Лабораторна робота №10

Тема: Серіалізація. Логування. Документування коду. XML та JSON парсери.

Мета роботи: практика роботи з XML та JSON парсерами, використання серіалізації, логування та докусентування коду.

Хід роботи:

Завдання 1. Створити maven Java проект java_lab_10 з пакетом com.education.ztu. Додати в проект код з лабораторної роботи №3 з пакету game. Для реалізації завдань додати необхідні залежності в файл pom.xml.

Завдання 2. Серіалізація:

- Додати до сутностей в пакеті game serialVersionUID (згенерувати за допомогою Inteliij IDEA)
- Виключити деякі поля з серіалізації на власний розсуд (використати ключове слово transient)
- Серіалізувати та десеріалізувати сутності.

Лістинг програми:

```
package com.education.ztu.game;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.*;


/**
 * Клас Game відповідає за логіку гри.
 */
public class Game {
    private static final Logger logger = LoggerFactory.getLogger(Game.class);


    /**
     * Головний метод, який виконує основну логіку гри:
     * 1. Створює учасників і команди.
     * 2. Серіалізує і десеріалізує команди.
     * 3. Логує важливі події гри.
     *
     * @param args аргументи командного рядка (не використовуються).
     */
    public static void main(String[] args) {
        Schoolar schoolar1 = new Schoolar("Ivan", 13);
        Schoolar schoolar2 = new Schoolar("Mariya", 15);
        Student student1 = new Student("Mykola", 20);
        Student student2 = new Student("Viktoria", 21);
```

```
        logger.info("Teams are going to play against each other");
        scollarTeam.playWith(scollarTeam2);

        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("teams.ser"))) {
            oos.writeObject(scollarTeam);
            oos.writeObject(studentTeam);
            oos.writeObject(employeeTeam);
            logger.info("Teams serialized successfully!");
        } catch (IOException e) {
            logger.error("Error serializing teams", e);
        }

        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("teams.ser"))) {
            Team<Schoolar> scollarTeamDeserialized = (Team<Schoolar>)
ois.readObject();
            Team<Student> studentTeamDeserialized = (Team<Student>)
ois.readObject();
            Team<Employee> employeeTeamDeserialized = (Team<Employee>)
ois.readObject();

            logger.info("Deserialized teams:");
            logger.info(scollarTeamDeserialized.toString());
            logger.info(studentTeamDeserialized.toString());
            logger.info(employeeTeamDeserialized.toString());
        } catch (IOException | ClassNotFoundException e) {
            logger.error("Error deserializing teams", e);
        }
    }
}
```

```
package com.education.ztu.game;

import java.io.Serial;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Team<T extends Participant> implements Cloneable, Serializable {
    @Serial
    private static final long serialVersionUID = -5917625402739212672L;
    private String name;
    private transient List<T> participants = new ArrayList<>();

    public Team(String name) {
        this.name = name;
    }

    public void addNewParticipant(T participant) {
        participants.add(participant);
        System.out.println("To the team " + name + " was added participant " +
participant.getName());

    }

    public void playWith(Team<T> team) {
        System.out.println("Team " + this.name + " is playing with team " +
team.name);
        String winnerName;
        Random random = new Random();
        winnerName = random.nextInt(2) == 0 ? this.name : team.name;
```

```
        System.out.println("The team  " + winnerName + " is winner!");
    }

    public Team<T> cloneTeam() {
        Team<T> clonedTeam = new Team<>(this.name);
        for (T participant : participants) {
            clonedTeam.addNewParticipant(participant);
        }
        return clonedTeam;
    }

    @Override
    public String toString() {
        return "Team{name='" + name + "', participants=" + participants + '}';
    }

}
```

Завдання 3. Логування:

- Додати логування до коду в пакеті game. Використати бібліотеки Log4J, SLF4J.
- Вивести логи в консоль та в файл.
- Використати різні рівні логування (trace, debug, info, warn, error, fatal)

```
• <?xml version="1.0" encoding="UTF-8"?>
   <project xmlns="http://maven.apache.org/POM/4.0.0"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
       <modelVersion>4.0.0</modelVersion>

       <groupId>com.education.ztu</groupId>
       <artifactId>lab10</artifactId>
       <version>1.0-SNAPSHOT</version>

       <properties>
           <maven.compiler.source>17</maven.compiler.source>
           <maven.compiler.target>17</maven.compiler.target>
           <project.build.sourceEncoding>UTF-
   8</project.build.sourceEncoding>
       </properties>
       <dependencies>

           <dependency>
               <groupId>org.apache.logging.log4j</groupId>
               <artifactId>log4j-core</artifactId>
               <version>2.20.0</version>
           </dependency>

           <dependency>
               <groupId>org.apache.logging.log4j</groupId>
               <artifactId>log4j-slf4j-impl</artifactId>
               <version>2.20.0</version>
           </dependency>


           <dependency>
               <groupId>com.google.code.gson</groupId>
               <artifactId>gson</artifactId>
               <version>2.8.8</version>
```

```
        </dependency>

    </dependencies>
</project>
```

Результат роботи програми:

```
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Teams created successfully!
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Teams are going to play against each other
Team Dragon is playing with team Rozumnyky
The team Rozumnyky is winner!
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Teams serialized successfully!
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Deserialized teams:
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Team{name='Dragon', participants=null}
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Team{name='Vpered', participants=null}
2024-12-14 22:56:32 INFO  com.education.ztu.game.Game - Team{name='Robotyagi', participants=null}
```

Рис.1

Завдання 4. Документування коду:

- Додати документаційні коментарі до коду в пакеті game
- Згенерувати документацію (щоб згенерувати JavaDoc у Inteliij IDEA необхідно натиснути Tools –> Generate JavaDoc –> вказати шлях, куди зберегти документацію)

```
/**
 * Клас Game відповідає за логіку гри.
 */
public class Game {
    private static final Logger logger = LoggerFactory.getLogger(Game.class);  9 usages


    /**
     * Головний метод, який виконує основну логіку гри:
     * 1. Створює учасників і команди.
     * 2. Серіалізує і десеріалізує команди.
     * 3. Логує важливі події гри.
     *
     * @param args аргументи командного рядка (не використовуються).
     */
    public static void main(String[] args) {
```

Рис.2

Завдання 5. XML парсери:

- Реалізувати читання та збереження XML файлу використовуючи DOM парсер.
- XML файл використати будь який за бажанням

Лістинг програми:

```
package com.education.ztu.game;

import org.w3c.dom.Element;
import org.w3c.dom.Node;
```

```
import org.w3c.dom.NodeList;
import org.w3c.dom.Document;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;

public class XMLParser {
    public void parseXML(String filePath) {
        try {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File(filePath));

            document.getDocumentElement().normalize();

            NodeList students = document.getElementsByTagName("student");

            for (int i = 0; i < students.getLength(); i++) {
                Node studentNode = students.item(i);

                if (studentNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element student = (Element) studentNode;
                    String name =
student.getElementsByTagName("name").item(0).getTextContent();
                    String age =
student.getElementsByTagName("age").item(0).getTextContent();

                    System.out.println("Name: " + name);
                    System.out.println("Age: " + age);
                    System.out.println();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    public void saveXML(String filePath) {
        try {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.newDocument();

            Element school = document.createElement("school");
            document.appendChild(school);

            Element student = document.createElement("student");
            school.appendChild(student);

            Element name = document.createElement("name");
            name.appendChild(document.createTextNode("Anna"));
            student.appendChild(name);

            Element age = document.createElement("age");
            age.appendChild(document.createTextNode("17"));
            student.appendChild(age);
```

```
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(new File(filePath));
            transformer.transform(source, result);

            System.out.println("XML file saved successfully!");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        XMLParser parser = new XMLParser();
        parser.parseXML("students.xml");
        parser.saveXML("new_students.xml");
    }
}
```

Результат виконання програми:

```
Name: Olivia
Age: 18

Name: Oleg
Age: 19

XML file saved successfully!
```

Рис.3

Завдання 6. JSON парсер:

• Провести перетворення сутностей з Java в JSON і навпаки з JSON в Java (використайте бібліотеки Gson або Jackson) Сутності для перетворень виберіть на власний розсуд

Лістинг програми:

```
package com.education.ztu.game;

import com.google.gson.Gson;

public class PersonJSON {
    private String name;
    private int age;

    public PersonJSON(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
```

```
    }

    public int getAge() {
        return age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person{name='" + name + "', age=" + age + "}";
    }

    public static void main(String[] args) {
        PersonJSON person = new PersonJSON("Tetiana", 18);
        Gson gson = new Gson();
        String json = gson.toJson(person);
        System.out.println("Java to JSON: " + json);
        PersonJSON newPerson = gson.fromJson(json, PersonJSON.class);
        System.out.println("JSON to Java: " + newPerson);
    }
}
```

Результат виконання програми:

```
Java to JSON: {"name":"Tetiana","age":18}
JSON to Java: Person{name='Tetiana', age=18}
```

Рис.4

Висновок: під час виконання лабораторної роботи я попрактикувала роботу з XML та JSON парсерами, використала серіалізації, логування та докусентування коду.