayList<Client>();

savedClients.add(firstSavedClient);

savedClients.add(secondSavedClient);

return savedClients;

}

private static <T> void writeObjectsToFile(FileOutputStream fileOutputStream, ArrayList<T> objects) {

try {

var objectOutputStreamForBookingRequest = new ObjectOutputStream(fileOutputStream);

for (var object : objects) {

objectOutputStreamForBookingRequest.writeObject(object);

}

objectOutputStreamForBookingRequest.close();

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

**BookingRequest.java:**

package entities;

import java.io.Serializable;

public class BookingRequest implements Serializable {

private final Client client;

private int places;

private final String roomClass;

private int days;

private boolean confirmed;

private boolean paid;

public BookingRequest(Client client, int places, String roomClass, int days) {

this.client = client;

this.places = places;

this.roomClass = roomClass;

this.days = days;

}

public Client getClient() {

return client;

}

public boolean isConfirmed() {

return confirmed;

}

public String getRoomClass() {

return this.roomClass;

}

public void setConfirmed(boolean confirmed) {

this.confirmed = confirmed;

}

public void setPaid(boolean paid) {

this.paid = paid;

}

public int getDays() {

return days;

}

}

**Client.java:**

package entities;

import java.io.Serializable;

public class Client implements Serializable {

private String name;

private transient String password;

public Client(String name, String password) {

this.name = name;

this.password = password;

}

public String getName() {

return name;

}

public BookingRequest createBookingRequest(int places, String roomClass, int days) {

return new BookingRequest(this, places, roomClass, days);

}

public void makePayment(BookingRequest request) {

if (request.isConfirmed()) {

System.out.println(name + " is making payment for the booking.");

request.setPaid(true);

} else {

System.out.println("Cannot make payment. The booking is not confirmed.");

}

}

}

**Main.java:**

import entities.Administrator;

import entities.BookingRequest;

import entities.Client;

import java.io.\*;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

var admin = new Administrator("Gleb");

var firstClient = new Client("Alice", "password123");

var secondClient = new Client("Bob", "securepass");

var firstBookingRequest = firstClient.createBookingRequest(3, "lux", 5);

var secondBookingRequest = secondClient.createBookingRequest(2, "standard", 3);

try {

var fileOutputStreamForBookingRequest = new FileOutputStream("booking\_requests.ser");

var bookingRequests = new ArrayList<BookingRequest>();

bookingRequests.add(firstBookingRequest);

bookingRequests.add(secondBookingRequest);

writeObjectsToFile(fileOutputStreamForBookingRequest, bookingRequests);

fileOutputStreamForBookingRequest.close();

var savedBookingRequests = getBookingRequests();

System.out.println("Deserialized Requests:");

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

var currentBookingRequest = savedBookingRequests.get(i);

System.out.println("Request " + i + ": Client name: " + currentBookingRequest.getClient().getName()

+ " Room class: " + currentBookingRequest.getRoomClass() + " On days: " + currentBookingRequest.getDays());

}

var processedRequests = admin.processRequests(savedBookingRequests);

var fileOutputStreamForClients = new FileOutputStream("clients.ser");

var clients = new ArrayList<Client>();

clients.add(firstClient);

clients.add(secondClient);

writeObjectsToFile(fileOutputStreamForClients, clients);

fileOutputStreamForClients.close();

var savedClients = getClients();

System.out.println("Deserialized Clients:");

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

var currentClient = savedClients.get(i);

System.out.println("Client " + i + ": " + currentClient.getName());

currentClient.makePayment(processedRequests.get(i));

}

} catch (IOException | ClassNotFoundException e) {

System.out.println(e.getMessage());

}

}

private static ArrayList<BookingRequest> getBookingRequests() throws IOException, ClassNotFoundException {

var fileInputStreamForBookingRequest = new FileInputStream("booking\_requests.ser");

var objectInputStreamForBookingRequest = new ObjectInputStream(fileInputStreamForBookingRequest);

var firstSavedBookingRequest = (BookingRequest) objectInputStreamForBookingRequest.readObject();

var secondSavedBookingRequest = (BookingRequest) objectInputStreamForBookingRequest.readObject();

objectInputStreamForBookingRequest.close();

fileInputStreamForBookingRequest.close();

var savedBookingRequests = new ArrayList<BookingRequest>();

savedBookingRequests.add(firstSavedBookingRequest);

savedBookingRequests.add(secondSavedBookingRequest);

return savedBookingRequests;

}

private static ArrayList<Client> getClients() throws IOException, ClassNotFoundException {

var fileInputStreamForClients = new FileInputStream("clients.ser");

var objectInputStreamForClients = new ObjectInputStream(fileInputStreamForClients);

var firstSavedClient = (Client) objectInputStreamForClients.readObject();

var secondSavedClient = (Client) objectInputStreamForClients.readObject();

objectInputStreamForClients.close();

fileInputStreamForClients.close();

var savedClients = new ArrayList<Client>();

savedClients.add(firstSavedClient);

savedClients.add(secondSavedClient);

return savedClients;

}

private static <T> void writeObjectsToFile(FileOutputStream fileOutputStream, ArrayList<T> objects) {

try {

var objectOutputStreamForBookingRequest = new ObjectOutputStream(fileOutputStream);

for (var object : objects) {

objectOutputStreamForBookingRequest.writeObject(object);

}

objectOutputStreamForBookingRequest.close();

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}