Міністерство науки і освіти України

Національний технічний університет

«Дніпровська Політехника»



**Звіт з лабораторної роботи №4**

**З дисципліни «Поглиблене програмування в середовищі Java»**

Виконав студент:

групи 121-21-1

Земляний Артем Сергійович

Прийняв:

Мінєєв О.С.

**м. Дніпро**

**2025 р.**

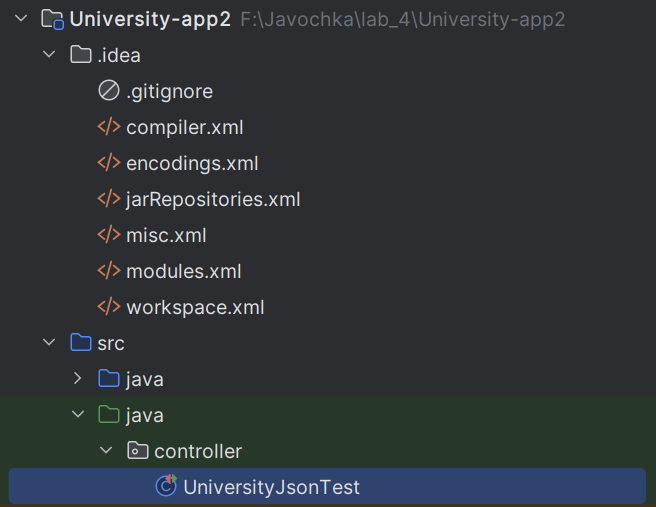
**Лабораторна робота №4**

**JUnit. Json**

**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.

**Виконання лабораторної роботи:**

Далі пишемо код, котрий буде записувати інформацію у форматі json, записує цей формат у файл, та зчитує з нього інформацію.  
  


package controller;

import model.\*;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.io.TempDir; // Для створення тимчасової папки

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

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

public class UniversityJsonTest {

private JsonManager jsonManager;

private University oldUniversity;

private final String testFileName = "test\_university.json";

private Path testFilePath; // Використовуємо Path для кращої роботи з файлами

@TempDir // JUnit створить тимчасову директорію для тестів

Path tempDir;

@BeforeEach // Метод, що виконується перед кожним тестом

void setUp() {

jsonManager = new JsonManager();

oldUniversity = createTestUniversity(); // Створюємо тестовий університет

testFilePath = tempDir.resolve(testFileName); // Створюємо шлях до файлу в тимчасовій папці

}

@AfterEach // Метод, що виконується після кожного тесту

void tearDown() throws IOException {

// Видаляємо тимчасовий файл, якщо він існує

Files.*deleteIfExists*(testFilePath);

// Тимчасова папка @TempDir видаляється автоматично JUnit

}

@Test

void testUniversitySerializationDeserializationEquality() {

try {

// 1. Зберегти oldUniversity у файл

System.*out*.println("Збереження університету у файл: " + testFilePath);

jsonManager.saveToFile(oldUniversity, testFilePath.toString());

*assertTrue*(Files.*exists*(testFilePath), "JSON файл повинен бути створений");

System.*out*.println("Університет збережено.");

// 2. Завантажити університет з файлу як newUniversity

System.*out*.println("Завантаження університету з файлу...");

University newUniversity = jsonManager.loadFromFile(testFilePath.toString());

*assertNotNull*(newUniversity, "Завантажений університет не повинен бути null");

System.*out*.println("Університет завантажено.");

// 3. Порівняти oldUniversity та newUniversity за допомогою equals()

System.*out*.println("Порівняння оригінального та завантаженого університетів...");

*assertEquals*(oldUniversity, newUniversity,

"Оригінальний та завантажений університети повинні бути еквівалентними (equals)");

// Додаткова перевірка: вони не повинні бути одним і тим же об'єктом у пам'яті

*assertNotSame*(oldUniversity, newUniversity,

"Оригінальний та завантажений університети не повинні бути одним і тим же об'єктом в пам'яті");

System.*out*.println("Тест успішно пройдено!");

} catch (IOException e) {

// Якщо сталася помилка IO, тест повинен впасти

*fail*("Помилка під час роботи з файлом JSON: " + e.getMessage());

}

}

*/\*\**

*\* Допоміжний метод для створення тестового університету зі структурою 2x2x2.*

*\* Університет -> 2 Факультети -> 4 Кафедри -> 8 Груп -> 16 Студентів*

*\*/*

private University createTestUniversity() {

HumanCreator humanCreator = new HumanCreator();

StudentCreator studentCreator = new StudentCreator();

GroupCreator groupCreator = new GroupCreator();

DepartmentCreator departmentCreator = new DepartmentCreator();

FacultyCreator facultyCreator = new FacultyCreator();

UniversityCreator universityCreator = new UniversityCreator();

// Керівники

Human rector = humanCreator.createHuman("Ректор", "Ректоренко", "Ректорович", Sex.*MALE*);

University university = universityCreator.createUniversity("Тестовий Університет", rector);

for (int f = 1; f <= 2; f++) { // 2 Факультети

Human dean = humanCreator.createHuman("Декан" + f, "Деканенко" + f, "Деканович" + f, f % 2 == 0 ? Sex.*FEMALE* : Sex.*MALE*);

Faculty faculty = facultyCreator.createFaculty("Факультет " + f, dean);

for (int d = 1; d <= 2; d++) { // 2 Кафедри на факультет

int depId = (f - 1) \* 2 + d;

Human headDep = humanCreator.createHuman("ЗавКаф" + depId, "Кафенко" + depId, "Кафедрович" + depId, Sex.*MALE*);

Department department = departmentCreator.createDepartment("Кафедра " + depId, headDep);

for (int g = 1; g <= 2; g++) { // 2 Групи на кафедру

int groupId = (depId - 1) \* 2 + g;

// Створимо одного студента заздалегідь, щоб призначити старостою

Student headStudent = studentCreator.createStudent("Староста" + groupId, "Старостенко" + groupId, "Групович" + groupId, Sex.*FEMALE*);

Group group = groupCreator.createGroup("Група " + groupId, headStudent); // Староста

// Додамо старосту до списку студентів групи

group.addStudent(headStudent);

for (int s = 1; s <= 1; s++) { // Ще 1 студент у групу (разом буде 2)

int studentId = (groupId -1 ) \* 2 + s + 1; // +1 бо староста вже є (номер 1)

Student student = studentCreator.createStudent("Студент" + studentId, "Студененко" + studentId, "Студенович" + studentId, s % 2 == 0 ? Sex.*FEMALE* : Sex.*MALE*);

group.addStudent(student);

}

department.addGroup(group); // Додати групу до кафедри

}

faculty.addDepartment(department); // Додати кафедру до факультету

}

university.addFaculty(faculty); // Додати факультет до університету

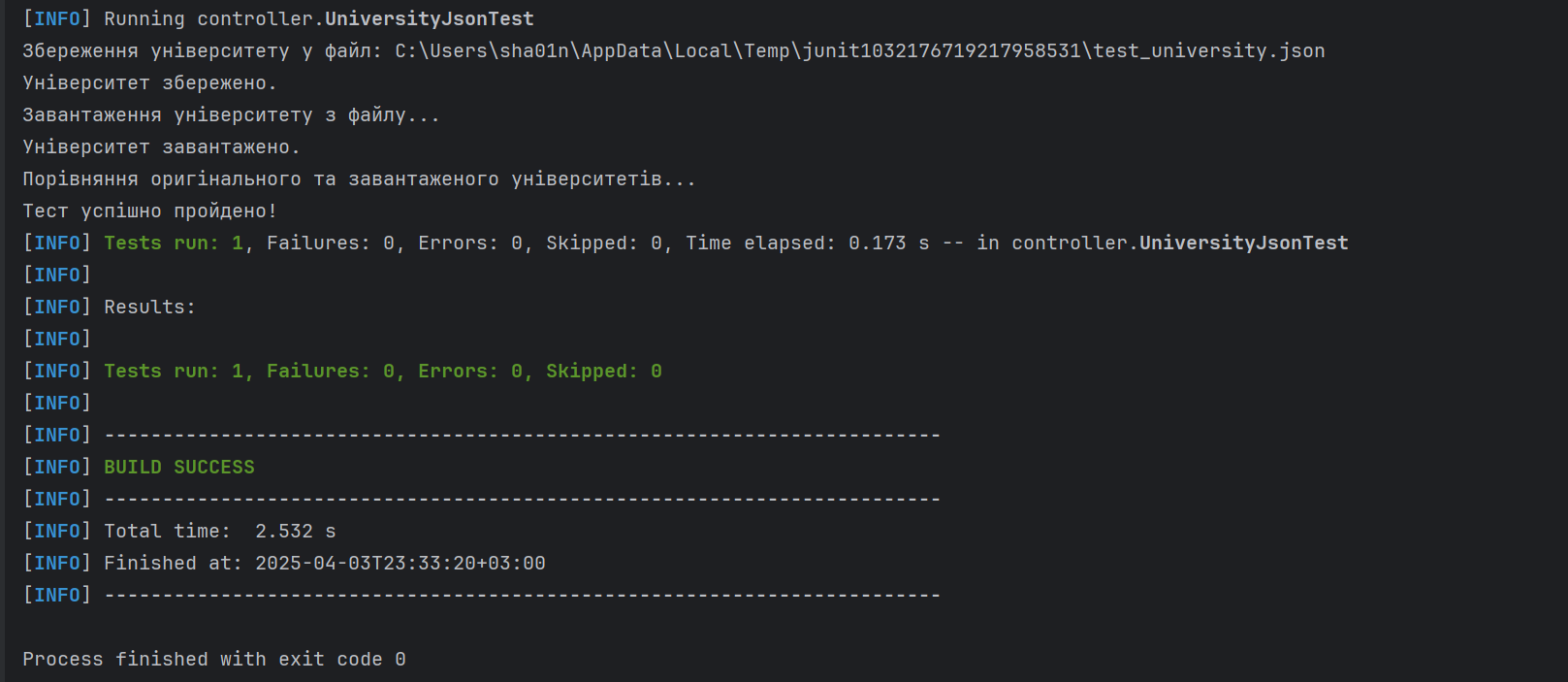
}

return university;

}

}

І перевіряємо:



**Висновок:** під час виконання лабораторної роботи, ми навчились записувати інформацію у форматі json, записувати цей формат у файл, та зчитувати з нього інформацію. Також освоїли тестування за допогою Junit та перевірку коректності обробки даних.