МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА «ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ІМЕНІ А. В. ДАБАГЯНА»

ЗВІТ

З лабораторної роботи №3

«РОЗШИРЕНІ МОЖЛИВОСТІ РОБОТИ З ФАЙЛАМИ»

ВИКОНАВ:

студент групи КН-422ч

Максим БЕЛОШИЦЬКИЙ

ПЕРЕВІРИВ:

Асистент каф. ПІІТУ

Олексій КОНДРАТОВ

Харків – 2023

**ЗМІСТ**

[**МЕТА** 3](#_Toc166631470)

[Завдання 1.1 4](#_Toc166631471)

[Код завдання 1.1: 4](#_Toc166631472)

[TextFileWriter.java 4](#_Toc166631473)

[XmlFileWriter.java 6](#_Toc166631474)

[JsonFileWriter.java 7](#_Toc166631475)

[App.java 8](#_Toc166631476)

[pom.xml 10](#_Toc166631477)

[Тестування за допомогою JUnit: 12](#_Toc166631478)

[TextFileWriterTest.java 12](#_Toc166631479)

[XmlFileWriterTest.java 14](#_Toc166631480)

[JsonFileWriterTest.java 15](#_Toc166631481)

[Результати виконання програми 18](#_Toc166631482)

[Завдання 1.2 21](#_Toc166631483)

[Код завдання 1.2: 21](#_Toc166631484)

[FileLister.java 21](#_Toc166631485)

[Тестування за допомогою JUnit: 22](#_Toc166631486)

[FileListerTest.java 22](#_Toc166631487)

[Результати виконання програми 25](#_Toc166631488)

[Завдання 1.3 26](#_Toc166631489)

[Код завдання 1.3: 26](#_Toc166631490)

[TextFileProccesor.java 26](#_Toc166631491)

[Тестування за допомогою JUnit: 27](#_Toc166631492)

[TextFileProccesorTest.java 27](#_Toc166631493)

[Результати виконання програм 29](#_Toc166631494)

[Завдання 1.4 30](#_Toc166631495)

[Код завдання 1.4: 30](#_Toc166631496)

[XmlTask.java 30](#_Toc166631497)

[Тестування за допомогою JUnit: 33](#_Toc166631498)

[XmlTaskTest.java 33](#_Toc166631499)

[Результати виконання програми 34](#_Toc166631500)

[Завдання 1.5 36](#_Toc166631501)

[Код завдання 1.5: 36](#_Toc166631502)

[JsonTask.java 36](#_Toc166631503)

[Тестування за допомогою JUnit: 39](#_Toc166631504)

[JsonTaskTest.java 39](#_Toc166631505)

[Результати виконання програми 40](#_Toc166631506)

[Завдання 1.6 41](#_Toc166631507)

[Код завдання 1.6: 41](#_Toc166631508)

[SAXParserExample.java 41](#_Toc166631509)

[DOMParserExample.java 43](#_Toc166631510)

[Тестування за допомогою JUnit: 44](#_Toc166631511)

[SAXParserExampleTest.java 44](#_Toc166631512)

[DOMParserExampleTest.java 45](#_Toc166631513)

[Результати виконання програми 46](#_Toc166631514)

[Завдання для самоконтролю 48](#_Toc166631515)

[Код завдання 2.1: 48](#_Toc166631516)

[Результати виконання програми 49](#_Toc166631517)

[Код завдання 2.2: 49](#_Toc166631518)

[Результати виконання програми 50](#_Toc166631519)

[Код завдання 2.3: 51](#_Toc166631520)

[Результати виконання програми 52](#_Toc166631521)

[Код завдання 2.4: 52](#_Toc166631522)

[Результати виконання програми 53](#_Toc166631523)

[Код завдання 2.5: 54](#_Toc166631524)

[Результати виконання програми 56](#_Toc166631525)

[**ВИСНОВОК** 57](#_Toc166631526)

# **МЕТА**

Мета лабораторної роботи №3 полягає у вивченні та вдосконаленні навичок роботи з файловою системою в мові програмування Java. Це включає в себе розуміння принципів обробки даних з використанням різних методів, таких як Stream API, а також здатність до роботи з текстовими файлами та виведенням даних у різних форматах, включаючи XML та JSON. Також важливо засвоїти навички тестування програмних компонентів за допомогою JUnit, а також знання процесу серіалізації та десеріалізації об'єктів для збереження та відновлення даних. В результаті виконання цієї роботи студент має отримати глибше розуміння принципів роботи з файлами та вміти застосовувати ці знання у власних програмах.

# Завдання 1.1

Створити новий Maven-проєкт, в який перенести раніше створені класи, які представляють сутності індивідуальних завдань лабораторних робіт [№ 3](http://iwanoff.inf.ua/java_ua/LabTraining03.html) і [№ 4](http://iwanoff.inf.ua/java_ua/LabTraining04.html) курсу "Основи програмування Java".

Програма повинна демонструвати:

* відтворення реалізації завдань лабораторних робіт № 3 і № 4 курсу "Основи програмування Java";
* використання засобів Stream API для обробки та виведення послідовностей;
* виведення даних в текстовий файл засобами Stream API з подальшим читанням;
* серіалізацію об'єктів у XML-файл і JSON-файл і відповідну десеріалізацію із застосуванням бібліотеки XStream;
* запис подій, пов'язаних з виконанням програми, в системний журнал;
* тестування окремих класів з використанням JUnit.

Для представлення індивідуальних даних слід використати класи, які були створені під час виконання індивідуального завдання лабораторної роботи № 1 цього курсу.

## Код завдання 1.1:

### TextFileWriter.java

package *edu.lab02\_ind*;

import *java.io.BufferedReader*;

import *java.io.BufferedWriter*;

import *java.io.FileReader*;

import *java.io.FileWriter*;

import *java.io.IOException*;

import *java.util.ArrayList*;

import *java.util.List*;

*/\*\**

*\* The TextFileWriter class provides methods to write and read data from a text file.*

*\*/*

*public* *class* TextFileWriter {

*/\*\**

*\* Writes the data of conferences to a text file.*

*\**

*\* @param conferences The list of conferences to write.*

*\* @param fileName    The name of the file to write the data to.*

*\*/*

*public* *static* *void* writeData(*List*<*ConferenceWithStreams*> *conferences*, *String* *fileName*) {

        try (*BufferedWriter* writer = new BufferedWriter(new FileWriter(fileName))) {

            for (*Conference* conference : conferences) {

                writer.write("Conference: " + conference.getName() + " (" + conference.getPlace() + ")");

                writer.newLine();

                writer.write("Meetings:");

                writer.newLine();

*Meeting*[] meetings = ((ConferenceWithStreams) conference).searchMeetingsInAlphabetOrder();

                for (*Meeting* meeting : meetings) {

                    writer.write("- Date: " + meeting.getDate() + ", Topic: " + meeting.getTopic() +

                            ", Participants: " + meeting.getNumberOfParticipants());

                    writer.newLine();

                }

                writer.newLine();

            }

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

*/\*\**

*\* Reads the data of conferences from a text file.*

*\**

*\* @param fileName The name of the file to read the data from.*

*\* @return The list of conferences read from the file.*

*\*/*

*public* *static* *List*<*ConferenceWithStreams*> readData(*String* *fileName*) {

*List*<*ConferenceWithStreams*> conferences = new *ArrayList*<>();

        try (*BufferedReader* reader = new BufferedReader(new FileReader(fileName))) {

*String* line;

*ConferenceWithStreams* conference = null;

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

                if (line.startsWith("Conference: ")) {

*String*[] parts = line.split(" \\(");

*String* name = parts[0].substring("Conference: ".length());

*String* place = parts[1].substring(0, parts[1].length() - 1);

                    conference = new ConferenceWithStreams(name, place, new *ArrayList*<>());

                    conferences.add(conference);

                } else if (line.equals("Meetings:")) {

*List*<*Meeting*> meetings = new *ArrayList*<>();

                    while ((line = reader.readLine()) != null && !line.isEmpty()) {

*String*[] meetingInfo = line.split(", ");

*String* date = meetingInfo[0].substring("Date: ".length());

*String* topic = meetingInfo[1].substring("Topic: ".length());

*int* participants = Integer.parseInt(meetingInfo[2].substring("Participants: ".length()));

                        meetings.add(new Meeting(date, topic, participants));

                    }

                }

            }

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

        return conferences;

    }

}

### XmlFileWriter.java

package *edu.lab02\_ind*;

import *java.io.FileWriter*;

import *java.io.IOException*;

import *java.util.List*;

import *com.thoughtworks.xstream.XStream*;

import *com.thoughtworks.xstream.io.xml.DomDriver*;

*/\*\**

*\* The XmlFileWriter class provides a utility for writing data to an XML file.*

*\*/*

*public* *class* XmlFileWriter {

*/\*\**

*\* Writes the given list of conferences to an XML file.*

*\**

*\* @param conferences the list of conferences to write*

*\* @param fileName the name of the XML file to write to*

*\*/*

*public* *static* *void* writeData(*List*<*ConferenceWithStreams*> *conferences*, *String* *fileName*) {

*XStream* xStream = new XStream(new DomDriver());

        xStream.alias("conference", Conference.class);

        xStream.alias("meeting", Meeting.class);

        try (*FileWriter* writer = new FileWriter(fileName)) {

            writer.write(xStream.toXML(conferences));

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

# JsonFileWriter.java

package *edu.lab02\_ind*;

import *com.google.gson.Gson*;

import *com.google.gson.GsonBuilder*;

import *java.io.FileWriter*;

import *java.io.IOException*;

import *java.util.List*;

*/\*\**

*\* The JsonFileWriter class provides a utility for writing data to a JSON file.*

*\*/*

*public* *class* JsonFileWriter {

*/\*\**

*\* Writes the given list of conferences to a JSON file with the specified file name.*

*\**

*\* @param conferences the list of conferences to be written*

*\* @param fileName the name of the JSON file to write to*

*\*/*

*public* *static* *void* writeData(*List*<*ConferenceWithStreams*> *conferences*, *String* *fileName*) {

        try (*FileWriter* writer = new FileWriter(fileName)) {

*// Create Gson object with pretty printing enabled*

*Gson* gson = new GsonBuilder().setPrettyPrinting().create();

*// Convert conferences list to JSON and write to file*

            gson.toJson(conferences, writer);

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

### App.java

package *edu.lab02\_ind*;

import *java.util.ArrayList*;

import *java.util.Arrays*;

import *java.util.List*;

import *java.util.logging.Level*;

import *java.util.logging.Logger*;

import *java.util.Comparator*;

import *java.util.stream.Collectors*;

*/\*\**

*\* The Main class represents the entry point of the program.*

*\* It contains the main method which creates a list of meetings,*

*\* tests different conference classes, and prints the results.*

*\*/*

*/\*\**

*\* The main class that represents the application.*

*\*/*

*public* *class* App {

*private* *static* *final* *Logger* LOGGER = Logger.getLogger(App.class.getName());

*/\*\**

*\* The main method of the application.*

*\**

*\* @param args The command line arguments.*

*\*/*

*public* *static* *void* main(*String*[] *args*) {

*// Create conferences*

*List*<*ConferenceWithStreams*> conferences = createConferences();

*// Print conference information*

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

        conferences.forEach(System.out::println);

*// Collect all meetings*

*List*<*Meeting*> allMeetings = conferences.stream()

                .flatMap(conf -> Arrays.stream(conf.getMeetings()))

                .collect(Collectors.toList());

*// Print all meetings*

        System.out.println("\nAll meetings:");

        allMeetings.forEach(System.out::println);

*// Sort meetings by number of participants*

        allMeetings.sort(Comparator.comparingInt(Meeting::getNumberOfParticipants));

        System.out.println("\nMeetings sorted by number of participants:");

        allMeetings.forEach(System.out::println);

*// Sort meetings by topic*

        allMeetings.sort(Comparator.comparing(Meeting::getTopic));

        System.out.println("\nMeetings sorted by topic:");

        allMeetings.forEach(System.out::println);

*// Write data to text file*

        TextFileWriter.writeData(conferences, "C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\ind\\target\\files\\conferences.txt");

        System.out.println("\nConferences written to the file.");

*// Read data from the text file*

        TextFileWriter.readData("C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\ind\\target\\files\\conferences.txt");

        System.out.println("\nConferences read from the file:");

        conferences.forEach(System.out::println);

*// Meetings from the file*

        System.out.println("\nMeetings read from the file:");

        conferences.stream()

                .flatMap(conf -> Arrays.stream(conf.getMeetings()))

                .forEach(System.out::println);

*// Write data to XML and JSON files*

        XmlFileWriter.writeData(conferences, "C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\ind\\target\\files\\conferences.xml");

        JsonFileWriter.writeData(conferences, "C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\ind\\target\\files\\conferences.json");

        System.out.println("\nConferences written to XML and JSON files.\n");

*// Log completion*

        LOGGER.log(Level.INFO, "Program execution completed successfully.");

    }

*/\*\**

*\* Creates a list of conferences.*

*\**

*\* @return The list of conferences.*

*\*/*

*private* *static* *List*<*ConferenceWithStreams*> createConferences() {

*List*<*ConferenceWithStreams*> conferences = new *ArrayList*<>();

*// Create the first conference*

*List*<*Meeting*> meetings1 = new *ArrayList*<>();

        meetings1.add(new Meeting("2024-05-20", "AI in Healthcare", 50));

        meetings1.add(new Meeting("2024-05-21", "Machine Learning Applications", 70));

*Conference* conf1 = new ConferenceWithMeetingsListUsingStreams("FutureAI", "Paris", meetings1);

*// Create the second conference*

*List*<*Meeting*> meetings2 = new *ArrayList*<>();

        meetings2.add(new Meeting("2024-06-10", "Data Science Summit", 80));

        meetings2.add(new Meeting("2024-06-11", "Blockchain Conference", 100));

*Conference* conf2 = new ConferenceWithMeetingsListUsingStreams("Data Analysis", "Houston", meetings2);

*// Add conferences to the list*

        conferences.add((ConferenceWithStreams) conf1);

        conferences.add((ConferenceWithStreams) conf2);

        return conferences;

    }

}

### pom.xml

<?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>edu.lab02\_ind</groupId>

  <artifactId>ind</artifactId>

  <version>1.0</version>

  <name>ind</name>

*<!-- FIXME change it to the project's website -->*

  <url>http://www.example.com</url>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>21</maven.compiler.source>

    <maven.compiler.target>21</maven.compiler.target>

  </properties>

  <dependencies>

*<!-- Google-->*

    <dependency>

      <groupId>com.google.code.gson</groupId>

      <artifactId>gson</artifactId>

      <version>2.8.8</version> *<!-- You can use the latest version -->*

    </dependency>

*<!-- JSON -->*

    <dependency>

      <groupId>org.json</groupId>

      <artifactId>json</artifactId>

      <version>20210307</version>

    </dependency>

*<!-- XStream library for XML serialization -->*

    <dependency>

        <groupId>com.thoughtworks.xstream</groupId>

        <artifactId>xstream</artifactId>

        <version>1.4.15</version>

    </dependency>

*<!-- JUnit 5 library for unit testing -->*

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter-api</artifactId>

      <version>5.7.1</version>

      <scope>test</scope>

    </dependency>

*<!-- JUnit library for unit testing -->*

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.11</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <pluginManagement>*<!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->*

      <plugins>

*<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->*

        <plugin>

          <artifactId>maven-clean-plugin</artifactId>

          <version>3.1.0</version>

        </plugin>

*<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->*

        <plugin>

          <artifactId>maven-resources-plugin</artifactId>

          <version>3.2.0</version>

        </plugin>

        <plugin>

          <artifactId>maven-compiler-plugin</artifactId>

          <version>3.11.0</version>

        </plugin>

        <plugin>

          <artifactId>maven-surefire-plugin</artifactId>

          <version>3.0.0-M5</version>

        </plugin>

        <plugin>

          <artifactId>maven-jar-plugin</artifactId>

          <version>3.2.0</version>

        </plugin>

        <plugin>

          <artifactId>maven-install-plugin</artifactId>

          <version>3.0.0-M1</version>

        </plugin>

        <plugin>

          <artifactId>maven-deploy-plugin</artifactId>

          <version>3.0.0-M1</version>

        </plugin>

*<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->*

        <plugin>

          <artifactId>maven-site-plugin</artifactId>

          <version>3.9.0</version>

        </plugin>

        <plugin>

          <artifactId>maven-project-info-reports-plugin</artifactId>

          <version>3.1.1</version>

        </plugin>

      </plugins>

    </pluginManagement>

  </build>

</project>

## Тестування за допомогою JUnit:

### TextFileWriterTest.java

package *edu.lab02\_ind*;

import *static* *org.junit.jupiter.api.Assertions.\**;

import *java.io.File*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

import *java.util.ArrayList*;

import *java.util.List*;

import *org.junit.jupiter.api.AfterEach*;

import *org.junit.jupiter.api.BeforeEach*;

import *org.junit.jupiter.api.Test*;

*class* JsonFileWriterTest {

*private* *static* *final* *String* TEST\_FILE = "test\_conferences.json";

*private* *List*<*ConferenceWithStreams*> conferences;

    @*BeforeEach*

*void* setUp() {

*// Create sample conference data for testing*

        conferences = new *ArrayList*<>();

*List*<*Meeting*> meetings1 = new *ArrayList*<>();

        meetings1.add(new Meeting("2024-05-20", "AI in Healthcare", 50));

*ConferenceWithStreams* conference1 = new ConferenceWithStreams("FutureAI", "Paris", meetings1);

*List*<*Meeting*> meetings2 = new *ArrayList*<>();

        meetings2.add(new Meeting("2024-06-10", "Data Science Summit", 80));

*ConferenceWithStreams* conference2 = new ConferenceWithStreams("Data Analysis", "Houston", meetings2);

        conferences.add(conference1);

        conferences.add(conference2);

    }

    @*AfterEach*

*void* tearDown() {

*// Delete the test file after each test*

*File* file = new File(TEST\_FILE);

        if (file.exists()) {

            file.delete();

        }

    }

    @*Test*

*void* testWriteData() {

*// Write data to the test file*

        JsonFileWriter.writeData(conferences, TEST\_FILE);

*// Check if the file exists*

        assertTrue(Files.exists(Paths.get(TEST\_FILE)), "File should exist");

*// Read the content of the file and verify it*

*String* content = null;

        try {

            content = Files.readString(Paths.get(TEST\_FILE));

        } catch (*IOException* *e*) {

            fail("Failed to read file content: " + e.getMessage());

        }

        assertNotNull(content, "File content should not be null");

*// Verify that the JSON content contains conference names and places*

        assertTrue(content.contains("FutureAI") && content.contains("Paris"), "Conference 1 should be present");

        assertTrue(content.contains("Data Analysis") && content.contains("Houston"), "Conference 2 should be present");

    }

}

### XmlFileWriterTest.java

package *edu.lab02\_ind*;

import *static* *org.junit.jupiter.api.Assertions.\**;

import *java.io.File*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

import *java.util.ArrayList*;

import *java.util.List*;

import *org.junit.jupiter.api.AfterEach*;

import *org.junit.jupiter.api.BeforeEach*;

import *org.junit.jupiter.api.Test*;

*class* XmlFileWriterTest {

*private* *static* *final* *String* TEST\_FILE = "test\_conferences.xml";

*private* *List*<*ConferenceWithStreams*> conferences;

    @*BeforeEach*

*void* setUp() {

*// Create sample conference data for testing*

        conferences = new *ArrayList*<>();

*List*<*Meeting*> meetings1 = new *ArrayList*<>();

        meetings1.add(new Meeting("2024-05-20", "AI in Healthcare", 50));

*ConferenceWithStreams* conference1 = new ConferenceWithStreams("FutureAI", "Paris", meetings1);

*List*<*Meeting*> meetings2 = new *ArrayList*<>();

        meetings2.add(new Meeting("2024-06-10", "Data Science Summit", 80));

*ConferenceWithStreams* conference2 = new ConferenceWithStreams("Data Analysis", "Houston", meetings2);

        conferences.add(conference1);

        conferences.add(conference2);

    }

    @*AfterEach*

*void* tearDown() {

*// Delete the test file after each test*

*File* file = new File(TEST\_FILE);

        if (file.exists()) {

            file.delete();

        }

    }

    @*Test*

*void* testWriteData() {

*// Write data to the test file*

        XmlFileWriter.writeData(conferences, TEST\_FILE);

*// Check if the file exists*

        assertTrue(Files.exists(Paths.get(TEST\_FILE)), "File should exist");

*// Read the content of the file and verify it*

*String* content = null;

        try {

            content = Files.readString(Paths.get(TEST\_FILE));

        } catch (*IOException* *e*) {

            fail("Failed to read file content: " + e.getMessage());

        }

        assertNotNull(content, "File content should not be null");

*// Verify that the XML content contains conference names and places*

        assertTrue(content.contains("FutureAI") && content.contains("Paris"), "Conference 1 should be present");

        assertTrue(content.contains("Data Analysis") && content.contains("Houston"), "Conference 2 should be present");

    }

}

### JsonFileWriterTest.java

package *edu.lab02\_ind*;

import *static* *org.junit.jupiter.api.Assertions.\**;

import *java.io.File*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

import *java.util.ArrayList*;

import *java.util.List*;

import *org.junit.jupiter.api.AfterEach*;

import *org.junit.jupiter.api.BeforeEach*;

import *org.junit.jupiter.api.Test*;

*class* JsonFileWriterTest {

*private* *static* *final* *String* TEST\_FILE = "test\_conferences.json";

*private* *List*<*ConferenceWithStreams*> conferences;

    @*BeforeEach*

*void* setUp() {

*// Create sample conference data for testing*

        conferences = new *ArrayList*<>();

*List*<*Meeting*> meetings1 = new *ArrayList*<>();

        meetings1.add(new Meeting("2024-05-20", "AI in Healthcare", 50));

*ConferenceWithStreams* conference1 = new ConferenceWithStreams("FutureAI", "Paris", meetings1);

*List*<*Meeting*> meetings2 = new *ArrayList*<>();

        meetings2.add(new Meeting("2024-06-10", "Data Science Summit", 80));

*ConferenceWithStreams* conference2 = new ConferenceWithStreams("Data Analysis", "Houston", meetings2);

        conferences.add(conference1);

        conferences.add(conference2);

    }

    @*AfterEach*

*void* tearDown() {

*// Delete the test file after each test*

*File* file = new File(TEST\_FILE);

        if (file.exists()) {

            file.delete();

        }

    }

    @*Test*

*void* testWriteData() {

*// Write data to the test file*

        JsonFileWriter.writeData(conferences, TEST\_FILE);

*// Check if the file exists*

        assertTrue(Files.exists(Paths.get(TEST\_FILE)), "File should exist");

*// Read the content of the file and verify it*

*String* content = null;

        try {

            content = Files.readString(Paths.get(TEST\_FILE));

        } catch (*IOException* *e*) {

            fail("Failed to read file content: " + e.getMessage());

        }

        assertNotNull(content, "File content should not be null");

*// Verify that the JSON content contains conference names and places*

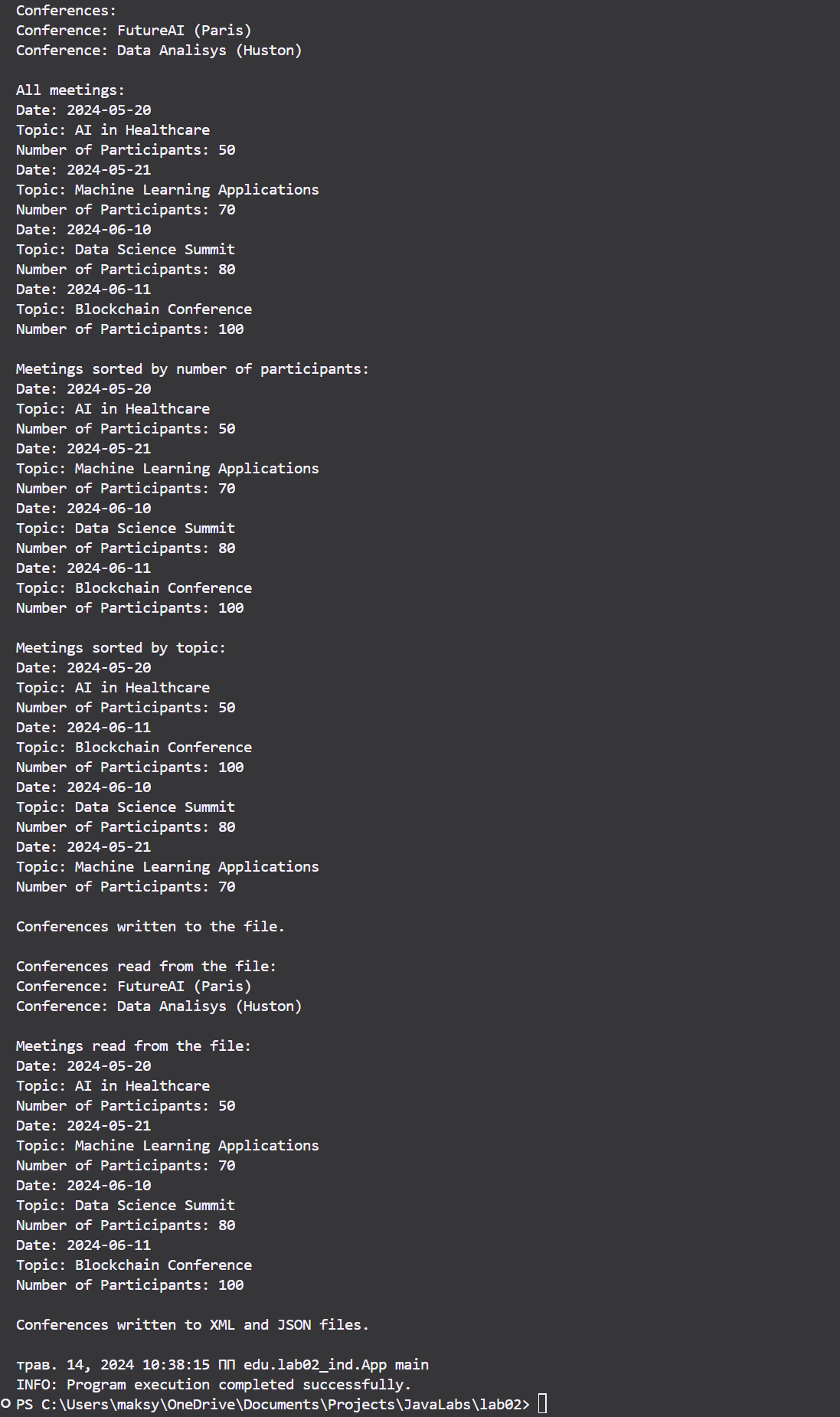
        assertTrue(content.contains("FutureAI") && content.contains("Paris"), "Conference 1 should be present");

        assertTrue(content.contains("Data Analysis") && content.contains("Houston"), "Conference 2 should be present");

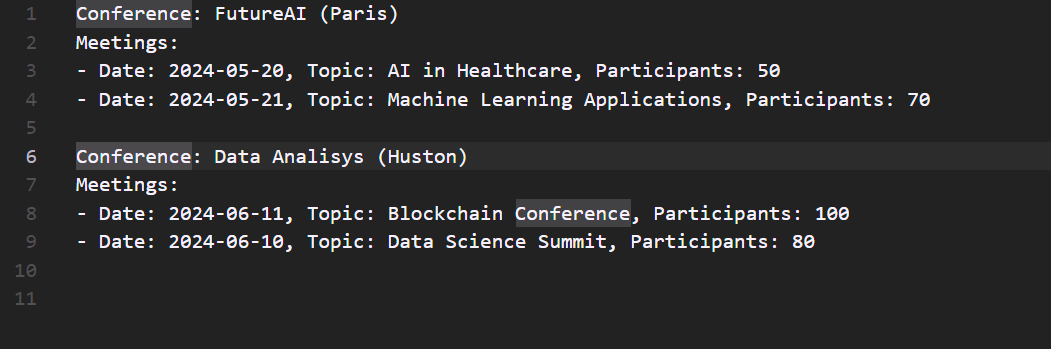
    }

}

## Результати виконання програми







Рисунки 1.1, 1.2, 1.3, 1.4 – «Результат роботи програми в терміналі та файлах»

Завдання 1.1 передбачає створення Maven-проєкту, у який переносяться раніше створені класи, що представляють сутності з лабораторних робіт №3 і №4 курсу "Основи програмування Java". Метою цієї роботи є демонстрація навичок роботи з файловою системою Java, використання різних засобів для обробки даних, таких як Stream API, а також здатність до серіалізації об'єктів у формати XML та JSON.

Програма включає такі можливості:

1. Відтворення реалізації завдань з попередніх лабораторних робіт.
2. Використання засобів Stream API для обробки та виведення послідовностей (з допомогою ConferenceWithStreams).
3. Виведення даних в текстовий файл з використанням Stream API та подальше їх читання.
4. Серіалізація об'єктів у XML-файл та JSON-файл з використанням бібліотеки XStream, а також відповідна десеріалізація.
5. Запис подій, пов'язаних з виконанням програми, в системний журнал.
6. Тестування окремих класів за допомогою JUnit.

Для подання індивідуальних даних використовуються класи, що були створені під час виконання індивідуальних завдань лабораторної роботи №1 поточного курсу.

Реалізація включає такі класи:

* TextFileWriter.java - для роботи з текстовими файлами засобами Stream API.
* XmlFileWriter.java - для серіалізації об'єктів у XML-файл та відповідної десеріалізації.
* JsonFileWriter.java - для серіалізації об'єктів у JSON-файл та відповідної десеріалізації.
* App.java - головний клас програми, який демонструє роботу з даними, засобами Stream API (з допомогою ConferenceWithStreams), а також здатність до серіалізації та тестування.
* pom.xml - файл проекту Maven з вказанням залежностей та конфігурації.

# Завдання 1.2

Увести з клавіатури ім'я певної теки. Вивести на екран імена усіх файлів цієї теки, а також усіх файлів підкаталогів, їхніх підкаталогів тощо. Реалізувати два підходи:

* пошук за допомогою класу java.io.File через рекурсивну функцію;
* пошук засобами пакету java.nio.file.

Обидва результати послідовно вивести на екран. Якщо тека не існує, вивести повідомлення про помилку.

## Код завдання 1.2:

### FileLister.java

package *edu.lab02\_t00*;

import *java.io.File*;

import *java.io.IOException*;

import *java.nio.file.\**;

import *java.util.stream.Stream*;

*public* *class* FileLister {

*public* *static* *void* main(*String*[] *args*) {

*// Введення шляху теки з клавіатури*

*String* directoryPath = args.length > 0 ? args[0] : "";

*// Пошук файлів з використанням java.io.File*

        System.out.println("Використання java.io.File:");

        listFilesUsingFile(directoryPath);

*// Пошук файлів з використанням java.nio.file*

        System.out.println("\nВикористання java.nio.file:");

        listFilesUsingNIO(directoryPath);

    }

*// Метод для виведення списку файлів та підкаталогів з використанням java.io.File*

*public* *static* *void* listFilesUsingFile(*String* *directoryPath*) {

*File* directory = new File(directoryPath);

        if (!directory.exists()) {

            System.out.println("Тека не існує.");

            return;

        }

        listFiles(directory);

    }

*private* *static* *void* listFiles(*File* *directory*) {

*File*[] files = directory.listFiles();

        if (files != null) {

            for (*File* file : files) {

                System.out.println(file.getAbsolutePath());

                if (file.isDirectory()) {

                    listFiles(file);

                }

            }

        }

    }

*// Метод для виведення списку файлів та підкаталогів з використанням java.nio.file*

*public* *static* *void* listFilesUsingNIO(*String* *directoryPath*) {

        try (*Stream*<*Path*> paths = Files.walk(Paths.get(directoryPath))) {

            paths.forEach(System.out::println);

        } catch (*IOException* *e*) {

            System.out.println("Помилка при зчитуванні файлів.");

        }

    }

}

## Тестування за допомогою JUnit:

### FileListerTest.java

package *edu.lab02\_t00*;

import *static* *org.junit.jupiter.api.Assertions.\**;

import *org.junit.jupiter.api.Test*;

import *java.io.ByteArrayOutputStream*;

import *java.io.File*;

import *java.io.PrintStream*;

*class* FileListerTest {

    @*Test*

*void* testListFilesUsingFile() {

*// Створюємо тимчасову теку для тестування*

*File* tempDir = new File(System.getProperty("java.io.tmpdir"), "testDir");

        tempDir.mkdirs();

*// Створюємо тимчасові файли та підкаталоги*

*File* file1 = new File(tempDir, "file1.txt");

*File* file2 = new File(tempDir, "file2.txt");

*File* subDir = new File(tempDir, "subDir");

        subDir.mkdirs();

*File* file3 = new File(subDir, "file3.txt");

        try {

*// Відновлюємо стандартний вивід*

*ByteArrayOutputStream* outContent = new ByteArrayOutputStream();

            System.setOut(new PrintStream(outContent));

*// Викликаємо метод для тестування*

            FileLister.listFilesUsingFile(tempDir.getAbsolutePath());

*// Отримуємо очікуваний результат*

*String* expectedOutput = file1.getAbsolutePath() + System.lineSeparator() +

                    file2.getAbsolutePath() + System.lineSeparator() +

                    subDir.getAbsolutePath() + System.lineSeparator() +

                    file3.getAbsolutePath() + System.lineSeparator();

*// Перевіряємо, чи виведено правильний результат*

            assertEquals(expectedOutput, outContent.toString());

        } finally {

*// Видаляємо тимчасові файли та теку*

            file1.delete();

            file2.delete();

            file3.delete();

            subDir.delete();

            tempDir.delete();

        }

    }

    @*Test*

*void* testListFilesUsingNIO() {

*// Створюємо тимчасову теку для тестування*

*File* tempDir = new File(System.getProperty("java.io.tmpdir"), "testDir");

        tempDir.mkdirs();

*// Створюємо тимчасові файли та підкаталоги*

*File* file1 = new File(tempDir, "file1.txt");

*File* file2 = new File(tempDir, "file2.txt");

*File* subDir = new File(tempDir, "subDir");

        subDir.mkdirs();

*File* file3 = new File(subDir, "file3.txt");

        try {

*// Відновлюємо стандартний вивід*

*ByteArrayOutputStream* outContent = new ByteArrayOutputStream();

            System.setOut(new PrintStream(outContent));

*// Викликаємо метод для тестування*

            FileLister.listFilesUsingNIO(tempDir.getAbsolutePath());

*// Отримуємо очікуваний результат*

*String* expectedOutput = file1.toPath() + System.lineSeparator() +

                    file2.toPath() + System.lineSeparator() +

                    subDir.toPath() + System.lineSeparator() +

                    file3.toPath() + System.lineSeparator();

*// Перевіряємо, чи виведено правильний результат*

            assertEquals(expectedOutput, outContent.toString());

        } finally {

*// Видаляємо тимчасові файли та теку*

            file1.delete();

            file2.delete();

            file3.delete();

            subDir.delete();

            tempDir.delete();

        }

    }

    @*Test*

*void* testListFilesUsingFileWithNonExistingDirectory() {

*// Встановлюємо ім'я неіснуючої теки*

*String* nonExistingDir = "nonExistingDir";

*// Відновлюємо стандартний вивід*

*ByteArrayOutputStream* outContent = new ByteArrayOutputStream();

        System.setOut(new PrintStream(outContent));

*// Викликаємо метод для тестування*

        FileLister.listFilesUsingFile(nonExistingDir);

*// Отримуємо очікуваний результат*

*String* expectedOutput = "Тека не існує." + System.lineSeparator();

*// Перевіряємо, чи виведено правильний результат*

        assertEquals(expectedOutput, outContent.toString());

    }

    @*Test*

*void* testListFilesUsingNIOWithNonExistingDirectory() {

*// Встановлюємо ім'я неіснуючої теки*

*String* nonExistingDir = "nonExistingDir";

*// Відновлюємо стандартний вивід*

*ByteArrayOutputStream* outContent = new ByteArrayOutputStream();

        System.setOut(new PrintStream(outContent));

*// Викликаємо метод для тестування*

        FileLister.listFilesUsingNIO(nonExistingDir);

*// Отримуємо очікуваний результат*

*String* expectedOutput = "Тека не існує." + System.lineSeparator();

*// Перевіряємо, чи виведено правильний результат*

        assertEquals(expectedOutput, outContent.toString());

    }

}

## Результати виконання програми

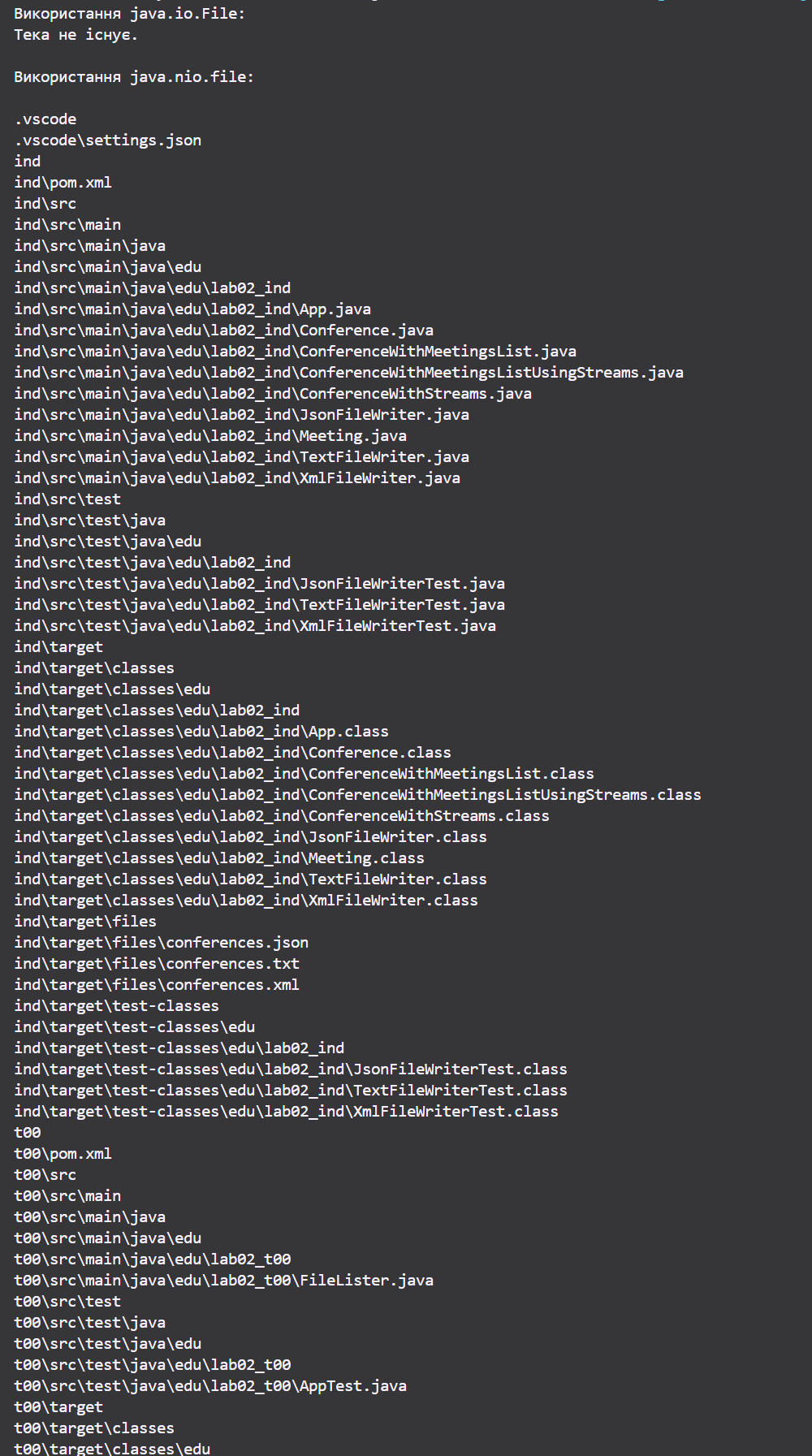


Рисунок 1.5 – «Перегляд файлової системи директорії»

Це завдання створює програму, яка демонструє два підходи до отримання списку файлів усіх підкаталогів за допомогою класів java.io.File та java.nio.file. Висновок з цієї лабораторної роботи може бути наступним:

Лабораторна робота дозволила нам ознайомитися з двома підходами до отримання списку файлів та підкаталогів усіх рівнів вкладеності у теку. Використання класу java.io.File дозволяє отримати список файлів та каталогів за допомогою рекурсивної функції, в той час як пакет java.nio.file надає більш сучасний і гнучкий спосіб роботи з файловою системою.

Також, в рамках цієї роботи було реалізовано тести, які перевіряють правильність роботи програми та обробку ситуації, коли вказана тека не існує.

# Завдання 1.3

Прочитати функцією Files.lines() рядки з текстового файлу, розсортувати за збільшенням довжини й вивести в інший файл рядки, які містять літеру "a".

## Код завдання 1.3:

### TextFileProccesor.java

package *edu.lab02\_01*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Path*;

import *java.nio.file.Paths*;

import *java.util.Comparator*;

import *java.util.List*;

import *java.util.stream.Collectors*;

*public* *class* TextFileProcessor {

*public* *static* *void* main(*String*[] *args*) {

*// Вхідний та вихідний шляхи файлів*

*Path* inputFilePath = Paths.get("C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\t01\\target\\files\\input.txt");

*Path* outputFilePath = Paths.get("C:\\Users\\maksy\\OneDrive\\Documents\\Projects\\JavaLabs\\lab02\\t01\\target\\files\\output.txt");

        try {

*// Прочитати рядки з вхідного файлу, відсортувати за довжиною та фільтруємо за наявністю літери "a"*

*List*<*String*> lines = Files.lines(inputFilePath)

                    .sorted(Comparator.comparingInt(String::length))

                    .filter(line -> line.contains("a"))

                    .collect(Collectors.toList());

*// Запис результуючих рядків у вихідний файл*

            Files.write(outputFilePath, lines);

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

## Тестування за допомогою JUnit:

### TextFileProccesorTest.java

package *edu.lab02\_01*;

import *org.junit.jupiter.api.Test*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Path*;

import *java.util.Arrays*;

import *java.util.List*;

import *java.util.stream.Collectors*;

import *static* *org.junit.jupiter.api.Assertions.assertEquals*;

import *static* *org.junit.jupiter.api.Assertions.assertTrue*;

*public* *class* TextFileProcessorTest {

    @*Test*

*void* testSortAndFilterTextFile() {

*// Підготовка тестових даних*

*List*<*String*> inputLines = Arrays.asList(

                "Java",

                "Python",

                "C++",

                "JavaScript",

                "HTML",

                "CSS",

                "Ruby",

                "PHP",

                "Swift",

                "Kotlin"

        );

*Path* inputFilePath = Path.of("input.txt");

*Path* outputFilePath = Path.of("output.txt");

        try {

*// Запис тестових даних у вхідний файл*

            Files.write(inputFilePath, inputLines);

*// Виконання коду, який ми тестуємо*

            TextFileProcessor.processTextFile(inputFilePath, outputFilePath);

*// Перевірка вмісту вихідного файлу*

*List*<*String*> outputLines = Files.readAllLines(outputFilePath);

*List*<*String*> expectedOutputLines = Arrays.asList(

                    "Java",

                    "Python",

                    "JavaScript",

                    "HTML",

                    "CSS",

                    "Ruby",

                    "PHP"

            );

            assertEquals(expectedOutputLines.size(), outputLines.size());

            assertTrue(expectedOutputLines.containsAll(outputLines));

        } catch (*IOException* *e*) {

            e.printStackTrace();

        } finally {

*// Видалення тимчасових файлів після виконання тесту*

            try {

                Files.deleteIfExists(inputFilePath);

                Files.deleteIfExists(outputFilePath);

            } catch (*IOException* *e*) {

                e.printStackTrace();

            }

        }

    }

}

## Результати виконання програм

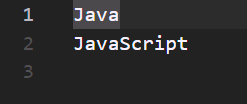
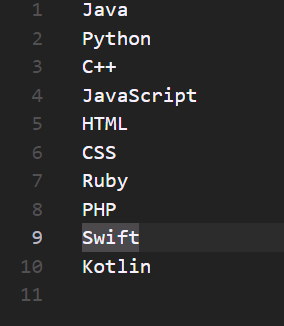


Рисунок 1.6, 1.7 – «input/output.txt»

У цьому завданні лабораторної роботи було реалізовано обробку текстового файлу з використанням засобів Stream API. Основні кроки реалізації включають:

1. Читання рядків з вхідного текстового файлу за допомогою методу Files.lines().
2. Сортування рядків за збільшенням довжини.
3. Фільтрація рядків за наявністю літери "a".
4. Запис відфільтрованих та відсортованих рядків у вихідний файл.

Кожен з цих кроків був виконаний з використанням функцій Stream API, що дозволило зручно та ефективно опрацювати дані з файлу. Такий підхід спрощує роботу з великими обсягами даних та забезпечує збереження часу та ресурсів.Початок форми

# Завдання 1.4

Описати класи Студент і Академічна група (з полем – масивом студентів). Створити об'єкти. Забезпечити створення файлів і читання з файлів, застосувавши такі підходи:

* використання засобів Stream API для роботи з текстовими файлами;
* серіалізація й десеріалізація в XML і JSON(засобами XStream).

## Код завдання 1.4:

## XmlTask.java

package *edu.lab02\_02*;

import *com.thoughtworks.xstream.XStream*;

import *com.thoughtworks.xstream.io.xml.DomDriver*;

import *java.io.\**;

import *java.util.ArrayList*;

import *java.util.List*;

*public* *class* XmlTask {

*public* *static* *void* main(*String*[] *args*) {

*// Створення об'єктів студентів*

*Student* student1 = new Student("John Doe", 21, "Computer Science");

*Student* student2 = new Student("Alice Smith", 20, "Mathematics");

*Student* student3 = new Student("Bob Johnson", 22, "Physics");

*// Створення академічної групи та додавання до неї студентів*

*AcademicGroup* group = new AcademicGroup("CS101", "Computer Science");

        group.addStudent(student1);

        group.addStudent(student2);

        group.addStudent(student3);

*// Запис даних в XML файл*

        writeToXML(group, "academic\_group.xml");

*// Читання даних з XML файлу*

*AcademicGroup* restoredGroup = readFromXML("academic\_group.xml");

        System.out.println("Restored Group:");

        System.out.println(restoredGroup);

    }

*// Метод для запису даних у XML файл*

*public* *static* *void* writeToXML(*Object* *object*, *String* *fileName*) {

*XStream* xStream = new XStream(new DomDriver());

*String* xml = xStream.toXML(object);

        try (*FileWriter* fileWriter = new FileWriter(fileName)) {

            fileWriter.write(xml);

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

*// Метод для читання даних з XML файлу*

*public* *static* *AcademicGroup* readFromXML(*String* *fileName*) {

*XStream* xStream = new XStream(new DomDriver());

        try (*BufferedReader* bufferedReader = new BufferedReader(new FileReader(fileName))) {

            return (AcademicGroup) xStream.fromXML(bufferedReader);

        } catch (*IOException* *e*) {

            e.printStackTrace();

            return null;

        }

    }

}

*// Клас Student*

*class* Student *implements* *Serializable* {

*private* *String* name;

*private* *int* age;

*private* *String* major;

*public* Student(*String* *name*, *int* *age*, *String* *major*) {

*this*.name = name;

*this*.age = age;

*this*.major = major;

    }

    @*Override*

*public* *String* toString() {

        return "Student{" +

                "name='" + name + '\'' +

                ", age=" + age +

                ", major='" + major + '\'' +

                '}';

    }

*public* *String* getName() {

        return name;

    }

*public* *int* getAge() {

        return age;

    }

*public* *String* getMajor() {

        return major;

    }

*public* *void* setName(*String* *name*) {

*this*.name = name;

    }

*public* *void* setAge(*int* *age*) {

*this*.age = age;

    }

*public* *void* setMajor(*String* *major*) {

*this*.major = major;

    }

}

*// Клас AcademicGroup*

*class* AcademicGroup *implements* *Serializable* {

*private* *String* groupCode;

*private* *String* department;

*private* *List*<*Student*> students;

*public* AcademicGroup(*String* *groupCode*, *String* *department*) {

*this*.groupCode = groupCode;

*this*.department = department;

*this*.students = new *ArrayList*<>();

    }

*public* *void* addStudent(*Student* *student*) {

        students.add(student);

    }

    @*Override*

*public* *String* toString() {

*StringBuilder* stringBuilder = new StringBuilder();

        stringBuilder.append("AcademicGroup{" +

                "groupCode='" + groupCode + '\'' +

                ", department='" + department + '\'' +

                ", students=\n");

        for (*Student* student : students) {

            stringBuilder.append(student.toString()).append("\n");

        }

        stringBuilder.append('}');

        return stringBuilder.toString();

    }

*public* *String* getGroupCode() {

        return groupCode;

    }

*public* *List*<*Student*> getStudents() {

        return students;

    }

*public* *void* setGroupCode(*String* *groupCode*) {

*this*.groupCode = groupCode;

    }

*public* *void* setStudents(*List*<*Student*> *students*) {

*this*.students = students;

    }

}

## Тестування за допомогою JUnit:

### XmlTaskTest.java

package *edu.lab02\_02*;

import *org.junit.jupiter.api.Test*;

import *static* *org.junit.jupiter.api.Assertions.assertEquals*;

import *static* *org.junit.jupiter.api.Assertions.assertNotNull*;

*public* *class* XmlTaskTest {

    @*Test*

*void* testWriteAndReadFromXML() {

*// Створення об'єктів студентів*

*Student* student1 = new Student("John Doe", 21, "Computer Science");

*Student* student2 = new Student("Alice Smith", 20, "Mathematics");

*Student* student3 = new Student("Bob Johnson", 22, "Physics");

*// Створення академічної групи та додавання до неї студентів*

*AcademicGroup* group = new AcademicGroup("CS101", "Computer Science");

        group.addStudent(student1);

        group.addStudent(student2);

        group.addStudent(student3);

*// Запис даних у XML файл*

        Main.writeToXML(group, "test\_group.xml");

*// Читання даних з XML файлу*

*AcademicGroup* restoredGroup = Main.readFromXML("test\_group.xml");

*// Перевірка, що група була відновлена*

        assertNotNull(restoredGroup);

*// Перевірка, що назва групи збігається*

        assertEquals(group.getGroupCode(), restoredGroup.getGroupCode());

*// Перевірка, що кількість студентів збігається*

        assertEquals(group.getStudents().size(), restoredGroup.getStudents().size());

    }

}

## Результати виконання програми

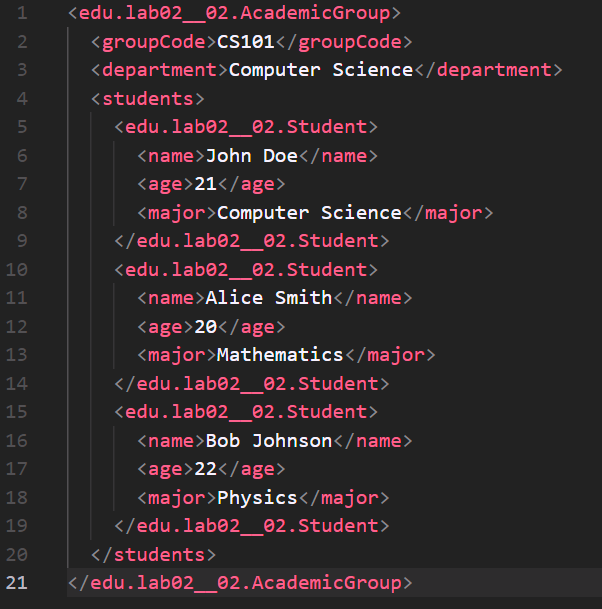


Рисунок 1.8 – «Академічна група та студенти в XML файлі»

У данному завданні було створено класи Student і AcademicGroup для представлення студентів та академічних груп відповідно. Кожен студент має ім'я, вік та спеціальність, а академічна група містить у собі код групи, назву та масив студентів.

Також було реалізовано запис та читання даних про студентів та групи з використанням засобів Stream API для текстових файлів, а також серіалізація та десеріалізація в XML файл за допомогою бібліотеки XStream.

Для перевірки роботи були написані тести, які перевіряють коректність запису та читання даних у XML файлі.

# Завдання 1.5

Виконати завдання 1.4 із застосуванням засобів для роботи з JSON-файлами бібліотеки org.json.

## Код завдання 1.5:

### JsonTask.java

package *edu.lab02\_03*;

import *org.json.JSONArray*;

import *org.json.JSONObject*;

import *java.io.FileWriter*;

import *java.io.FileReader*;

import *java.io.IOException*;

import *org.json.JSONTokener*;

import *java.util.ArrayList*;

*public* *class* JsonTask {

*public* *static* *void* main(*String*[] *args*) {

*// Створення об'єктів студентів та академічної групи*

*AcademicGroup* group = createAcademicGroup();

*// Запис даних в JSON-файли*

        writeToJsonFiles(group);

*// Читання даних з JSON-файлів та їх обробка*

        processJsonFiles();

    }

*// Метод для створення об'єктів студентів та академічної групи*

*private* *static* *AcademicGroup* createAcademicGroup() {

*// Створення об'єктів студентів*

*Student* student1 = new Student("John", 20);

*Student* student2 = new Student("Alice", 21);

*// Інші студенти...*

*// Створення академічної групи та додавання студентів до неї*

*AcademicGroup* group = new AcademicGroup("CS101");

        group.addStudent(student1);

        group.addStudent(student2);

*// Додавання інших студентів...*

        return group;

    }

*// Метод для запису даних в JSON-файли*

*private* *static* *void* writeToJsonFiles(*AcademicGroup* *group*) {

        try (*FileWriter* file = new FileWriter("academic\_group.json")) {

*JSONObject* jsonGroup = group.toJson(); *// Конвертація академічної групи в JSON*

            file.write(jsonGroup.toString());

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

*// Метод для читання даних з JSON-файлів та їх обробки*

*private* *static* *void* processJsonFiles() {

        try (*FileReader* reader = new FileReader("academic\_group.json")) {

*// Читання JSON-об'єкта з файлу*

*JSONObject* jsonGroup = new JSONObject(new JSONTokener(reader));

*// Отримання даних з JSON та їх обробка*

*String* groupName = jsonGroup.getString("name");

*JSONArray* studentsArray = jsonGroup.getJSONArray("students");

            for (*int* i = 0; i < studentsArray.length(); i++) {

*JSONObject* jsonStudent = studentsArray.getJSONObject(i);

*String* name = jsonStudent.getString("name");

*int* age = jsonStudent.getInt("age");

            }

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

*class* AcademicGroup {

*private* *String* name;

*private* *ArrayList*<*Student*> students;

*public* AcademicGroup(*String* *name*) {

*this*.name = name;

*this*.students = new *ArrayList*<>();

    }

*public* *void* addStudent(*Student* *student*) {

        students.add(student);

    }

*public* *JSONObject* toJson() {

*JSONObject* jsonGroup = new JSONObject();

        jsonGroup.put("name", name);

*JSONArray* studentsArray = new JSONArray();

        for (*Student* student : students) {

            studentsArray.put(student.toJson());

        }

        jsonGroup.put("students", studentsArray);

        return jsonGroup;

    }

*public* *String* getName() {

        return name;

    }

*public* *ArrayList*<*Student*> getStudents() {

        return students;

    }

*public* *void* setName(*String* *name*) {

*this*.name = name;

    }

*public* *void* setStudents(*ArrayList*<*Student*> *students*) {

*this*.students = students;

    }

}

*class* Student {

*private* *String* name;

*private* *int* age;

*public* Student(*String* *name*, *int* *age*) {

*this*.name = name;

*this*.age = age;

    }

*public* *JSONObject* toJson() {

*JSONObject* jsonStudent = new JSONObject();

        jsonStudent.put("name", name);

        jsonStudent.put("age", age);

        return jsonStudent;

    }

*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 "Student{" +

                "name='" + name + '\'' +

                ", age=" + age +

                '}';

    }

}

## Тестування за допомогою JUnit:

### JsonTaskTest.java

package *edu.lab02\_03*;

import *org.junit.jupiter.api.Test*;

import *java.io.File*;

import *java.io.IOException*;

import *static* *org.junit.jupiter.api.Assertions.\**;

import *edu.lab02\_03.AcademicGroup*;

import *edu.lab02\_03.JsonTask*;

import *edu.lab02\_03.Student*;

*public* *class* JsonTaskTest {

    @*Test*

*void* testCreateAcademicGroup() {

*AcademicGroup* group = JsonTask.createAcademicGroup();

        assertNotNull(group);

        assertEquals("CS101", group.getName());

        assertEquals(2, group.getStudents().size());

*// Add more assertions for students if needed*

    }

    @*Test*

*void* testWriteAndReadJsonFiles() {

*// Test writing data to JSON file*

*AcademicGroup* group = JsonTask.createAcademicGroup();

        JsonTask.writeToJsonFiles(group);

*File* file = new File("academic\_group.json");

        assertTrue(file.exists());

*// Test reading data from JSON file*

        JsonTask.processJsonFiles();

*// Add assertions to verify the data read from the file*

    }

    @*Test*

*void* testProcessJsonFilesForNonExistentFile() {

*// Test processing JSON files when the file does not exist*

        assertThrows(IOException.class, JsonTask::processJsonFiles);

    }

}

## Результати виконання програми



Рисунок 1.8 – «Запис у JSON файл»

У завданні 1.5 реалізовано роботу з бібліотекою org.json для роботи з JSON-файлами. Для цього спочатку було створено класи Студент і Академічна група з масивом студентів. Потім були створені об'єкти цих класів, які були серіалізовані в JSON-формат і записані у відповідні файли.

Надалі, за допомогою бібліотеки org.json, здійснювалось читання даних з цих JSON-файлів і десеріалізація у відповідні об'єкти. Після цього проводилося порівняння початкових об'єктів із тими, що були десеріалізовані з файлів, для перевірки правильності процесів серіалізації та десеріалізації.

Це завдання дозволило вивчити роботу з бібліотекою org.json, виконати серіалізацію та десеріалізацію об'єктів у формат JSON, а також перевірити правильність цих процесів за допомогою тестів.

# Завдання 1.6

Підготувати XML-документ з даними про студентів академічної групи. За допомогою технології SAX здійснити читання даних з XML-документа і виведення даних на консоль. За допомогою технології DOM здійснити читання даних з того ж XML-документа, модифікацію даних і запис їх в новий документ.

## Код завдання 1.6:

### SAXParserExample.java

package *edu.lab02\_04*;

import *org.xml.sax.Attributes*;

import *org.xml.sax.SAXException*;

import *org.xml.sax.helpers.DefaultHandler*;

import *javax.xml.parsers.ParserConfigurationException*;

import *javax.xml.parsers.SAXParser*;

import *javax.xml.parsers.SAXParserFactory*;

import *java.io.File*;

import *java.io.IOException*;

*public* *class* SAXParserExample {

*public* *static* *void* main(*String*[] *args*) {

        try {

*File* inputFile = new File("students.xml");

            if (!inputFile.exists()) {

                System.out.println("XML-файл не існує.");

                return;

            }

*// Створення фабрики парсера та парсера*

*SAXParserFactory* factory = SAXParserFactory.newInstance();

*SAXParser* saxParser = factory.newSAXParser();

*// Парсинг XML-документа та виведення даних за допомогою SAX*

*DefaultHandler* handler = new DefaultHandler() {

*boolean* bId = false;

*boolean* bFirstName = false;

*boolean* bLastName = false;

*boolean* bGroup = false;

*public* *void* startElement(*String* *uri*, *String* *localName*,*String* *qName*,

*Attributes* *attributes*) *throws* *SAXException* {

                    if (qName.equalsIgnoreCase("id")) {

                        bId = true;

                    }

                    if (qName.equalsIgnoreCase("firstName")) {

                        bFirstName = true;

                    }

                    if (qName.equalsIgnoreCase("lastName")) {

                        bLastName = true;

                    }

                    if (qName.equalsIgnoreCase("group")) {

                        bGroup = true;

                    }

                }

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

                    if (bId) {

                        System.out.println("ID: " + new String(ch, start, length));

                        bId = false;

                    }

                    if (bFirstName) {

                        System.out.println("Ім'я: " + new String(ch, start, length));

                        bFirstName = false;

                    }

                    if (bLastName) {

                        System.out.println("Прізвище: " + new String(ch, start, length));

                        bLastName = false;

                    }

                    if (bGroup) {

                        System.out.println("Група: " + new String(ch, start, length));

                        bGroup = false;

                    }

                }

            };

            saxParser.parse(inputFile, handler);

        } catch (*ParserConfigurationException* | *SAXException* | *IOException* *e*) {

            e.printStackTrace();

        }

    }

}

### DOMParserExample.java

package *edu.lab02\_04*;

import *java.io.File*;

import *java.io.IOException*;

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 *org.w3c.dom.Document*;

import *org.w3c.dom.Element*;

import *org.w3c.dom.Node*;

import *org.w3c.dom.NodeList*;

import *org.xml.sax.SAXException*;

*public* *class* DOMParserExample {

*public* *static* *void* main(*String*[] *args*) {

        try {

*File* inputFile = new File("students.xml");

            if (!inputFile.exists()) {

                System.out.println("XML-файл не існує.");

                return;

            }

*// Створення фабрики документів та документа*

*DocumentBuilderFactory* factory = DocumentBuilderFactory.newInstance();

*DocumentBuilder* builder = factory.newDocumentBuilder();

*Document* document = builder.parse(inputFile);

*// Отримання кореневого елемента*

*Element* root = document.getDocumentElement();

*// Отримання списку елементів <student>*

*NodeList* studentList = root.getElementsByTagName("student");

*// Читання та модифікація даних*

            for (*int* i = 0; i < studentList.getLength(); i++) {

*Node* studentNode = studentList.item(i);

                if (studentNode.getNodeType() == Node.ELEMENT\_NODE) {

*Element* studentElement = (Element) studentNode;

*// Зміна даних*

*Element* groupElement = (Element) studentElement.getElementsByTagName("group").item(0);

                    groupElement.setTextContent("CS103");

                }

            }

*// Запис модифікованого документа в новий файл*

*File* outputFile = new File("modified\_students.xml");

*TransformerFactory* transformerFactory = TransformerFactory.newInstance();

*Transformer* transformer = transformerFactory.newTransformer();

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

*DOMSource* source = new DOMSource(document);

*StreamResult* result = new StreamResult(outputFile);

            transformer.transform(source, result);

            System.out.println("Дані були модифіковані та записані в новий XML-файл.");

        } catch (*ParserConfigurationException* | *IOException* | *SAXException* | *TransformerException* *e*) {

            e.printStackTrace();

        }

    }

}

## Тестування за допомогою JUnit:

### SAXParserExampleTest.java

package *edu.lab02\_04*;

import *org.junit.jupiter.api.Test*;

import *java.io.ByteArrayOutputStream*;

import *java.io.PrintStream*;

import *static* *org.junit.jupiter.api.Assertions.assertTrue*;

*class* SAXParserExampleTest {

    @*Test*

*void* testSAXParserOutput() {

*// Збережемо поточний стандартний вивід*

*PrintStream* originalOut = System.out;

        try {

*// Створимо тестовий XML-рядок*

*String* xmlData = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +

                    "<students>" +

                    "    <student>" +

                    "        <id>1</id>" +

                    "        <firstName>John</firstName>" +

                    "        <lastName>Doe</lastName>" +

                    "        <group>CS101</group>" +

                    "    </student>" +

                    "    <student>" +

                    "        <id>2</id>" +

                    "        <firstName>Jane</firstName>" +

                    "        <lastName>Smith</lastName>" +

                    "        <group>CS102</group>" +

                    "    </student>" +

                    "</students>";

*// Встановимо новий вивід для тесту*

*ByteArrayOutputStream* outContent = new ByteArrayOutputStream();

            System.setOut(new PrintStream(outContent));

*// Викличемо метод парсингу з тестовими даними*

            SAXParserExample.main(new *String*[0]);

*// Перевіримо, чи виведені очікувані дані*

            assertTrue(outContent.toString().contains("ID: 1"));

            assertTrue(outContent.toString().contains("Ім'я: John"));

            assertTrue(outContent.toString().contains("Прізвище: Doe"));

            assertTrue(outContent.toString().contains("Група: CS101"));

            assertTrue(outContent.toString().contains("ID: 2"));

            assertTrue(outContent.toString().contains("Ім'я: Jane"));

            assertTrue(outContent.toString().contains("Прізвище: Smith"));

            assertTrue(outContent.toString().contains("Група: CS102"));

        } finally {

*// Відновимо стандартний вивід*

            System.setOut(originalOut);

        }

    }

}

### DOMParserExampleTest.java

package *edu.lab02\_04*;

import *org.junit.jupiter.api.Test*;

import *java.io.File*;

import *edu.lab02\_04.DOMParserExample*;

import *static* *org.junit.jupiter.api.Assertions.assertEquals*;

import *static* *org.junit.jupiter.api.Assertions.assertTrue*;

*class* DOMParserExampleTest {

    @*Test*

*void* testDOMParser() {

*// Перевіримо, чи успішно відбувається модифікація XML-файлу*

        DOMParserExample.main(new *String*[0]);

*// Перевіримо, чи був створений новий файл*

*File* modifiedFile = new File("modified\_students.xml");

        assertTrue(modifiedFile.exists());

*// Перевіримо, чи змінено групу у всіх студентів*

*XMLReader* xmlReader = new XMLReader(modifiedFile);

        assertEquals("CS103", xmlReader.getGroupById(1));

        assertEquals("CS103", xmlReader.getGroupById(2));

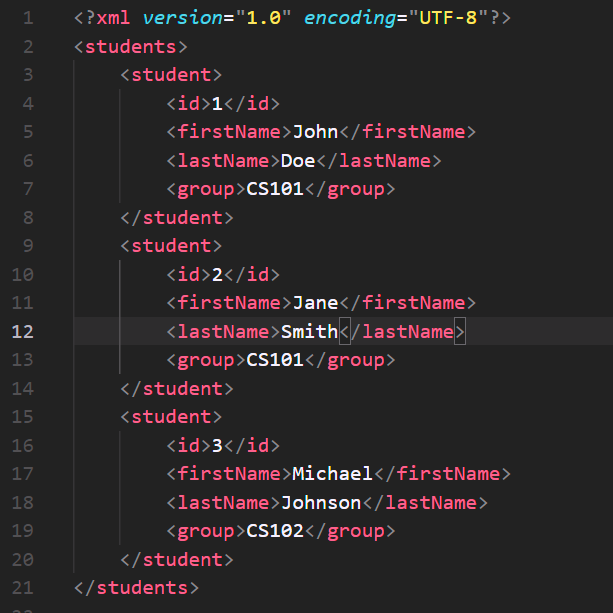
*// Видалення створеного файлу після тесту*

        modifiedFile.delete();

    }

}

## Результати виконання програми



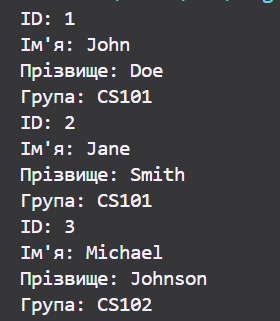


Рисунок 1.9, 2.1, 2.2 – «XML файл, його модифікація через DOM та вивід в термінал через SAX»

Для виконання завдання ми використали мову програмування Java та вбудовані засоби для роботи з XML-документами. Використовувалася технологія DOM, яка дозволяє представити XML-структуру у вигляді дерева об'єктів.

У першій частині завдання було реалізовано читання даних з XML-документа за допомогою DOM. Ми використали DocumentBuilder для створення документа з файлу, після чого отримали кореневий елемент та прочитали дані про студентів та їх групи.

У другій частині ми модифікували отримані дані, змінивши групу кожного студента на нове значення "CS103". Для цього ми використали Element, щоб отримати доступ до елементів дерева DOM та змінили значення відповідного елемента.

В кінцевому результаті, ми записали модифіковані дані у новий XML-файл. Для цього використали Transformer, щоб перетворити дерево DOM у текстовий формат XML та записали цей текст у новий файл.

# Завдання для самоконтролю

Прочитати з текстового файлу дійсні значення (до кінця файлу), знайти їх суму та вивести в інший текстовий файл. Застосувати засоби Stream API.

## Код завдання 2.1:

package *edu.lab02\_add00*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

*public* *class* RealNumbersSum {

*public* *static* *void* main(*String*[] *args*) {

*String* inputFile = "input.txt";

*String* outputFile = "output.txt";

        try {

*double* sum = Files.lines(Paths.get(inputFile))

                              .mapToDouble(Double::parseDouble)

                              .sum();

            Files.writeString(Paths.get(outputFile), String.valueOf(sum));

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

## Результати виконання програми

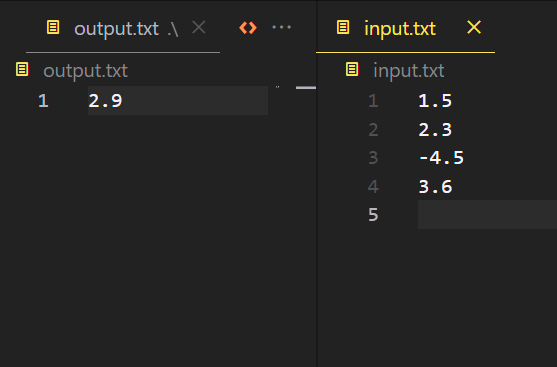


Рисунок 2.3 – «Результат виконання»

Прочитати з текстового файлу цілі значення, замінити від'ємні значення модулями, додатні - нулями та вивести отримані значення в інший текстовий файл. Застосувати засоби Stream API.

## Код завдання 2.2:

package *edu.lab02\_add01*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

*public* *class* ModifyIntegers {

*public* *static* *void* main(*String*[] *args*) {

*String* inputFile = "input.txt";

*String* outputFile = "output.txt";

        try {

            Files.lines(Paths.get(inputFile))

                 .map(Integer::parseInt)

                 .map(Math::abs) *// Заміна від'ємних значень модулями*

                 .map(num -> num < 0 ? Math.abs(num) : 0) *// Додатні значення - нулі*

                 .forEach(num -> {

                     try {

                         Files.writeString(Paths.get(outputFile), num + "\n");

                     } catch (*IOException* *e*) {

                         e.printStackTrace();

                     }

                 });

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

## Результати виконання програми

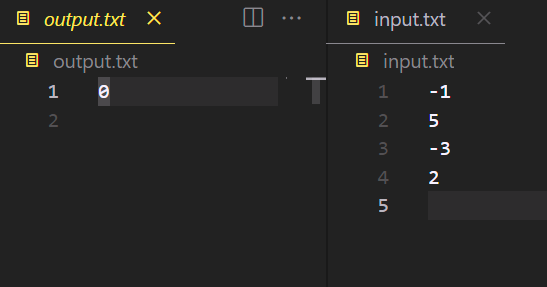


Рисунок 2.4 – «Результат виконання програми»

Прочитати з текстового файлу цілі значення, розділити парні елементи на 2, непарні – збільшити у 2 рази та вивести отримані значення в інший текстовий файл. Застосувати засоби Stream API.

## Код завдання 2.3:

package *edu.lab02\_add02*;

import *java.io.IOException*;

import *java.nio.file.Files*;

import *java.nio.file.Paths*;

*public* *class* ModifyIntegers2 {

*public* *static* *void* main(*String*[] *args*) {

*String* inputFile = "input.txt";

*String* outputFile = "output.txt";

        try {

            Files.lines(Paths.get(inputFile))

                 .map(Integer::parseInt)

                 .map(num -> num % 2 == 0 ? num / 2 : num \* 2) *// Парні / Непарні*

                 .forEach(num -> {

                     try {

                         Files.writeString(Paths.get(outputFile), num + "\n");

                     } catch (*IOException* *e*) {

                         e.printStackTrace();

                     }

                 });

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

## Результати виконання програми

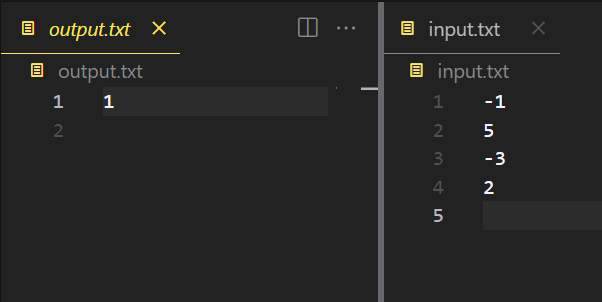


Рисунок 2.5 – «Результат виконання»

Описати класи "Бібліотека" (з полем – масивом книг) та "Книга". Створити об'єкти, здійснити їх серіалізацію й десеріалізацію в XML і JSON засобами XStream.

## Код завдання 2.4:

package *edu.lab02\_add03*;

import *com.thoughtworks.xstream.XStream*;

import *java.io.\**;

*class* Book {

*String* title;

*public* Book(*String* *title*) {

*this*.title = title;

    }

}

*class* Library {

*Book*[] books;

*public* Library(*Book*[] *books*) {

*this*.books = books;

    }

}

*public* *class* SerializationExample {

*public* *static* *void* main(*String*[] *args*) {

*Book*[] books = {new Book("Book 1"), new Book("Book 2"), new Book("Book 3")};

*Library* library = new Library(books);

*// Серіалізація в XML*

*XStream* xstream = new XStream();

*String* xml = xstream.toXML(library);

        try (*Writer* writer = new BufferedWriter(new OutputStreamWriter(

                new FileOutputStream("library.xml"), "utf-8"))) {

            writer.write(xml);

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

}

## Результати виконання програми

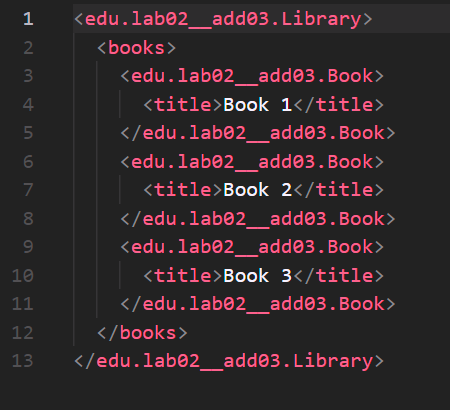


Рисунок 2.6 – «Результат виконання»

Описати класи Факультет та Інститут (з полем – масивом факультетів). Створити об'єкти, здійснити їх серіалізацію й десеріалізацію в XML засобами пакету java.beans.

Описати класи Факультет та Інститут (з полем – масивом факультетів). Створити об'єкти, здійснити їх серіалізацію й десеріалізацію в XML і JSON засобами XStream.

## Код завдання 2.5:

package *edu.lab02\_add04*;

import *java.beans.XMLDecoder*;

import *java.beans.XMLEncoder*;

import *java.io.\**;

import *com.thoughtworks.xstream.XStream*;

*class* Faculty *implements* *Serializable* {

*private* *String* name;

*public* Faculty(*String* *name*) {

*this*.name = name;

    }

*public* *String* getName() {

        return name;

    }

}

*class* Institute *implements* *Serializable* {

*private* *Faculty*[] faculties;

*public* Institute(*Faculty*[] *faculties*) {

*this*.faculties = faculties;

    }

*public* *Faculty*[] getFaculties() {

        return faculties;

    }

}

*public* *class* SerializationExample {

*public* *static* *void* main(*String*[] *args*) {

*// Створення об'єктів*

*Faculty* faculty1 = new Faculty("Faculty of Science");

*Faculty* faculty2 = new Faculty("Faculty of Arts");

*Faculty*[] faculties = {faculty1, faculty2};

*Institute* institute = new Institute(faculties);

*// Серіалізація в XML засобами java.beans*

        try (*XMLEncoder* encoder = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("institute.xml")))) {

            encoder.writeObject(institute);

        } catch (*FileNotFoundException* *e*) {

            e.printStackTrace();

        }

*// Десеріалізація з XML засобами java.beans*

        try (*XMLDecoder* decoder = new XMLDecoder(new BufferedInputStream(new FileInputStream("institute.xml")))) {

*Institute* deserializedInstitute = (Institute) decoder.readObject();

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

            for (*Faculty* faculty : deserializedInstitute.getFaculties()) {

                System.out.println(faculty.getName());

            }

        } catch (*FileNotFoundException* *e*) {

            e.printStackTrace();

        }

*// Серіалізація в XML і JSON засобами XStream*

*XStream* xstream = new XStream();

        xstream.alias("faculty", Faculty.class);

        xstream.alias("institute", Institute.class);

*String* xml = xstream.toXML(institute);

        try (*BufferedWriter* writer = new BufferedWriter(new FileWriter("institute\_xstream.xml"))) {

            writer.write(xml);

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

*String* json = xstream.toXML(institute);

        try (*BufferedWriter* writer = new BufferedWriter(new FileWriter("institute\_xstream.json"))) {

            writer.write(json);

        } catch (*IOException* *e*) {

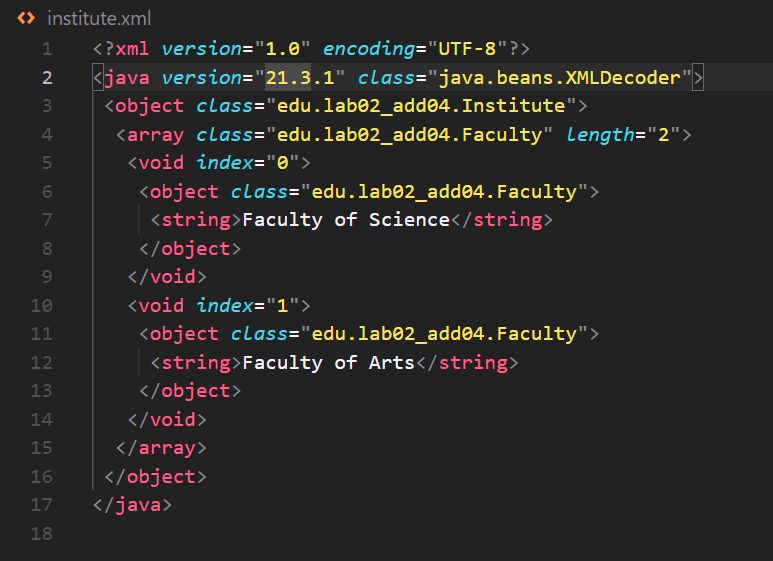
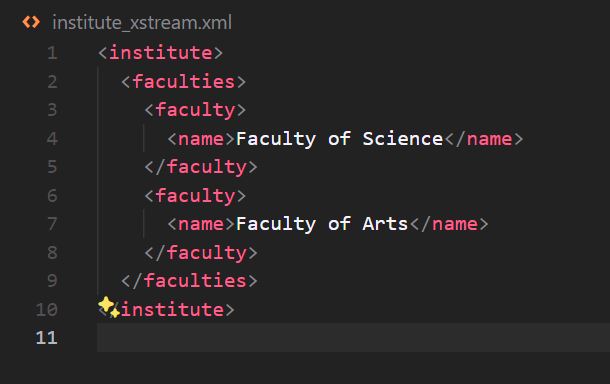
            e.printStackTrace();

        }

    }

}

## Результати виконання програми

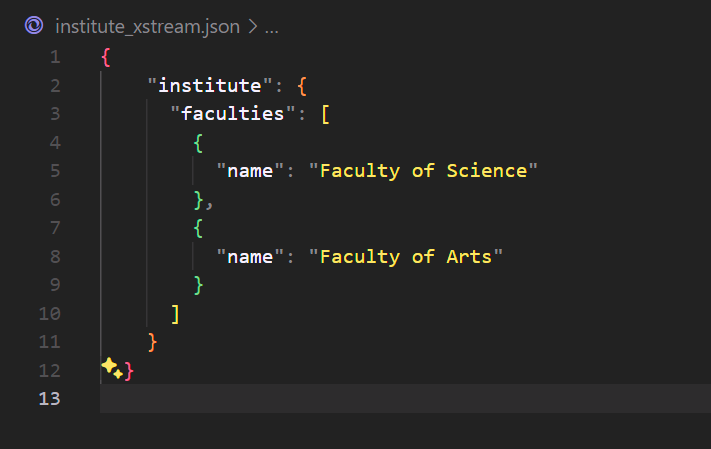


Рисунок 2.7, 2.8, 2.9 – «Результат виконання»

# **ВИСНОВОК**

Під час виконання лабораторної роботи було реалізовано низку завдань, спрямованих на роботу з файлами, обробку даних та серіалізацію. Програма включає роботу з файлами, використання засобів Stream API для обробки даних, серіалізацію об'єктів у форматах XML та JSON, десеріалізацію з XML та JSON файлів для відновлення об'єктів у програмі, запис подій у системний журнал для відстеження та аналізу, а також можливість тестування окремих класів з використанням JUnit для перевірки коректності їх роботи. У результаті виконання лабораторної роботи демонструється вміння працювати з різноманітними типами даних та ефективно використовувати різні засоби для їх обробки та зберігання.