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

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

Лабораторна робота №4

з дисципліни:

«Базова Джава»

Виконав:
студент групи 123-21-1
Скоропад Є.В.
Перевірив:
Доцент кафедри САіУ
Мінєєв О.С.

Дніпро
2025

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.

Клас JsonManager

```
package org.university.controller;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import org.university.model.University;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class JsonManager {
    private final Gson gson;

    public JsonManager() {
        this.gson = new GsonBuilder()
                .setPrettyPrinting()
                .create();
    }

    public void writeUniversityToJson(University university, String filePath)
throws IOException {
        try (FileWriter writer = new FileWriter(filePath)) {
            gson.toJson(university, writer);
        }
    }

    public University readUniversityFromJson(String filePath) throws
IOException {
        try (FileReader reader = new FileReader(filePath)) {
            return gson.fromJson(reader, University.class);
        }
    }
}
```

Дописуємо метож equals до кожного класу модуля model:

BaseUnit

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    BaseUnit baseUnit = (BaseUnit) o;
    return name.equals(baseUnit.name) &&
            head.equals(baseUnit.head);
}
```

Department

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Department that = (Department) o;
    return groups.equals(that.groups);
}
```

Faculty

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Faculty faculty = (Faculty) o;
    return departments.equals(faculty.departments);
}
```

Group

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Group group = (Group) o;
    return students.equals(group.students);
}
```

Human

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Human human = (Human) o;
    return firstName.equals(human.firstName) &&
            lastName.equals(human.lastName) &&
            patronymic.equals(human.patronymic) &&
```

```
            sex == human.sex;
}
```

Student

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Student student = (Student) o;
    return studentId.equals(student.studentId) &&
            recordBookNumber.equals(student.recordBookNumber) &&
            birthDate.equals(student.birthDate);
}
```

University

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    University university = (University) o;
    return faculties.equals(university.faculties);
}
```

Клас UniversityJsonTest

```
package org.university;

import org.university.controller.*;
import org.university.model.*;
import java.io.IOException;

public class UniversityJsonTest {
    public void testUniversityJsonSerialization(University oldUniversity)
throws IOException {
        // Test JSON serialization of the passed university
        JsonManager jsonManager = new JsonManager();
        String filePath = "./test_university.json";

        jsonManager.writeUniversityToJson(oldUniversity, filePath);
        University newUniversity =
jsonManager.readUniversityFromJson(filePath);

        // Compare
        if (!oldUniversity.equals(newUniversity)) {
            throw new AssertionError("Universities are not equal");
        }
    }
}
```

Клас Run

```
package org.university;

import org.university.controller.*;
import org.university.model.*;

import org.university.view.UniversityView;

import java.io.IOException;
import java.util.Scanner;

public class Run {
    public static void main(String[] args) {
        // Create a typical university and output it
        University university = createTypicalUniversity();
        UniversityView view = new UniversityView();
        view.displayUniversity(university);

        // Test JSON serialization
        UniversityJsonTest test = new UniversityJsonTest();
        try {
            test.testUniversityJsonSerialization(university); // Pass the
university to the test
            System.out.println("University JSON serialization successful");
        } catch (IOException e) {
            System.out.println("Error test execution: " + e.getMessage());
        } catch (AssertionError e) {
            System.out.println("Test failed: " + e.getMessage());
        }
    }

    public static University createTypicalUniversity() {
        // Create university rector
        Human rector = new HumanCreator("John", "Smith", "William",
Sex.MALE).create();

        // Create faculty dean
        Human dean = new HumanCreator("Michael", "Johnson", "Robert",
Sex.MALE).create();

        // Create department head
        Human departmentHead = new HumanCreator("Sarah", "Davis", "Jane",
Sex.FEMALE).create();

        // Create group curator
        Human groupCurator = new HumanCreator("Oleksand", "Mineev",
"Sergiyovich", Sex.MALE).create();

        // Create students
        StudentCreator[] students1 = {
                new StudentCreator("Sasha", "TikTok", "Olegovna", Sex.FEMALE,
"123-21-1", "RB1231", "2003-05-15"),
                new StudentCreator("Misha", "Bulkin", "Ilonovich", Sex.MALE,
"123-21-2", "RB1232", "2004-08-22")
        };
        StudentCreator[] students2 = {
                new StudentCreator("Justin", "Biber", "Grigirovich",
Sex.MALE, "122-21-1", "RB1224", "2003-03-10"),
                new StudentCreator("Max", "Pain", "MacCoffe", Sex.MALE, "122-
21-2", "RB1212", "2003-11-05")
        };
```

```
        StudentCreator[] students3 = {
                new StudentCreator("Anna", "Kovalenko", "Viktorivna",
Sex.FEMALE, "121-21-1", "RB1241", "2004-07-19"),
                new StudentCreator("Dmytro", "Shevchenko", "Petrovych",
Sex.MALE, "121-21-2", "RB1242", "2004-09-30")
        };

        StudentCreator[] students4 = {
                new StudentCreator("Olena", "Moroz", "Ivanivna", Sex.FEMALE,
"125-21-1", "RB1253", "2003-01-25"),
                new StudentCreator("Igor", "Pavlov", "Oleksandrovych",
Sex.MALE, "125-21-2", "RB1254", "2003-12-12")
        };

        // Create groups
        GroupCreator group1 = new GroupCreator("AdvancedJava", groupCurator,
students1);
        GroupCreator group2 = new GroupCreator("BasicJava", groupCurator,
students2);
        GroupCreator group3 = new GroupCreator("English", groupCurator,
students3);
        GroupCreator group4 = new GroupCreator("Spanish", groupCurator,
students4);

        // Create departments
        DepartmentCreator department1 = new DepartmentCreator(
                "Computer Science",
                departmentHead,
                new GroupCreator[]{group1, group2}
        );
        DepartmentCreator department2 = new DepartmentCreator(
                "Translators",
                departmentHead,
                new GroupCreator[]{group3, group4}
        );

        // Create faculty
        FacultyCreator faculty = new FacultyCreator(
                "Faculty of Science",
                dean,
                new DepartmentCreator[]{department1, department2}
        );

        // Create university
        UniversityCreator university = new UniversityCreator(
                "National University",
                rector,
                new FacultyCreator[]{faculty}
        );

        return university.create();
    }
}
```

Файл json – test_university.json.


```
{
  "faculties": [
    {
      "departments": [
        {
          "groups": [
            {
              "students": [
                {
                  "studentId": "123-21-1",
                  "recordBookNumber": "RB1231",
                  "birthDate": "2000-05-15",
                  "firstName": "Sasha",
                  "lastName": "TikTok",
                  "patronymic": "Olegovna",
                  "sex": "FEMALE"
                },
                {
                  "studentId": "123-21-2",
                  "recordBookNumber": "RB1232",
                  "birthDate": "1999-08-22",
                  "firstName": "Misha",
                  "lastName": "Bulkin",
                  "patronymic": "Ilonovich",
                  "sex": "MALE"
                }
              ],
              "name": "AdvancedJava",
              "head": {
                "firstName": "Oleksand",
                "lastName": "Mineev",
                "patronymic": "Sergiyovich",
                "sex": "MALE"
              }
            },
            {
              "students": [
                {
                  "studentId": "122-22-4",
                  "recordBookNumber": "RB1224",
                  "birthDate": "2001-03-10",
                  "firstName": "Justin",
                  "lastName": "Biber",
                  "patronymic": "Grigirovich",
                  "sex": "MALE"
                },
                {
                  "studentId": "121-24-2",
                  "recordBookNumber": "RB1212",
                  "birthDate": "2000-11-05",
                  "firstName": "Max",
                  "lastName": "Pain",
                  "patronymic": "MacCoffe",
                  "sex": "MALE"
                }
              ],
              "name": "BasicJava",
              "head": {
                "firstName": "Oleksand",
                "lastName": "Mineev",
```

```
            "patronymic": "Sergiyovich",
            "sex": "MALE"
          }
        }
      ],
      "name": "Computer Science",
      "head": {
        "firstName": "Sarah",
        "lastName": "Davis",
        "patronymic": "Jane",
        "sex": "FEMALE"
      }
    },
    {
      "groups": [
        {
          "students": [
            {
              "studentId": "124-23-1",
              "recordBookNumber": "RB1241",
              "birthDate": "2002-07-19",
              "firstName": "Anna",
              "lastName": "Kovalenko",
              "patronymic": "Viktorivna",
              "sex": "FEMALE"
            },
            {
              "studentId": "124-23-2",
              "recordBookNumber": "RB1242",
              "birthDate": "2001-09-30",
              "firstName": "Dmytro",
              "lastName": "Shevchenko",
              "patronymic": "Petrovych",
              "sex": "MALE"
            }
          ],
          "name": "English",
          "head": {
            "firstName": "Oleksand",
            "lastName": "Mineev",
            "patronymic": "Sergiyovich",
            "sex": "MALE"
          }
        },
        {
          "students": [
            {
              "studentId": "125-24-3",
              "recordBookNumber": "RB1253",
              "birthDate": "2000-01-25",
              "firstName": "Olena",
              "lastName": "Moroz",
              "patronymic": "Ivanivna",
              "sex": "FEMALE"
            },
            {
              "studentId": "125-24-4",
              "recordBookNumber": "RB1254",
              "birthDate": "1999-12-12",
              "firstName": "Igor",
              "lastName": "Pavlov",
              "patronymic": "Oleksandrovych",
              "sex": "MALE"
            }
```

```
              ],
              "name": "Spanish",
              "head": {
                "firstName": "Oleksand",
                "lastName": "Mineev",
                "patronymic": "Sergiyovich",
                "sex": "MALE"
              }
            }
          ],
          "name": "Translators",
          "head": {
            "firstName": "Sarah",
            "lastName": "Davis",
            "patronymic": "Jane",
            "sex": "FEMALE"
          }
        }
      ],
      "name": "Faculty of Science",
      "head": {
        "firstName": "Michael",
        "lastName": "Johnson",
        "patronymic": "Robert",
        "sex": "MALE"
      }
    }
  ],
  "name": "National University",
  "head": {
    "firstName": "John",
    "lastName": "Smith",
    "patronymic": "William",
    "sex": "MALE"
  }
}
```