**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 125-21-3**

**Войтович Микита**

**Прийняв:**

**Мінєєв О.С.**

**м. Дніпро**

**2025 р.**

**Войтович Микита 125-21-3**

**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:**

В папці model редагуємо наші минулі файли, для підтримки json:

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.List;  
import java.util.Objects;  
import java.util.stream.Collectors;  
  
// Клас кафедри  
public class Department extends OrganizationUnit {  
 private List<Group> groups;  
  
 public Department(String title, Person director, List<Group> groups) {  
 super(title, director);  
 this.groups = groups;  
 }  
  
 // Безпараметровий конструктор (необхідний для Gson)  
 public Department() {}  
  
 // Отримання списку груп  
 public List<Group> obtainGroups() { return groups; }  
  
 // Сетер для JSON-десеріалізації  
 public void assignGroups(List<Group> teams) {  
 this.groups = groups;  
 }  
  
 // Перевизначений equals() для порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Department department = (Department) obj;  
 return Objects.*equals*(title, department.title) &&  
 Objects.*equals*(director, department.director) &&  
 Objects.*equals*(groups, department.groups);  
 }  
  
 // Перевизначений hashCode() для коректної роботи у HashSet, HashMap  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, director, groups);  
 }  
  
 @Override  
 public String toString() {  
 String groupList = groups.isEmpty() ? "No groups" : groups.stream()  
 .map(Group::toString)  
 .collect(Collectors.*joining*(",\n "));  
 return String.*format*("Department{name='%s', Head=%s, Groups=[\n %s\n]}", title, director, groupList);  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.List;  
import java.util.stream.Collectors;  
import java.util.Objects;  
  
// Клас факультету  
public class Faculty extends OrganizationUnit {  
 private List<Department> departments;  
  
 public Faculty(String title, Person director, List<Department> departments) {  
 super(title, director);  
 this.departments = departments;  
 }  
  
 // Безпараметровий конструктор (необхідний для Gson)  
 public Faculty() {}  
  
 // Отримання списку кафедр  
 public List<Department> obtainDepartments() { return departments; }  
  
 // Сетер для JSON-десеріалізації  
 public void assignDepartments(List<Department> departments) {  
 this.departments = departments;  
 }  
  
 // equals() для порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Faculty faculty = (Faculty) obj;  
 return Objects.*equals*(title, faculty.title) &&  
 Objects.*equals*(director, faculty.director) &&  
 Objects.*equals*(departments, faculty.departments);  
 }  
  
 // hashCode() для коректної роботи в HashSet, HashMap  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, director, departments);  
 }  
  
 @Override  
 public String toString() {  
 String departmentList = departments.isEmpty() ? "No departments" : departments.stream()  
 .map(Department::toString)  
 .collect(Collectors.*joining*(",\n "));  
 return String.*format*("Faculty{name='%s', Head=%s, Departments=[\n %s\n]}", title, director, departmentList);  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.List;  
import java.util.Objects;  
  
// Клас групи  
public class Group extends OrganizationUnit {  
 private List<Student> students;  
  
 public Group(String title, Person director, List<Student> students) {  
 super(title, director);  
 this.students = students;  
 }  
  
 // Безпараметровий конструктор (необхідний для Gson)  
 public Group() {}  
  
 // Отримання списку студентів у групі  
 public List<Student> obtainStudents() {  
 return students;  
 }  
  
 // Сетер для JSON-десеріалізації  
 public void assignStudents(List<Student> students) {  
 this.students = students;  
 }  
  
 // equals() для порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Group group = (Group) obj;  
 return Objects.*equals*(title, group.title) &&  
 Objects.*equals*(director, group.director) &&  
 Objects.*equals*(students, group.students);  
 }  
  
 // hashCode() для коректної роботи в HashSet, HashMap  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, director, students);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("Group{name='%s', Head=%s, Students=%s}", title, director, students);  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.Objects;  
  
// Абстрактний клас базового підрозділу  
public abstract class OrganizationUnit {  
 protected String title;  
 protected Person director;  
  
 public OrganizationUnit(String title, Person director) {  
 this.title = title;  
 this.director = director;  
 }  
  
 // Безпараметровий конструктор (необхідний для Gson)  
 public OrganizationUnit() {}  
  
 public String getTitle() {  
 return title;  
 }  
  
 public Person getDirector() {  
 return director;  
 }  
  
 // Сетери для JSON-десеріалізації  
 public void assignTitle(String name) {  
 this.title = title;  
 }  
  
 public void assignDirector(Person head) {  
 this.director = director;  
 }  
  
 // equals() для порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 OrganizationUnit that = (OrganizationUnit) obj;  
 return Objects.*equals*(title, that.title) &&  
 Objects.*equals*(director, that.director);  
 }  
  
 // hashCode() для коректної роботи в HashSet, HashMap  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, director);  
 }  
  
 @Override  
 public String toString() {  
 return title + " (Head: " + director + ")";  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.Objects;  
  
// Абстрактний клас особи  
public abstract class Person {  
 private String name;  
 private String surname;  
 private String patronymic;  
 private Gender gender;  
  
 public enum Gender { *MALE*, *FEMALE* }  
  
 public Person(String name, String surname, String patronymic, Gender gender) {  
 this.name = name;  
 this.surname = surname;  
 this.patronymic = patronymic;  
 this.gender = gender;  
 }  
  
 // Безпараметровий конструктор (необхідний для Gson)  
 public Person() {}  
  
 public String getName() { return name; }  
 public String getSurname() { return surname; }  
 public String getPatronymic() { return patronymic; }  
 public Gender getGender() { return gender; }  
  
 // Сетери для JSON-десеріалізації  
 public void assignName(String name) { this.name = name; }  
 public void assignSurname(String surname) { this.surname = surname; }  
 public void assignPatronymic(String patronymic) { this.patronymic = patronymic; }  
 public void assignGender(Gender gender) { this.gender = gender; }  
  
 // equals() для порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Person person = (Person) obj;  
 return Objects.*equals*(name, person.name) &&  
 Objects.*equals*(surname, person.surname) &&  
 Objects.*equals*(patronymic, person.patronymic) &&  
 gender == person.gender;  
 }  
  
 // hashCode() для коректної роботи в HashSet, HashMap  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, surname, patronymic, gender);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s %s %s (%s)", name, surname, patronymic, gender);  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
// Клас студента  
public class Student extends Person {  
 public Student(String name, String surname, String patronymic, Gender gender) {  
 super(name, surname, patronymic, gender);  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s", super.toString());  
 }  
}

package Voytovic\_Mykyta.lab4.model;  
  
import java.util.List;  
import java.util.Objects;  
  
// Клас університету  
public class University extends OrganizationUnit {  
 private List<Faculty> faculties;  
  
 public University(String title, Person director, List<Faculty> faculties) {  
 super(title, director);  
 this.faculties = faculties;  
 }  
  
 // Порожній конструктор (необхідний для Gson)  
 public University() {}  
  
 // Геттери та сеттери  
 public List<Faculty> obtainFaculties() {  
 return faculties;  
 }  
  
 public void assignFaculties(List<Faculty> faculties) {  
 this.faculties = faculties;  
 }  
  
 // equals() для коректного порівняння об'єктів  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 University that = (University) obj;  
 return Objects.*equals*(title, that.title) &&  
 Objects.*equals*(director, that.director) &&  
 Objects.*equals*(faculties, that.faculties);  
 }  
  
 // hashCode() для унікальної ідентифікації об'єкта  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(title, director, faculties);  
 }  
  
  
 @Override  
 public String toString() {  
 return String.*format*("University{name='%s', Rector=%s, Faculties=%s}", title, director, faculties);  
 }  
}

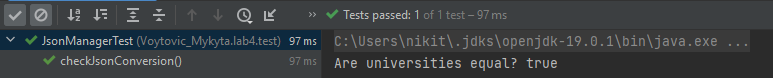
Далі пишемо код, котрий буде записувати інформацію у форматі json, записує цей формат у файл, та зчитує з нього інформацію.

package Voytovic\_Mykyta.lab4.json;  
  
import com.google.gson.\*;  
import com.google.gson.reflect.TypeToken;  
import Voytovic\_Mykyta.lab4.model.Person;  
import Voytovic\_Mykyta.lab4.model.Student;  
  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.lang.reflect.Type;  
  
public class JsonManager {  
 private static final Gson *gson* = new GsonBuilder()  
 .registerTypeAdapter(Person.class, new HumanConverter()) // Користувацький конвертер  
 .setPrettyPrinting()  
 .create();  
  
 // Метод для перетворення об'єкта у JSON-рядок  
 public static String convertToJson(Object entity) {  
 return *gson*.toJson(entity);  
 }  
  
 // Метод для перетворення JSON-рядка у об'єкт  
 public static <T> T convertFromJson(String jsonContent, Class<T> targetClass) {  
 return *gson*.fromJson(jsonContent, targetClass);  
 }  
  
 // Метод для збереження об'єкта у JSON-файл  
 public static void writeToJson(Object entity, String fileName) {  
 try (FileWriter writer = new FileWriter(fileName)) {  
 *gson*.toJson(entity, writer);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 // Метод для завантаження об'єкта з JSON-файлу  
 public static <T> T readFromJson(String fileName, Class<T> targetClass) {  
 try (FileReader reader = new FileReader(fileName)) {  
 return *gson*.fromJson(reader, targetClass);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 return null;  
 }  
 }  
}  
  
// Користувацький конвертер для Human  
class HumanConverter implements JsonSerializer<Person>, JsonDeserializer<Person> {  
 @Override  
 public JsonElement serialize(Person src, Type typeOfSrc, JsonSerializationContext context) {  
 JsonObject jsonObject = context.serialize(src).getAsJsonObject();  
 jsonObject.addProperty("category", src.getClass().getSimpleName()); // Додаємо категорію  
 return jsonObject;  
 }  
  
 @Override  
 public Person deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException {  
 JsonObject jsonObject = json.getAsJsonObject();  
 String category = jsonObject.get("category").getAsString(); // Отримуємо категорію  
  
 if ("Student".equals(category)) {  
 return context.deserialize(jsonObject, Student.class);  
 } else {  
 throw new JsonParseException("Unknown type: " + category);  
 }  
 }  
}

І тепер пишемо текст, для перевірки:

package Voytovic\_Mykyta.lab4.test;  
  
import Voytovic\_Mykyta.lab4.json.JsonManager;  
import Voytovic\_Mykyta.lab4.model.\*;  
import org.junit.jupiter.api.Test;  
import java.util.Arrays;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class JsonManagerTest {  
 @Test  
 void checkJsonConversion() {  
 // Створення керівника університету  
 Person director = new Student("Michael", "Anderson", "", Person.Gender.*MALE*);  
  
 // Створення початкового університету  
 University oldUniversity = new University("Global Tech University", director, Arrays.*asList*(  
 new Faculty("Computer Science", new Student("Dr. Emily", "White", "", Person.Gender.*FEMALE*), Arrays.*asList*(  
 new Department("Artificial Intelligence", new Student("Dr. Sophia", "Taylor", "", Person.Gender.*FEMALE*), Arrays.*asList*(  
 new Group("AI-301", new Student("James", "Smith", "", Person.Gender.*MALE*), Arrays.*asList*(  
 new Student("James", "Smith", "", Person.Gender.*MALE*),  
 new Student("Olivia", "Johnson", "", Person.Gender.*FEMALE*)  
 )),  
 new Group("AI-302", new Student("Lucas", "Brown", "", Person.Gender.*MALE*), Arrays.*asList*(  
 new Student("Lucas", "Brown", "", Person.Gender.*MALE*),  
 new Student("Emma", "Williams", "", Person.Gender.*FEMALE*)  
 ))  
 )),  
 new Department("Cybersecurity", new Student("Dr. Robert", "Davis", "", Person.Gender.*MALE*), Arrays.*asList*(  
 new Group("CS-401", new Student("Charlotte", "Miller", "", Person.Gender.*FEMALE*), Arrays.*asList*(  
 new Student("Charlotte", "Miller", "", Person.Gender.*FEMALE*),  
 new Student("Daniel", "Moore", "", Person.Gender.*MALE*)  
 )),  
 new Group("CS-402", new Student("Henry", "Taylor", "", Person.Gender.*MALE*), Arrays.*asList*(  
 new Student("Henry", "Taylor", "", Person.Gender.*MALE*),  
 new Student("Sophia", "Anderson", "", Person.Gender.*FEMALE*)  
 ))  
 ))  
 ))  
 ));  
  
 // Збереження у JSON-файл  
 String jsonFilePath = "university.json";  
 JsonManager.*writeToJson*(oldUniversity, jsonFilePath);  
  
 // Зчитування з JSON-файлу  
 University newUniversity = JsonManager.*readFromJson*(jsonFilePath, University.class);  
  
 // Перевірка чи рівні об'єкти oldUniversity і newUniversity  
 System.*out*.println("Are universities equal? " + oldUniversity.equals(newUniversity));  
  
 // Перевірка відповідності початкового та відновленого університетів  
 *assertEquals*(oldUniversity, newUniversity, "Помилка: збереження або відновлення JSON виконано некоректно");  
 }  
}

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.