**Міністерство освіти і науки України**

**Національний технічний університет**

**«Дніпровська політехніка»**

****

**ЗВІТ**

**про виконання практичних робіт**

**з дисципліни**

**«Поглиблене програмування в середовищі Java»**

**Лабораторна робота № 4**

Виконав:

студент гр. 121-21-2

Ремез Максим Валентинович

Перевірив:

Доц. Мінєєв О.С.

**Дніпро**

**2025**

**Тема:** 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.

**Код програми:**

JsonManager.java

package org.example.utils;  
  
import com.google.gson.Gson;  
import com.google.gson.GsonBuilder;  
import org.example.model.University;  
  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class JsonManager {  
 private static final Gson *gson* = new GsonBuilder().setPrettyPrinting().create();  
  
 public static void saveToJson(University university, String filePath) {  
 try (FileWriter writer = new FileWriter(filePath)) {  
 *gson*.toJson(university, writer);  
 System.*out*.println("✅ Університет успішно збережено у файл: " + filePath);  
 } catch (IOException e) {  
 System.*err*.println("❌ Помилка при записі у файл: " + filePath);  
 e.printStackTrace();  
 }  
 }  
   
 public static University loadFromJson(String filePath) {  
 try (FileReader reader = new FileReader(filePath)) {  
 University university = *gson*.fromJson(reader, University.class);  
 System.*out*.println("✅ Університет успішно завантажено з файлу: " + filePath);  
 return university;  
 } catch (IOException e) {  
 System.*err*.println("❌ Помилка при читанні файлу: " + filePath);  
 e.printStackTrace();  
 return null;  
 }  
 }  
}

JsonManagerTest.java

package tests;  
  
import org.example.model.\*;  
import org.junit.jupiter.api.Test;  
import org.example.utils.JsonManager;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class JsonManagerTest {  
  
 private static final String *FILE\_PATH* = "university.json";  
  
 @Test  
 public void testJsonSerializationAndDeserialization() {  
 // Створюємо тестовий університет  
 University oldUniversity = createTestUniversity();  
  
 // Записуємо в JSON  
 JsonManager.*saveToJson*(oldUniversity, *FILE\_PATH*);  
  
 // Читаємо з JSON  
 University newUniversity = JsonManager.*loadFromJson*(*FILE\_PATH*);  
  
 // Перевіряємо, що університети рівні  
 *assertEquals*(oldUniversity, newUniversity);  
 }  
  
 private University createTestUniversity() {  
 Human rector = new Human("Іван", "Петров", "Олексійович", Sex.*MALE*) {};  
 University university = new University("Тестовий університет", rector);  
  
 for (int i = 1; i <= 2; i++) {  
 Human dean = new Human("Декан" + i, "Прізвище", "Ім'я", Sex.*FEMALE*) {};  
 Faculty faculty = new Faculty("Факультет " + i, dean);  
  
 for (int j = 1; j <= 2; j++) {  
 Human headOfDepartment = new Human("ЗавКаф" + j, "Прізвище", "Ім'я", Sex.*MALE*) {};  
 Department department = new Department("Кафедра " + j, headOfDepartment);  
  
 for (int k = 1; k <= 2; k++) {  
 Human headOfGroup = new Human("Староста" + k, "Прізвище", "Ім'я", Sex.*FEMALE*) {};  
 Group group = new Group("Група " + k, headOfGroup);  
  
 for (int m = 1; m <= 2; m++) {  
 Student student = new Student("Студент" + m, "Прізвище", "Ім'я", Sex.*MALE*);  
 group.addStudent(student);  
 }  
  
 department.addGroup(group);  
 }  
  
 faculty.addDepartment(department);  
 }  
  
 university.addFaculty(faculty);  
 }  
  
 return university;  
 }  
}

**Оновлені model**

Human.java

package org.example.model;  
  
import java.util.Objects;  
  
public abstract class Human {  
 protected String firstName;  
 protected String lastName;  
 protected String middleName;  
 protected Sex gender;  
  
 public Human(String firstName, String lastName, String middleName, Sex gender) {  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.middleName = middleName;  
 this.gender = gender;  
 }  
  
 // Додаємо геттери  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public Sex getGender() {  
 return gender;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Human human = (Human) o;  
 return Objects.*equals*(firstName, human.firstName) &&  
 Objects.*equals*(lastName, human.lastName) &&  
 Objects.*equals*(middleName, human.middleName) &&  
 gender == human.gender;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName, middleName, gender);  
 }  
  
 @Override  
 public String toString() {  
 return firstName + " " + middleName + " " + lastName + " (" + gender + ")";  
 }  
}

Student.java

package org.example.model;  
  
import java.util.Objects;  
  
public class Student extends Human {  
 public Student(String firstName, String lastName, String middleName, Sex gender) {  
 super(firstName, lastName, middleName, gender);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Student student = (Student) o;  
 return Objects.*equals*(getFirstName(), student.getFirstName()) &&  
 Objects.*equals*(getLastName(), student.getLastName()) &&  
 Objects.*equals*(getMiddleName(), student.getMiddleName()) &&  
 getGender() == student.getGender();  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getFirstName(), getLastName(), getMiddleName(), getGender());  
 }  
}

Group.java

package org.example.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Group {  
 private String name;  
 private Human head;  
 private List<Student> students;  
  
 public Group(String name, Human head) {  
 this.name = name;  
 this.head = head;  
 this.students = new ArrayList<>();  
 }  
  
 public void addStudent(Student student) {  
 students.add(student);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Group group = (Group) o;  
 return Objects.*equals*(name, group.name) &&  
 Objects.*equals*(students, group.students);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, students);  
 }  
}

Department.java

package org.example.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Department {  
 private String name;  
 private Human head;  
 private List<Group> groups;  
  
 public Department(String name, Human head) {  
 this.name = name;  
 this.head = head;  
 this.groups = new ArrayList<>();  
 }  
  
 public void addGroup(Group group) {  
 groups.add(group);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Department that = (Department) o;  
 return Objects.*equals*(name, that.name) &&  
 Objects.*equals*(groups, that.groups);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, groups);  
 }  
}

Faculty.java

package org.example.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class Faculty {  
 private String name;  
 private Human head;  
 private List<Department> departments;  
  
 public Faculty(String name, Human head) {  
 this.name = name;  
 this.head = head;  
 this.departments = new ArrayList<>();  
 }  
  
 public void addDepartment(Department department) {  
 departments.add(department);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Faculty faculty = (Faculty) o;  
 return Objects.*equals*(name, faculty.name) &&  
 Objects.*equals*(departments, faculty.departments);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, departments);  
 }  
}

University.java

package org.example.model;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class University {  
 private String name;  
 private Human head;  
 private List<Faculty> faculties;  
  
 public University(String name, Human head) {  
 this.name = name;  
 this.head = head;  
 this.faculties = new ArrayList<>();  
 }  
  
 public void addFaculty(Faculty faculty) {  
 faculties.add(faculty);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 University that = (University) o;  
 return Objects.*equals*(name, that.name) &&  
 Objects.*equals*(faculties, that.faculties);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, faculties);  
 }  
  
}

**Результат роботи програми:**

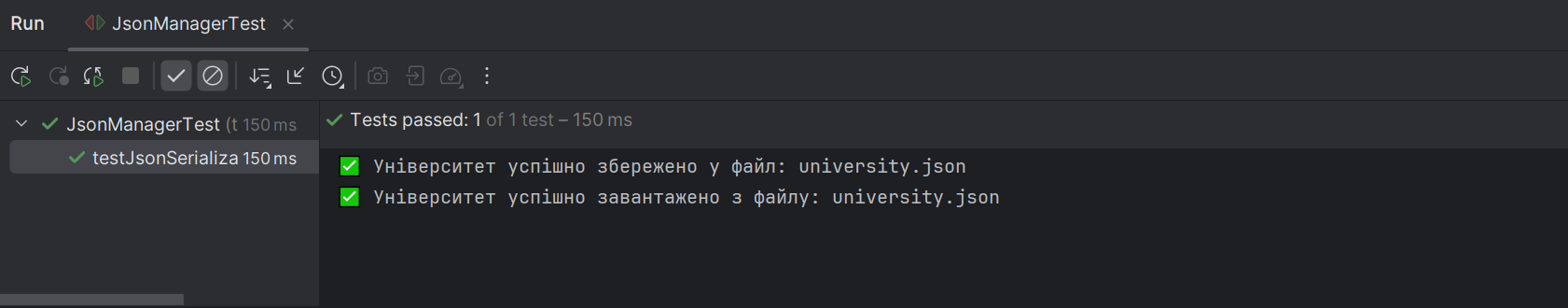
****

Рисунок 1 – Результат проходження тесту порівняння