

**Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»**

**ІНСТИТУТ ЕЛЕКТРОЕНЕРГЕТИКИ  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
Кафедра інформаційних технологій та комп'ютерної інженерії**



**Дисципліни “Програмування в середовищі Java”**

Виконав: студент групи 123-21-2  
Яковенко Д.Е.

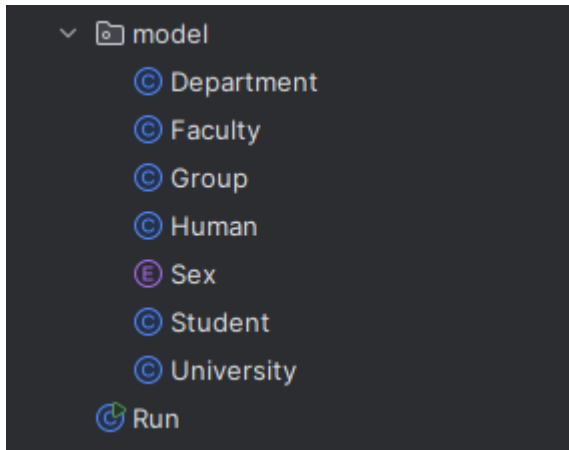
Перевірив:  
Мінєєв Олександр Сергійович

**Дніпро  
2023**

## Лабораторна робота номер 4. JUnit. Json

Копіюємо усі класи з 3 лабораторної роботи

Пакет model



Клас Human це базовий клас, що представляє людину. Використовується для студентів, керівників груп, кафедр та факультетів.

```
package model;

import java.util.Objects;

public class Human { 1 inheritor  Danil Yakovenko
    private String firstName; 6 usages
    private String lastName; 6 usages
    private String patronymic; 6 usages
    private Sex sex; 6 usages
}
```

Клас Sex. Перерахування, що визначає стать людини

```
package model;

public enum Sex { 14 usages  Danil Yakovenko
    MALE, FEMALE 4 usages
}
```

Клас Student є підкласом Human, що представляє студента

```

public class Student extends Human { 9 usages  ⚡ Danil Yakovenko
    // Конструктор
    public Student(String firstName, String lastName, String patronymic, Sex sex) { 1 usage  ⚡ Danil Yakovenko
        super(firstName, lastName, patronymic, sex);
    }
}

```

Клас Group містить список студентів та керівника групи

```

public class Group { 8 usages  ⚡ Danil Yakovenko
    private String name; 5 usages
    private Human head; 5 usages
    private List<Student> students; 6 usages
}

```

Клас Department містить список груп

```

public class Department { 8 usages  ⚡ Danil Yakovenko
    private String name; 5 usages
    private Human head; 5 usages
    private List<Group> groups; 6 usages

    // Конструктор
    public Department(String name, Human head) { 1 usage  ⚡ Danil Yakovenko
        this.name = name;
        this.head = head;
        this.groups = new ArrayList<>();
    }
}

```

Клас Faculty включає список кафедр

```

6
7 public class Faculty { 8 usages  ⚡ Danil Yakovenko
8     private String name; 5 usages
9     private Human head; 5 usages
10    private List<Department> departments; 6 usages
11
12    // Конструктор
13    public Faculty(String name, Human head) { 1 usage  ⚡ Danil Yakovenko
14        this.name = name;
15        this.head = head;
16        this.departments = new ArrayList<>();
17    }
}

```

## Клас University

```
public class University { 13 usages  ⤴ Danil Yakovenko
    private String name; 5 usages
    private Human head; 5 usages
    private List<Faculty> faculties; 6 usages

    // Конструктор
    public University(String name, Human head) { 1 usage  ⤴ Danil Yakovenko
        this.name = name;
        this.head = head;
        this.faculties = new ArrayList<>();
    }

    // Метод для додавання факультету
    public void addFaculty(Faculty faculty) { 1 usage  ⤴ Danil Yakovenko
        faculties.add(faculty);
    }
}
```

Пакет controller для створення об'єктів

StudentCreator – створює студентів

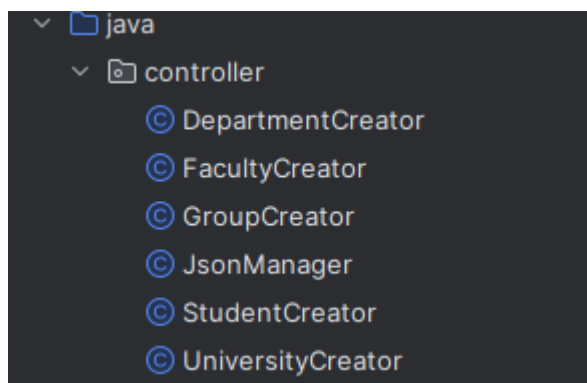
GroupCreator – створює групи

DepartmentCreator – створює кафедри

FacultyCreator – створює факультети

UniversityCreator – створює університет

Ці класи використовують принцип інкапсуляції для зручного створення об'єктів.



Клас JsonManager містить методи для серіалізації (saveUniversityToJson) та десеріалізації (loadUniversityFromJson) університету в JSON

```
1 package controller;
2
3 import com.google.gson.Gson;
4 import com.google.gson.GsonBuilder;
5 import model.University;
6
7 import java.io.FileReader;
8 import java.io.FileWriter;
9 import java.io.IOException;
10
11 public class JsonManager { 3 usages  ▲ Danil Yakovenko
12     private static final Gson gson = new GsonBuilder().setPrettyPrinting().create(); 2 usages
13
14     public static void saveUniversityToJson(University university, String filePath) { 1 usage  ▲ Danil Yakovenko
15         try (FileWriter writer = new FileWriter(filePath)) {
16             gson.toJson(university, writer);
17         } catch (IOException e) {
18             e.printStackTrace();
19         }
20     }
21
22     public static University loadUniversityFromJson(String filePath) { 1 usage  ▲ Danil Yakovenko
23         try (FileReader reader = new FileReader(filePath)) {
24             return gson.fromJson(reader, University.class);
25         } catch (IOException e) {
26             e.printStackTrace();
27             return null;
28         }
29     }
30 }
```

Запуск програми клас Run містить main-метод для створення об'єктів та їх виводу

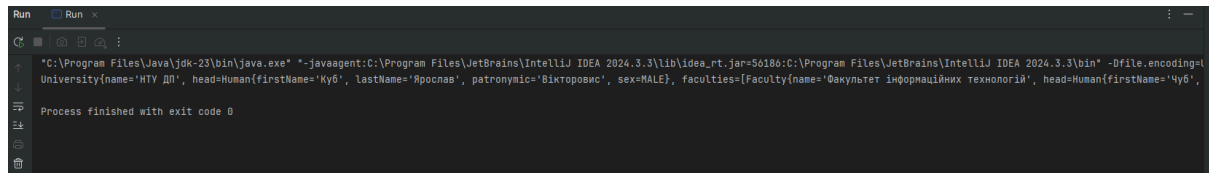
```
1 import controller.*;
2 import model.*;
3
4 public class Run { 1 usage  ▲ Danil Yakovenko
5     public static void main(String[] args) { 1 usage  ▲ Danil Yakovenko
6         // Створення університету
7         UniversityCreator universityCreator = new UniversityCreator();
8         Human rector = new Human( firstName: "Ky6", lastName: "Ярослав", patronymic: "Вікторович", Sex.MALE);
9         University university = universityCreator.createUniversity( name: "НТУ ДП", rector);
10
11         // Створення факультету
12         FacultyCreator facultyCreator = new FacultyCreator();
13         Human dean = new Human( firstName: "Чуб", lastName: "Олена", patronymic: "Олександрівна", Sex.FEMALE);
14         Faculty faculty = facultyCreator.createFaculty( name: "Факультет інформаційних технологій", dean);
15         university.addFaculty(faculty);
16
17         // Створення кафедри
18         DepartmentCreator departmentCreator = new DepartmentCreator();
19         Human headOfDepartment = new Human( firstName: "Олеп", lastName: "Давид", patronymic: "Володимирович", Sex.MALE);
20         Department department = departmentCreator.createDepartment( name: "Кафедра програмного забезпечення", headOfDepartment);
21         faculty.addDepartment(department);
22
23         // Створення групи
24         GroupCreator groupCreator = new GroupCreator();
25         Human headOfGroup = new Human( firstName: "Майя", lastName: "Іванова", patronymic: "Петрівна", Sex.FEMALE);
26         Group group = groupCreator.createGroup( name: "Група 123-21-2", headOfGroup);
27         department.addGroup(group);
28
29         // Створення студентів
30         StudentCreator studentCreator = new StudentCreator();
31         Student student1 = studentCreator.createStudent( firstName: "Плюський", lastName: "Данило", patronymic: "Олександрович", Sex.MALE);
32         Student student2 = studentCreator.createStudent( firstName: "Гладка", lastName: "Анастасія", patronymic: "Ігорівна", Sex.FEMALE);
33         group.addStudent(student1);
34         group.addStudent(student2);
35
36         // Виведення інформації про університет
37         System.out.println(university);
38     }
39 }
```

У цьому класі:

Створюється університет.

Додаються факультет, кафедра, група та студенти.

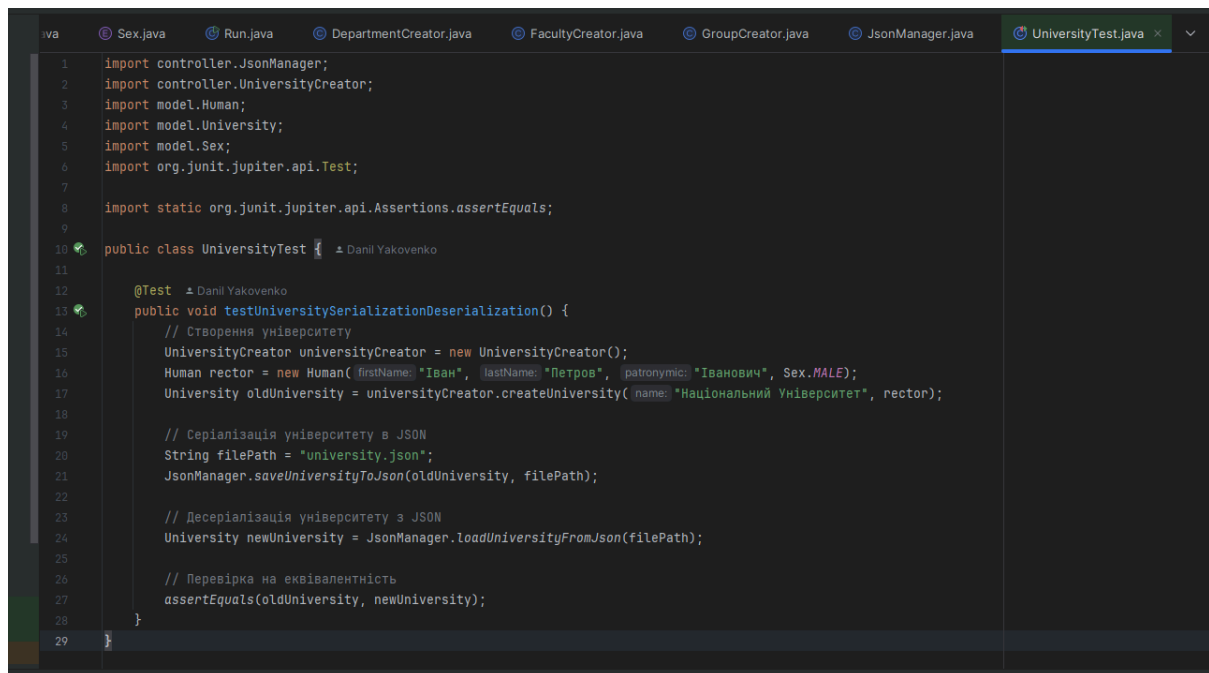
Виводиться інформація про університет.



```
Run
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=56186:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\bin" -Dfile.encoding=UTF-8
University(name='НТУ ДН', head=Human(firstName='Куб', lastName='Ярослав', patronymic='Вікторович', sex=MALE), faculties=[Faculty(name='Факультет інформаційних технологій', head=Human(firstName='Чуб', lastName='Віктор', patronymic='Вікторович', sex=MALE))])
Process finished with exit code 0
```

## Тестування

Клас UniversityTest перевіряє серіалізацію та десеріалізацію JSON



```
import controller.JsonManager;
import controller.UniversityCreator;
import model.Human;
import model.University;
import model.Sex;
import org.junit.jupiter.api.Test;

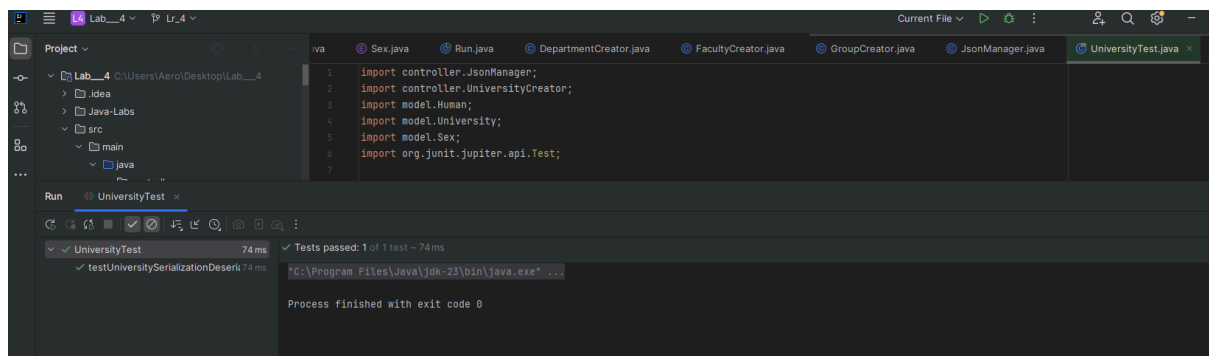
import static org.junit.jupiter.api.Assertions.assertEquals;

public class UniversityTest {
    @Test
    public void testUniversitySerializationDeserialization() {
        // Створення університету
        UniversityCreator universityCreator = new UniversityCreator();
        Human rector = new Human(firstName="Іван", lastName="Петров", patronymic="Іванович", sex=Sex.MALE);
        University oldUniversity = universityCreator.createUniversity(name="Національний університет", rector);

        // Серіалізація університету в JSON
        String filePath = "university.json";
        JsonManager.saveUniversityToJson(oldUniversity, filePath);

        // Десеріалізація університету з JSON
        University newUniversity = JsonManager.loadUniversityFromJson(filePath);

        // Перевірка на еквівалентність
        assertEquals(oldUniversity, newUniversity);
    }
}
```



```
Project: Lab_4
C:\Users\Aero\Desktop\Lab_4
src
main
java

Run: UniversityTest
74 ms
Tests passed: 1 of 1 test - 74 ms
testUniversitySerializationDeser: 74 ms
"C:\Program Files\Java\jdk-23\bin\java.exe" ...
Process finished with exit code 0
```

## Висновок

У лабораторній роботі реалізовано модель університету з використанням ООП та патерну MVC також було додано можливість серіалізації об'єктів у JSON, а також проведено тестування за допомогою JUnit. Код структурований і дотримується принципів чистої архітектури.