**Міністерство освіти і науки України**

**Національний технічний університет**

**«Дніпровська політехніка»**

****

**ЗВІТ**

**про виконання лабораторних робіт**

**з дисципліни**

**«Поглиблене програмування в середовищі Java»**

**Лабораторна робота № 4**

Виконала:

студентка гр. 121-21-2

Векилова Рейхана Шамсіївна

Прийняв:

Мінєєв О. С.

**Дніпро**

**2025**

**Тема:** JUnit. Json.

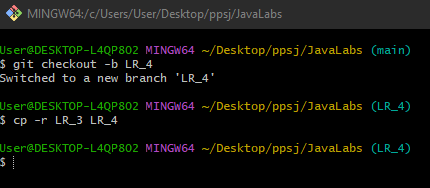
**Мета:** Навчитися працювати з форматом JSON в Java, використовуючи сторонні бібліотеки, такі як Gson або Jackson, та протестувати правильність роботи з JSON за допомогою JUnit.

**Завдання:** Додати до лабораторної роботи 3 можливість запису університету  у формат json, запис цього формату у файл, зчитування цього формату файлу, та створення об'єкту з текстового формату json.  В  проекті повинен бути зроблений JUnit тест,  який буде виглядати наступним чином:  створити об'єкт університет (oldUniversity),  в якому в кожному підрозділі маються два підрозділи нижчого рівня. Наприклад на факультеті дві кафедри, на кожній кафедрі дві групи, на кожній групі два студенти.  Цей об'єкт повинен бути записаний в файл у форматі json. Потім з цього файлу зчитаний та відновлений як newUniversity. В тесті повинні бути порівняні newUniversity та oldUniversity за допомогою методу equals.  Якщо все зроблено правильно то університети повинні бути еквівалентні, а метод equals повинен повернути True.  Для запису та зчитування університету у форматі json повинен бути зроблений клас JsonManager.  Для безпосереднього перетворення університету  у формат json та його відновлення цього формату, можливо використання сторонніх бібліотек наприклад Gson, Jackson  чи будь-яких інших.

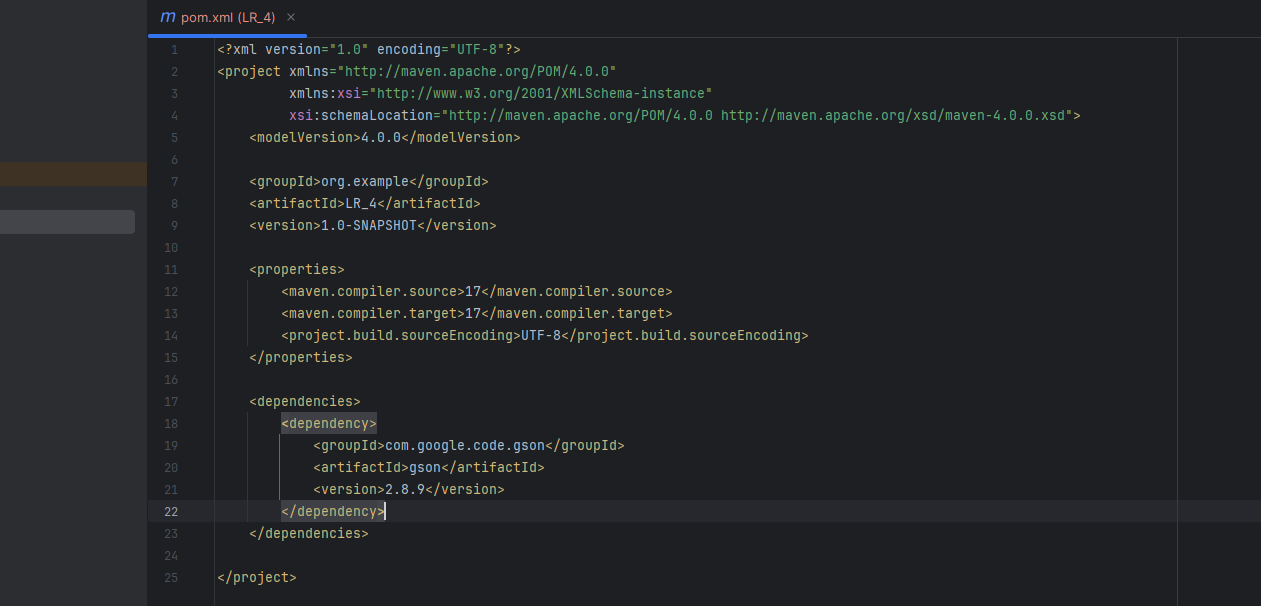
Для початку розробки лабораторної роботи номер 4 повністю скопіювати програмний код лабораторної роботи номер 3. Не змішувати ці роботи не в якому разі.

Хід роботи

1. Створимо нову гілку для цієї лабораторної роботи та скопіюємо файли лабораторної роботи №3 у теку поточної роботи (Рис. 1).

Рис. 1 — Результат виконання пункту 1

2. Додамо бібліотеку Gson у pom.xml (Рис. 2).

Рис. 2 — Результат виконання пункту 2

3. Створимо клас JsonManager, який буде зберігати та відновлювати університет у JSON-файл university.json.

*Лістинг «JsonManager.java»*

package org.example.json;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import org.example.model.University;

import org.example.model.Human;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class JsonManager {

private static final String FILE\_PATH = "university.json";

private static final Gson gson = new GsonBuilder()

.registerTypeAdapter(Human.class, new HumanAdapter()) // Реєструємо адаптер

.setPrettyPrinting()

.create();

public static void saveToJson(University university) {

try (FileWriter writer = new FileWriter(FILE\_PATH)) {

gson.toJson(university, writer);

System.out.println("Університет збережено у JSON-файл.");

} catch (IOException e) {

e.printStackTrace();

}

}

public static University loadFromJson() {

try (FileReader reader = new FileReader(FILE\_PATH)) {

return gson.fromJson(reader, University.class);

} catch (IOException e) {

e.printStackTrace();

return null;

}

}

}

4. Додамо залежності для підключення JUnit (Рис. 3).

Рис. 3 — Результат виконання пункту 4

5. Створимо HumanAdapter, який під час десеріалізації визначатиме, який підклас (Student або Professor) потрібно створити.

*Лістинг «HumanAdapter.java»*

package org.example.json;

import com.google.gson.\*;

import org.example.model.Human;

import org.example.model.Professor;

import org.example.model.Student;

import java.lang.reflect.Type;

public class HumanAdapter implements JsonDeserializer<Human>, JsonSerializer<Human> {

@Override

public Human deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException {

JsonObject jsonObject = json.getAsJsonObject();

String className = jsonObject.get("className").getAsString();

if ("Professor".equals(className)) {

return context.deserialize(json, Professor.class);

} else if ("Student".equals(className)) {

return context.deserialize(json, Student.class);

}

throw new JsonParseException("Unknown class: " + className);

}

@Override

public JsonElement serialize(Human src, Type typeOfSrc, JsonSerializationContext context) {

JsonObject jsonObject = context.serialize(src).getAsJsonObject();

jsonObject.addProperty("className", src.getClass().getSimpleName());

return jsonObject;

}

}

6. Додамо JUnit-тест.

*Лістинг «JsonManagerTest.java»*

package org.example.json;

import org.example.controller.UniversityCreator;

import org.example.model.University;

import org.junit.jupiter.api.Test;

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

public class JsonManagerTest {

@Test

public void testJsonSerialization() {

University oldUniversity = UniversityCreator.createTypicalUniversity();

System.out.println("Оригінальний університет:");

System.out.println(oldUniversity);

JsonManager.saveToJson(oldUniversity);

System.out.println("\n JSON-файл успішно збережено.");

University newUniversity = JsonManager.loadFromJson();

System.out.println("\nВідновлений університет:");

System.out.println(newUniversity);

assertNotNull(newUniversity, " Помилка: newUniversity є null");

assertEquals(oldUniversity.toString(), newUniversity.toString(), " Університети не збігаються");

System.out.println("\n ТЕСТ ПРОЙДЕНО УСПІШНО! Університети однакові.");

}

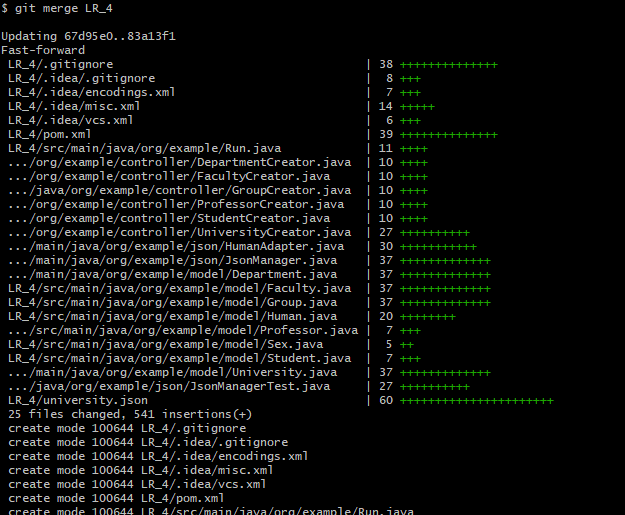
}

7. Протестуємо проєкт (Рис. 4).

Рис. 4 — Результат роботи програми

Тестування показало працездатність розробленої програми.

8. Додаємо файли лабораторної роботи до GitHub репозиторію та мерджимо гілки (Рис. 5).

Рис. 5 — Результат виконання пункту 8

**Висновки:** у ході виконання лабораторної роботи навчилася працювати з форматом JSON в Java, використовуючи сторонні бібліотеки, такі як Gson або Jackson, та протестувала правильність роботи з JSON за допомогою JUnit.