**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 №6

по дисциплине: «Визуальные средства разработки программных приложений»

на тему: «Обработка данных с использованием XML»

Выполнил: студент гр. ИТИ-31

Ковшаров Г. Ю.  
Принял: преподаватель

Ястребов А. А.

Гомель 2024

**Цель работы**: изучить методы обработки данных с использованием XML в Java.

**Задание:**

Выполнить задания, сохраняя объекты приложения в одном или нескольких файлах XML. Четные варианты используют SAX-парсер, нечетные DOM-парсер.

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 в XML, для системы заказов гостиницы. Результат работы программы представлен на рисунке 1.

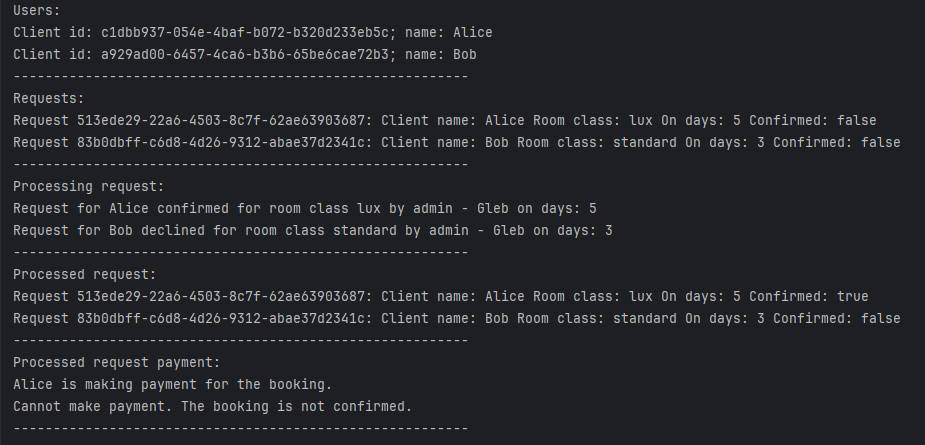


Рисунок 1 – Результат работы программы

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 в XML.

**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.util.UUID;

import java.io.Serializable;

public class BookingRequest implements Serializable {

private String id;

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) {

UUID uuid = UUID.randomUUID();

this.id = uuid.toString();

this.client = client;

this.places = places;

this.roomClass = roomClass;

this.days = days;

}

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

this.id = id;

this.client = client;

this.places = places;

this.roomClass = roomClass;

this.days = days;

}

public Client getClient() {

return client;

}

public int getPlaces() {

return places;

}

public boolean isConfirmed() {

return confirmed;

}

public String getRoomClass() {

return this.roomClass;

}

public String getId() {

return id;

}

public void setConfirmed(boolean confirmed) {

this.confirmed = confirmed;

}

public void setPaid(boolean paid) {

this.paid = paid;

}

public boolean isPaid() {

return paid;

}

public int getDays() {

return days;

}

public void makePayment() {

if (isConfirmed()) {

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

setPaid(true);

} else {

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

}

}

}

**Client.java:**

package entities;

import java.util.UUID;

import java.io.Serializable;

public class Client implements Serializable {

private String id;

private String name;

public Client(String name) {

UUID uuid = UUID.randomUUID();

this.id = uuid.toString();

this.name = name;

}

public Client(String id, String name) {

this.id = id;

this.name = name;

}

public String getName() {

return name;

}

public String getId() {

return id;

}

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

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

}

}

**Main.java:**

import entities.Administrator;

import entities.BookingRequest;

import entities.Client;

import xmlHelpers.parsers.BookingRequestSAXParser;

import xmlHelpers.parsers.ClientSAXParser;

import xmlHelpers.writers.XMLWriter;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.TransformerException;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) throws ParserConfigurationException, TransformerException {

XMLWriter xmlWriter = new XMLWriter();

var admin = new Administrator("Gleb");

var firstClient = new Client("Alice");

var secondClient = new Client("Bob");

var clients = new ArrayList<Client>();

clients.add(firstClient);

clients.add(secondClient);

xmlWriter.saveClientsToXml(clients, "D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\clients.xml");

var clientsFromXml = ClientSAXParser.parseClients("D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\clients.xml");

System.out.println("Users:");

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

var currentClient = clientsFromXml.get(i);

System.out.println("Client id: " + currentClient.getId() + "; name: " + currentClient.getName());

}

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

var firstBookingRequest = clientsFromXml.get(0).createBookingRequest(3, "lux", 5);

var secondBookingRequest = clientsFromXml.get(1).createBookingRequest(2, "standard", 3);

var bookingRequests = new ArrayList<BookingRequest>();

bookingRequests.add(firstBookingRequest);

bookingRequests.add(secondBookingRequest);

xmlWriter.saveBookingRequestsToXml(bookingRequests, "D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\booking-requests.xml");

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

var bookingRequestsFromXml = BookingRequestSAXParser.parseBookingRequests("D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\booking-requests.xml");

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

var currentBookingRequest = bookingRequestsFromXml.get(i);

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

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

+ currentBookingRequest.getDays() + " Confirmed: " + currentBookingRequest.isConfirmed());

}

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

System.out.println("Processing request:");

var processedRequests = admin.processRequests(bookingRequests);

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

xmlWriter.saveBookingRequestsToXml(processedRequests, "D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\processed-booking-requests.xml");

var processedBookingRequestsFromXml = BookingRequestSAXParser.parseBookingRequests("D:\\ProgrammingAndProjects\\Studies\\5sem\\VSRPP\\lab6\\processed-booking-requests.xml");

System.out.println("Processed request:");

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

var currentBookingRequest = processedBookingRequestsFromXml.get(i);

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

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

+ currentBookingRequest.getDays() + " Confirmed: " + currentBookingRequest.isConfirmed());

}

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

System.out.println("Processed request payment:");

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

var currentBookingRequest = processedBookingRequestsFromXml.get(i);

currentBookingRequest.makePayment();

}

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

}

}

**XMLWriter.java:**

package xmlHelpers.writers;

import entities.BookingRequest;

import entities.Client;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import java.util.List;

public class XMLWriter {

public void saveClientsToXml(List<Client> clients, String filePath) throws ParserConfigurationException, TransformerException {

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document document = builder.newDocument();

Element rootElement = document.createElement("Clients");

document.appendChild(rootElement);

for (Client client : clients) {

Element clientElement = createClientElement(client, document);

rootElement.appendChild(clientElement);

}

writeDocumentToFile(document, filePath);

}

private Element createClientElement(Client client, Document document) {

Element clientElement = document.createElement("Client");

Element clientId = document.createElement("Id");

clientId.appendChild(document.createTextNode(client.getId()));

Element clientName = document.createElement("Name");

clientName.appendChild(document.createTextNode(client.getName()));

clientElement.appendChild(clientId);

clientElement.appendChild(clientName);

return clientElement;

}

public void saveBookingRequestsToXml(List<BookingRequest> requests, String filePath) throws ParserConfigurationException, TransformerException {

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document document = builder.newDocument();

Element rootElement = document.createElement("BookingRequests");

document.appendChild(rootElement);

for (BookingRequest request : requests) {

Element requestElement = createBookingRequestElement(request, document);

rootElement.appendChild(requestElement);

}

writeDocumentToFile(document, filePath);

}

private Element createBookingRequestElement(BookingRequest request, Document document) {

Element requestElement = document.createElement("BookingRequest");

Element requestId = document.createElement("Id");

requestId.appendChild(document.createTextNode(request.getId()));

Element clientElement = createClientElement(request.getClient(), document);

requestElement.appendChild(clientElement);

Element places = document.createElement("Places");

places.appendChild(document.createTextNode(String.valueOf(request.getPlaces())));

Element roomClass = document.createElement("RoomClass");

roomClass.appendChild(document.createTextNode(request.getRoomClass()));

Element days = document.createElement("Days");

days.appendChild(document.createTextNode(String.valueOf(request.getDays())));

Element confirmed = document.createElement("Confirmed");

confirmed.appendChild(document.createTextNode(String.valueOf(request.isConfirmed())));

Element paid = document.createElement("Paid");

paid.appendChild(document.createTextNode(String.valueOf(request.isPaid())));

requestElement.appendChild(requestId);

requestElement.appendChild(places);

requestElement.appendChild(roomClass);

requestElement.appendChild(days);

requestElement.appendChild(confirmed);

requestElement.appendChild(paid);

return requestElement;

}

private void writeDocumentToFile(Document document, String filePath) throws TransformerException {

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource domSource = new DOMSource(document);

StreamResult streamResult = new StreamResult(new File(filePath));

transformer.transform(domSource, streamResult);

}

}

**BookingRequestSAXParser.java:**

package xmlHelpers.parsers;

import entities.BookingRequest;

import entities.Client;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import java.util.ArrayList;

import java.util.List;

public class BookingRequestSAXParser extends DefaultHandler {

private List<BookingRequest> bookingRequests = new ArrayList<>();

private String currentId;

private Client currentClient;

private int currentPlaces;

private String currentRoomClass;

private int currentDays;

private boolean currentConfirmed;

private boolean currentPaid;

private StringBuilder currentValue = new StringBuilder();

public List<BookingRequest> getBookingRequests() {

return bookingRequests;

}

@Override

public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {

currentValue.setLength(0);

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

currentClient = null;

}

}

@Override

public void characters(char[] ch, int start, int length) throws SAXException {

currentValue.append(new String(ch, start, length));

}

@Override

public void endElement(String uri, String localName, String qName) throws SAXException {

switch (qName) {

case "Id":

currentId = currentValue.toString();

break;

case "Name":

currentClient = new Client(currentId, currentValue.toString());

break;

case "Places":

currentPlaces = Integer.parseInt(currentValue.toString());

break;

case "RoomClass":

currentRoomClass = currentValue.toString();

break;

case "Days":

currentDays = Integer.parseInt(currentValue.toString());

break;

case "Confirmed":

currentConfirmed = Boolean.parseBoolean(currentValue.toString());

break;

case "Paid":

currentPaid = Boolean.parseBoolean(currentValue.toString());

break;

case "BookingRequest":

BookingRequest bookingRequest = new BookingRequest(currentId, currentClient, currentPlaces, currentRoomClass, currentDays);

bookingRequest.setConfirmed(currentConfirmed);

bookingRequest.setPaid(currentPaid);

bookingRequests.add(bookingRequest);

break;

}

}

public static List<BookingRequest> parseBookingRequests(String xmlFilePath) {

try {

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser saxParser = factory.newSAXParser();

BookingRequestSAXParser handler = new BookingRequestSAXParser();

saxParser.parse(xmlFilePath, handler);

return handler.getBookingRequests();

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}

**ClientSAXParser.java:**

package xmlHelpers.parsers;

import entities.Client;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import java.util.ArrayList;

import java.util.List;

public class ClientSAXParser extends DefaultHandler {

private List<Client> clients = new ArrayList<>();

private String currentId;

private String currentName;

private StringBuilder currentValue = new StringBuilder();

public List<Client> getClients() {

return clients;

}

@Override

public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {

currentValue.setLength(0);

}

@Override

public void characters(char[] ch, int start, int length) throws SAXException {

currentValue.append(new String(ch, start, length));

}

@Override

public void endElement(String uri, String localName, String qName) throws SAXException {

switch (qName) {

case "Id":

currentId = currentValue.toString();

break;

case "Name":

currentName = currentValue.toString();

break;

case "Client":

Client client = new Client(currentId, currentName);

clients.add(client);

break;

}

}

public static List<Client> parseClients(String xmlFilePath) {

try {

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser saxParser = factory.newSAXParser();

ClientSAXParser handler = new ClientSAXParser();

saxParser.parse(xmlFilePath, handler);

return handler.getClients();

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}