Міністерство освіти і науки України

Національний технічний університет «Дніпровська політехніка»

Факультет інформаційних технологій

Кафедра безпеки інформації та телекомунікацій



**Лабораторна робота**

з дисципліни «Програмування в середовищі Java»

**Виконав:** ст. гр. 125-20-2

Корнєв. А. Д.

**Перевірив:**

Мінєєв О. С.

Дніпро

НТУ «ДП»

2024

Лабоарторна робота 0

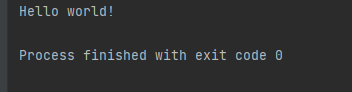
Hello World

Встановити Intelliji Idea та Java jdk останньої версії. Cтворити maven проект та розробити в цьому проекті типову програму Hello world. Програма повинна видавати на екран напис Hello world та закінчувати свою роботу.

Код:

public class lab0 {  
 public static void main(String[] args) {  
 System.*out*.println("Hello world!");  
 }  
}

Результат виконання:



Лабораторна робота 1

CVS. GIT

Під'єднати до intellij Idea систему CVS. А саме GIT. Створити аккаунт в хмарному середовищі github, під'єднати свій проект в intellij Idea до свого аккаунту github та завантажити нульову лабораторну роботу на github аккаунт. Кожну нову лабораторну роботу робити в окремії гілці(з іменем лабораторної наприклад «LR\_3») а потім після того як її написали мержити гілку до мастера.

Результат виконання:

https://github.com/AndriyKornev/java.git

Лабораторна робота 2

Основи

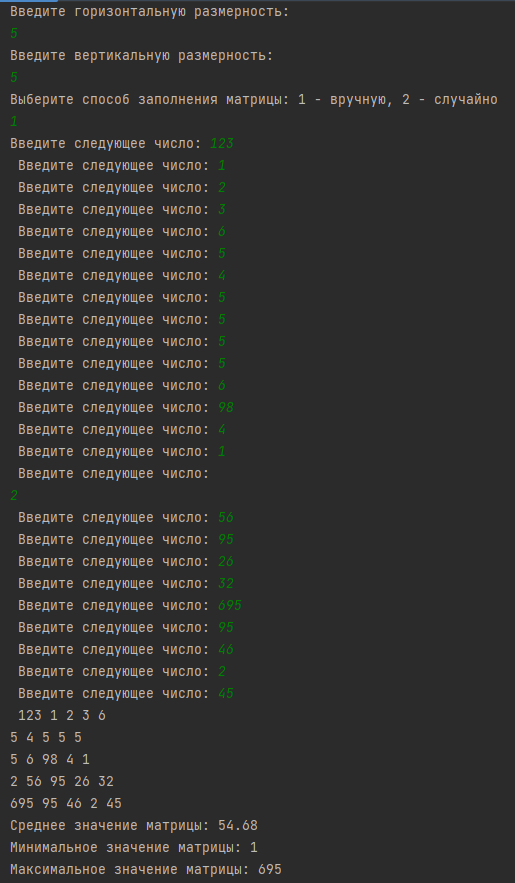
Розробити програму, що дозволить вам створити, як з клавіатури так і рандомно матрицю цілих чисел типу int заданої ширини та висоти(ввести з клавіатури), але не більше 20 на 20. Створити можливість пошуку в цій матриці мінімального і максимального елементу та розрахунок середнього арифметичного. Програма може бути написана в одному класі, обов'язково розбиття на методи. Обов'язкове використання клавіатури, під час вибору ручного чи рандомного створення матриці. Створення системи зчитування з клавіатури зробити будь-яким способом, наприклад завдяки класу Scanner. Scanner являє собою найпростішу систему сканування клавіатури. Диапазон рандомних чисел для створення елементів матриці повинен зверігатись в спеціальних константах.

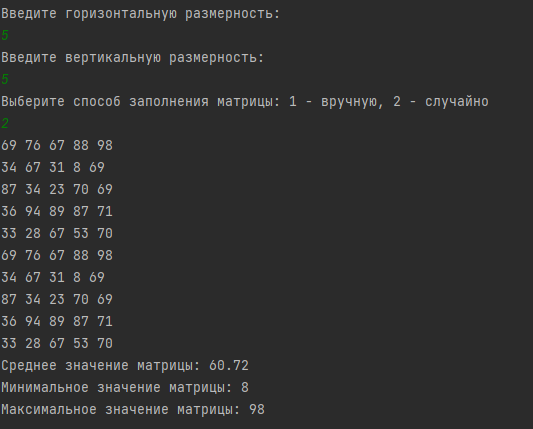
Як завдання підвищеної складності додати розрахунок середнього геометричного елементів матриці.

Код:

import java.util.Scanner;  
  
public class lab2 {  
 public static final int *MAX\_DIMENSION* = 20;  
 public static final int *MIN\_DIMENSION* = 1;  
 public static final int *MAX\_VALUE* = 100;  
 public static final int *MIN\_VALUE* = 5;  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введите горизонтальную размерность: ");  
 int x = *inputMatrixDimension*(scanner);  
 System.*out*.println("Введите вертикальную размерность: ");  
 int y = *inputMatrixDimension*(scanner);  
 System.*out*.println("Выберите способ заполнения матрицы: 1 - вручную, 2 - случайно");  
 int fillMethod = scanner.nextInt();  
 if (fillMethod == 1) {  
 *manualFill*(x, y, scanner);  
 } else if (fillMethod == 2) {  
 *randomFill*(x, y, scanner);  
 }  
 }  
  
 public static int inputMatrixDimension(Scanner scanner) {  
 int dimension = scanner.nextInt();  
 while (dimension > *MAX\_DIMENSION* || dimension < *MIN\_DIMENSION*) {  
 System.*out*.println("Размер не в диапазоне " + *MIN\_DIMENSION* + " - " + *MAX\_DIMENSION* + ". Пожалуйста, введите другое число:");  
 dimension = scanner.nextInt();  
 }  
 return dimension;  
 }  
  
 public static int[][] manualFill(int x, int y, Scanner scanner) {  
 int[][] matrix = new int[x][y];  
 for (int i = 0; i < x; i++) {  
 for (int j = 0; j < y; j++) {  
 System.*out*.print("Введите следующее число: ");  
 matrix[i][j] = scanner.nextInt();  
 System.*out*.print(" ");  
 }  
 }  
 *printMatrix*(matrix);  
 return matrix;  
 }  
  
 public static int[][] randomFill(int x, int y, Scanner scanner) {  
 int[][] matrix = new int[x][y];  
 for (int i = 0; i < x; i++) {  
 for (int j = 0; j < y; j++) {  
 matrix[i][j] = (int) Math.*round*(Math.*random*() \* (*MAX\_VALUE* - *MIN\_VALUE*)) + *MIN\_VALUE*;  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 *printMatrix*(matrix);  
 return matrix;  
 }  
  
 private static void printMatrix(int[][] matrix) {  
 for (int[] row : matrix) {  
 for (int num : row) {  
 System.*out*.print(num + " ");  
 }  
 System.*out*.println();  
 }  
 *calculateStatistics*(matrix);  
 }  
  
 private static void calculateStatistics(int[][] matrix) {  
 int sum = 0;  
 int min = Integer.*MAX\_VALUE*;  
 int max = Integer.*MIN\_VALUE*;  
 for (int[] row : matrix) {  
 for (int num : row) {  
 sum += num;  
 if (num < min) min = num;  
 if (num > max) max = num;  
 }  
 }  
 double average = (double) sum / (matrix.length \* matrix[0].length);  
 System.*out*.println("Среднее значение матрицы: " + average);  
 System.*out*.println("Минимальное значение матрицы: " + min);  
 System.*out*.println("Максимальное значение матрицы: " + max);  
 }  
}

Результат виконання:





Лабораторна робота 3

ООП

Створити програму що буде створювати та обробляти комплексний об'єкт під назвою університет(university). Програма повинна складатися з трьох частин: модель вид та контролер згідно з парадигмою mvc (Model View Controller).  Кожній з цих груп повинна відповідати package з  відповідною назвою.  В моделі повинні знаходитись усі класи що відповідають за структурні підрозділи університету. Серед них:   університет, факультет, кафедра, група, студент, людина (Human). Усі вони повинні містити  назву  типу string та голову типу Human. Студент також повинен бути породжений від  Human.  Human повинен мати поля ім'я, прізвище, побатькові та стать.  Усі поля повинні бути строковими окрім поля стать. Стать повинна використовувати спеціальний  enum  типу Sex(стать).

В цій лабораторній роботі  група View Нам не потрібна.

 Що стосується групи контроллер (controller)  то вона повинна містити менеджери що дозволяють нам створити відповідні підрозділи наприклад  StudentCreator, FacultyCreator, GroupCreator  та інші, кожен з яких повинен використовувати можливості нижчого за рівнем   створювача. Програма повинна також містити клас Run, в якому буде знаходитись точка входу та методи, що повинні дати можливість створити університет.   Процес створення університету повинен бути зроблений в методі createTypycalUniversity.

В програмі активно рекомендується використовувати абстрактні класи та інтерфейси.

Код:

Model Package

Sex model

package modelLab3;  
  
public enum Sex {  
 *MALE*,  
 *FEMALE*,  
}

Human model

package modelLab3;  
  
import java.util.Objects;  
  
public class Human {  
  
 protected String firstName;  
 protected String lastName;  
 protected String middleName;  
 protected Sex gender;  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public void setMiddleName(String middleName) {  
 this.middleName = middleName;  
 }  
  
 public Sex getGender() {  
 return gender;  
 }  
  
 public void setGender(Sex gender) {  
 this.gender = gender;  
 }  
  
 @Override  
 public String toString() {  
 return "Person{" +  
 "firstName='" + firstName + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", middleName='" + middleName + '\'' +  
 ", gender=" + gender +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Human)) return false;  
 Human person = (Human) o;  
 return Objects.*equals*(firstName, person.firstName) &&  
 Objects.*equals*(lastName, person.lastName) &&  
 Objects.*equals*(middleName, person.middleName) &&  
 gender == person.gender;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName, middleName, gender);  
 }  
}

Group model

package modelLab3;  
  
import java.util.List;  
import java.util.Objects;  
  
public class Group {  
 private int maxCapacity;  
 List<Student> students;  
  
 public int getMaxCapacity() {  
 return maxCapacity;  
 }  
  
 public void setMaxCapacity(int maxCapacity) {  
 this.maxCapacity = maxCapacity;  
 }  
  
 public List<Student> getStudents() {  
 return students;  
 }  
  
 public void setStudents(List<Student> students) {  
 this.students = students;  
 }  
  
 @Override  
 public String toString() {  
 return "Group{" +  
 "maxCapacity=" + maxCapacity +  
 ", students=" + students.toString() +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Group)) return false;  
 Group group = (Group) o;  
 return maxCapacity == group.maxCapacity &&  
 students.equals(group.students);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(maxCapacity, students);  
 }  
}

Faculty model

package modelLab3;  
  
import java.util.List;  
import java.util.Objects;  
  
public class Faculty extends Object {  
 private Human dean;  
 private int maxDepartments;  
 private List<Department> departments;  
  
 public Human getDean() {  
 return dean;  
 }  
  
 public void setDean(Human dean) {  
 this.dean = dean;  
 }  
  
 public int getMaxDepartments() {  
 return maxDepartments;  
 }  
  
 public void setMaxDepartments(int maxDepartments) {  
 this.maxDepartments = maxDepartments;  
 }  
  
 public List<Department> getDepartments() {  
 return departments;  
 }  
  
 public void setDepartments(List<Department> departments) {  
 this.departments = departments;  
 }  
  
 @Override  
 public String toString() {  
 return "Faculty{" +  
 "dean=" + dean +  
 ", maxDepartments=" + maxDepartments +  
 ", departments=" + departments +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Faculty)) return false;  
 Faculty faculty = (Faculty) o;  
 return maxDepartments == faculty.maxDepartments &&  
 dean.equals(faculty.dean) &&  
 departments.equals(faculty.departments);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(dean, maxDepartments, departments);  
 }  
}

Department model

package modelLab3;  
  
import java.util.List;  
import java.util.Objects;  
  
public class Department {  
 private int maxGroups;  
 private List<Group> groups;  
  
 public int getMaxGroups() {  
 return maxGroups;  
 }  
  
 public void setMaxGroups(int maxGroups) {  
 this.maxGroups = maxGroups;  
 }  
  
 public List<Group> getGroups() {  
 return groups;  
 }  
  
 public void setGroups(List<Group> groups) {  
 this.groups = groups;  
 }  
  
 @Override  
 public String toString() {  
 return "Department{" +  
 "maxGroups=" + maxGroups +  
 ", groups=" + groups +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Department)) return false;  
 Department that = (Department) o;  
 return maxGroups == that.maxGroups &&  
 groups.equals(that.groups);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(maxGroups, groups);  
 }  
}

University model

package modelLab3;  
  
import java.util.List;  
import java.util.Objects;  
  
public class University {  
 private String name;  
 private Human president;  
 private List<Faculty> faculties;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Human getPresident() {  
 return president;  
 }  
  
 public void setPresident(Human president) {  
 this.president = president;  
 }  
  
 public List<Faculty> getFaculties() {  
 return faculties;  
 }  
  
 public void setFaculties(List<Faculty> faculties) {  
 this.faculties = faculties;  
 }  
  
 @Override  
 public String toString() {  
 return "University{" +  
 "name='" + name + '\'' +  
 ", president=" + president +  
 ", faculties=" + faculties +  
 '}';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof University)) return false;  
 University that = (University) o;  
 return Objects.*equals*(name, that.name) &&  
 Objects.*equals*(president, that.president) &&  
 Objects.*equals*(faculties, that.faculties);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, president, faculties);  
 }  
}

Controller Package

Human Controller

package controllerLab3;  
  
import modelLab3.Human;  
import modelLab3.Sex;  
  
public class HumanCreator {  
 public Human createPerson(String firstName, String lastName, String middleName, Sex gender) {  
 Human person = new Human();  
 person.setFirstName(firstName);  
 person.setLastName(lastName);  
 person.setMiddleName(middleName);  
 person.setGender(gender);  
 return person;  
 }  
 public Human createTypicalPerson(){  
 return createPerson("Roger", "Gol", "D", Sex.*MALE*);  
 }  
}

Group Controller

package controllerLab3;  
  
import modelLab3.Group;  
import modelLab3.Student;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class GroupCreator {  
  
  
 Group createGroup(int studentCount, int maxCapacity) {  
 Group group = new Group();  
 StudentCreator studentCreator = new StudentCreator();  
 List<Student> students = new ArrayList<>();  
 for (int i = 0; i < studentCount; i++) {  
 students.add(studentCreator.createTypicalStudent());  
 }  
 group.setStudents(students);  
 group.setMaxCapacity(maxCapacity);  
 return group;  
 }  
  
 Group createTypicalGroup() {  
 Group group = createGroup(3, 10);  
 return group;  
 }  
  
}

Faculty Controller

package controllerLab3;  
import modelLab3.Department;  
import modelLab3.Faculty;  
import modelLab3.Human;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class FacultyCreator {  
  
 Faculty createFaculty(int departmentCount, int maxDepartments, Human dean) {  
 Faculty faculty = new Faculty();  
 DepartmentCreator departmentCreator = new DepartmentCreator();  
 List<Department> departments = new ArrayList<>();  
 for (int i = 0; i < departmentCount; i++) {  
 departments.add(departmentCreator.createTypicalDepartment());  
 }  
  
 faculty.setDepartments(departments);  
 faculty.setMaxDepartments(maxDepartments);  
 faculty.setDean(dean);  
 return faculty;  
 }  
  
 Faculty createTypicalFaculty() {  
 HumanCreator humanCreator = new HumanCreator();  
 Faculty faculty = createFaculty(2, 3, humanCreator.createTypicalPerson());  
 return faculty;  
 }  
}

Student Controller

package controllerLab3;  
  
  
import modelLab3.Human;  
import modelLab3.Student;  
  
public class StudentCreator {  
 int ID = 0;  
  
 public Student createStudent(Human person, int studentId) {  
  
  
 Student student = new Student();  
 student.setFirstName(person.getFirstName());  
 student.setLastName(person.getLastName());  
 student.setMiddleName(person.getMiddleName());  
  
 student.setStudentId(studentId);  
 student.setGender(person.getGender());  
  
 return student;  
 }  
  
 public Student createTypicalStudent() {  
 HumanCreator humanCreator = new HumanCreator();  
 Human typicalPerson = humanCreator.createTypicalPerson();  
 ID = ID + 1;  
 Student typicalStudent = createStudent(typicalPerson, ID);  
 return typicalStudent;  
 }  
  
}

Department Controller

package controllerLab3;  
  
import modelLab3.Group;  
import modelLab3.Department;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class DepartmentCreator {  
  
  
 public Department createDepartment(int groupCount, int maxGroups) {  
 Department department = new Department();  
 GroupCreator groupCreator = new GroupCreator();  
 List<Group> groups = new ArrayList<>();  
 for (int i = 0; i < groupCount; i++) {  
 groups.add(groupCreator.createTypicalGroup());  
 }  
 department.setGroups(groups);  
 department.setMaxGroups(maxGroups);  
 return department;  
 }  
  
 public Department createTypicalDepartment() {  
 Department department = createDepartment(4, 6);  
 return department;  
 }  
}

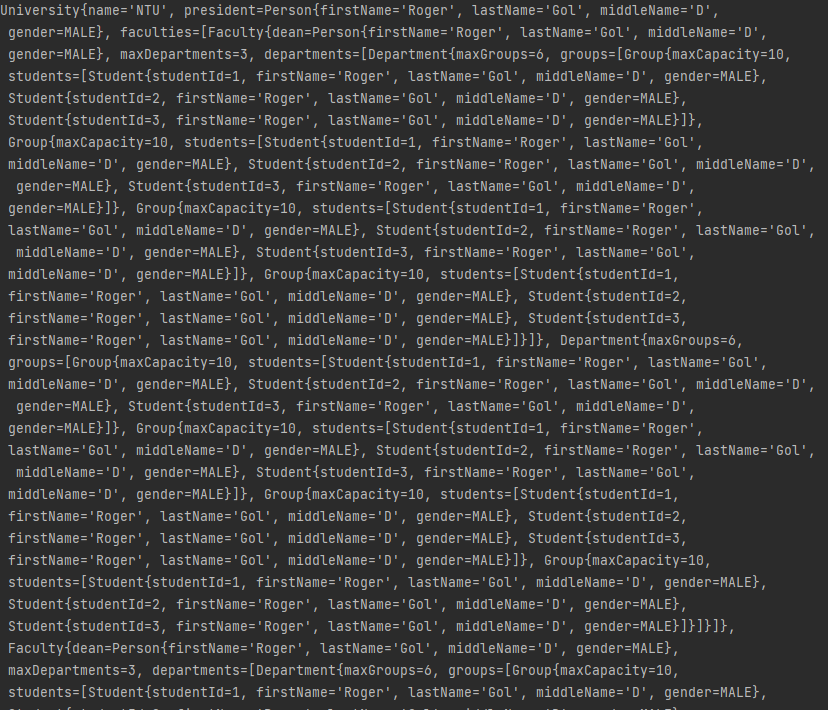
University Controller

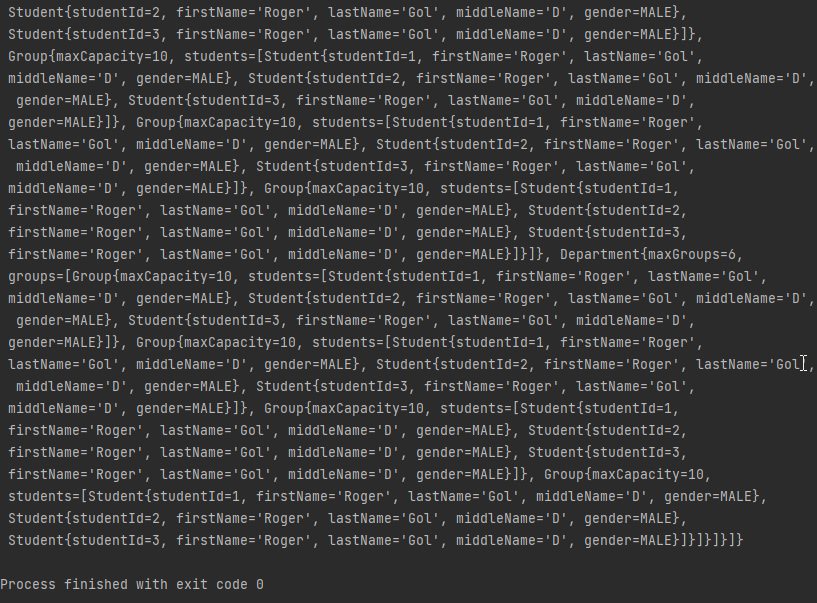
package controllerLab3;  
import modelLab3.Human;  
import modelLab3.Faculty;  
import modelLab3.University;  
import java.util.ArrayList;  
import java.util.List;  
  
public class UniversityCreator {  
  
  
 public University createUniversity(String name, int facultyCount, Human president) {  
 University university = new University();  
 FacultyCreator facultyCreator = new FacultyCreator();  
 university.setName(name);  
 university.setPresident(president);  
 List<Faculty> faculties = new ArrayList<>();  
 for (int i = 1; i < facultyCount; i++) {  
 faculties.add(facultyCreator.createTypicalFaculty());  
 }  
 university.setFaculties(faculties);  
  
 return university;  
  
 }  
  
 public University createTypicalUniversity() {  
 HumanCreator humanCreator = new HumanCreator();  
 University university = createUniversity("NTU", 3, humanCreator.createTypicalPerson());  
  
 return university;  
 }  
  
}

lab3

import modelLab3.University;  
import controllerLab3.UniversityCreator;  
  
public class lab3 {  
  
 public static void main(String[] args) {  
 UniversityCreator universityCreator = new UniversityCreator();  
 University university = universityCreator.createTypicalUniversity();  
 System.*out*.println(university.toString());  
 }  
  
}

Результат виконання:





Лабораторна робота 4

JUnit. Json

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:

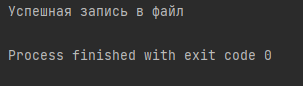
Json Manager

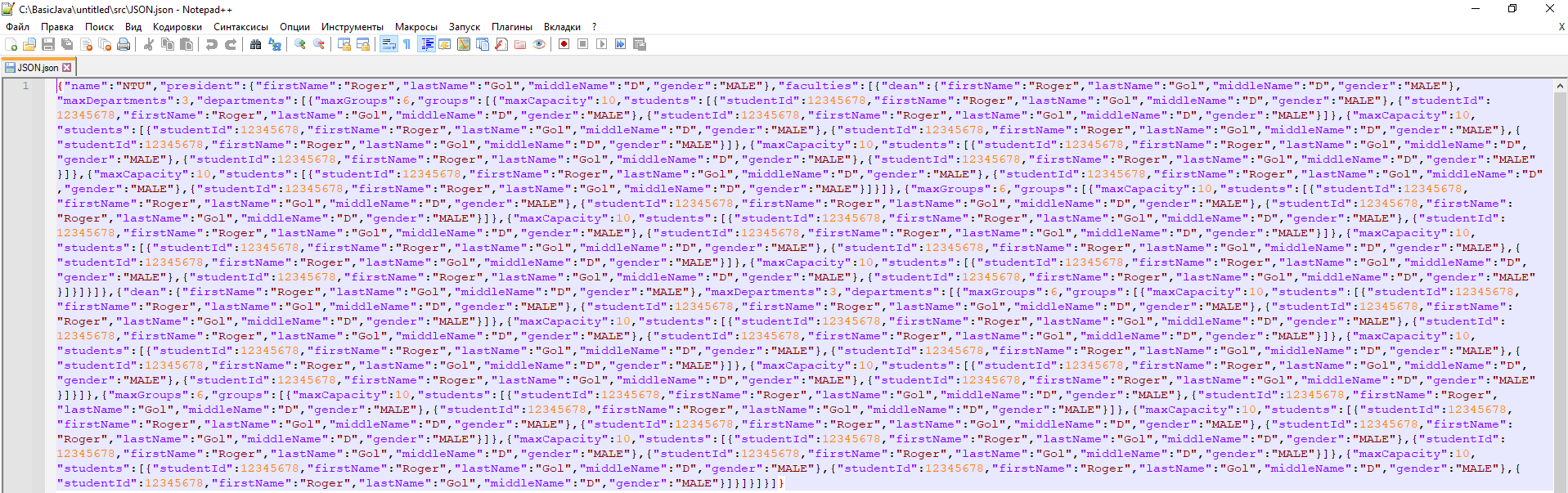
package lab4;  
  
import com.google.gson.Gson;  
import modelLab4.University;  
  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class JsonManager {  
  
 public void writeToJson(Object object, String filePath) {  
  
 Gson gson = new Gson();  
  
 String jsonFile = gson.toJson(object);  
  
 try {  
 FileWriter fileWriter = new FileWriter(filePath);  
  
 fileWriter.write(jsonFile);  
 fileWriter.close();  
  
 System.*out*.println("Успешная запись в файл");  
  
 } catch (IOException e) {  
  
 System.*out*.println("Ошибка чтения из файла");  
  
 e.printStackTrace();  
 }  
 }  
  
 public Object readFromJson(String filePath) {  
  
 Gson gson = new Gson();  
  
 try {  
 FileReader fileReader = new FileReader(filePath);  
  
 return gson.fromJson(fileReader, University.class);  
 } catch (FileNotFoundException e) {  
  
 System.*out*.println("Файл не найден: " + filePath);  
  
 e.printStackTrace();  
 } catch (Exception e) {  
  
 System.*out*.println("Ошибка чтения JSON из файла: " + filePath);  
  
 e.printStackTrace();  
 }  
 return null;  
 }  
}

Json Test

package lab4;  
  
import controllerLab4.UniversityCreator;  
import modelLab4.University;  
  
import org.junit.Test;  
import org.junit.Assert;  
  
public class JsonTest {  
 @Test  
 public void testFileWriteJsonToFile() {  
  
 String filePath = "./src/JSON.json";  
  
 JsonManager jsonManager = new JsonManager();  
 UniversityCreator universityCreator = new UniversityCreator();  
  
 University originalUniversity = universityCreator.createTypicalUniversity();  
  
 jsonManager.writeToJson(originalUniversity, filePath);  
 University readUniversity = (University) jsonManager.readFromJson(filePath);  
  
 Assert.*assertEquals*(readUniversity, originalUniversity);  
 }  
}

Результат виконання:





Лабораторна робота 5

Jdbc

**Завдання**: Створити базу даних в будь якому сервері баз даних. Створити таблицю з переліком студентів вказати їх прізвище, ім'я, по батькові, день народження номер залікової книжки та ID.

Створити програму що буде дозволяти виводити на екран інформацію про студентів які народилися в тому чи іншому місяці року. Програма повинна завдяки системі jdbc під'єднатися до вашої бази даних та робити до неї запроси. Вимог до розробки бази даних немає. Програма ж має бути написана за усіма стандартами ООП. Та може бути спроектована за двох принципів:

- при будь-якій ситуації буде забиратися весь перелік студентів, а вже на стороні java буде зроблено пошук необхідного

- SQL запрос буде сформований згідно запросу який зробив користувач і вже сервер управління баз даних буде вирішувати, які самі студенти народилися в тому чи іншому місяці.

У висновку обов'язково пояснити чому вибрали той чи інший принцип, які в нього переваги та недоліки. Оцінка не залежить від того який сервер управління баз даних вибрали. Перелік студентів зробити не менше 20 людей. Місяць червень зробити місяцем, коли в жодного зі студентів немає дня народження.

SQL код створення бази даних розмістити проекті 5 лабораторної роботи в файлі database в пекеджі resources. Для використання цієї лабораторної роботи рекомендується активно використовувати знання отримані на дисципліні що стосуються розробки баз даних.

До паперового звіту обов'язково додати принтскрин з програми в якій ви дивитесь інформацію вашого сервера управління баз даних, де показати створену таблицю, її ім'я та загальні відомості бази даних, наприклад назва, ім'я, назва користувача адміністратора, пароль тощо. Для роботи з сервером управління баз даних рекомендуємо використовувати програмне забезпечення компанії jetbrains datagrip. Або вбудовану панель користування базами даних, що міститься у середовищі intellij Idea, яка на сьогоднішній день підтримує майже всі сервери управління баз даних.

Код:

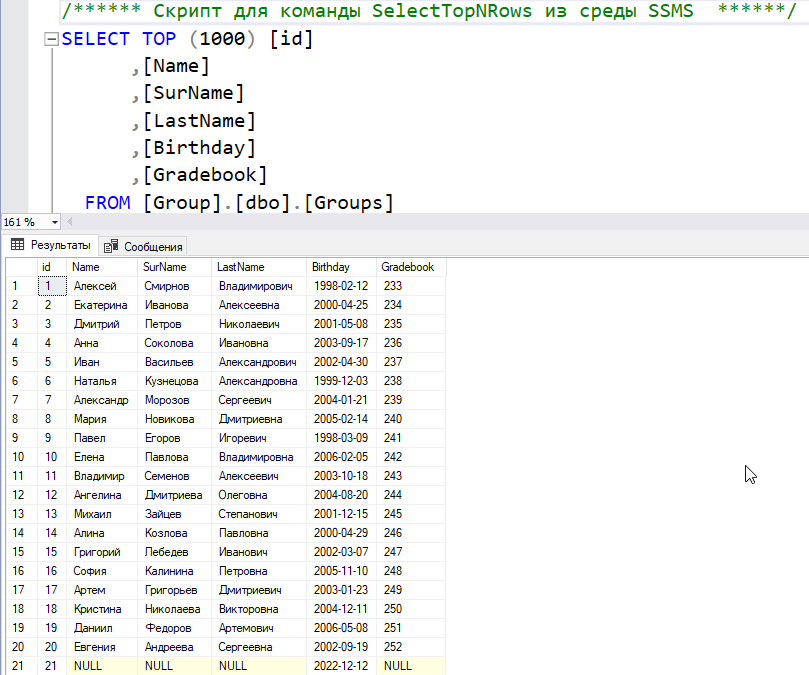
exequteRequest

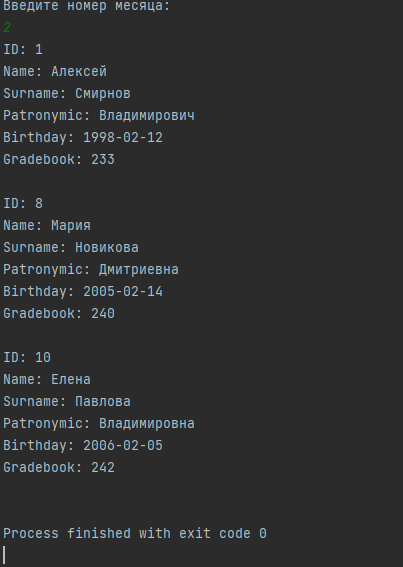
package RequestPackage;  
  
import java.sql.\*;  
  
public class executeRequest {  
 public static void executeRequest(int month) {  
 Connection connection = null;  
 Statement statement = null;  
 ResultSet resultSet = null;  
  
 try {  
 connection = DriverManager.*getConnection*("jdbc:sqlserver://localhost;database=Group;encrypt=false;trustServerCertificate=true", "Andriy", "qwe123321");  
  
 statement = connection.createStatement();  
  
 String sqlQuery = "SELECT \* FROM Groups WHERE MONTH(Birthday) = " + month;  
  
 resultSet = statement.executeQuery(sqlQuery);  
  
 while (resultSet.next()) {  
 int id = resultSet.getInt("ID");  
 String name = resultSet.getString("Name");  
 String surname = resultSet.getString("SurName");  
 String lastName = resultSet.getString("LastName");  
 Date birthday = resultSet.getDate("Birthday");  
 String gradebook = resultSet.getString("Gradebook");  
  
 System.*out*.println("ID: " + id);  
 System.*out*.println("Name: " + name);  
 System.*out*.println("Surname: " + surname);  
 System.*out*.println("Patronymic: " + lastName);  
 System.*out*.println("Birthday: " + birthday);  
 System.*out*.println("Gradebook: " + gradebook);  
 System.*out*.println();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (resultSet != null) resultSet.close();  
 if (statement != null) statement.close();  
 if (connection != null) connection.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
}

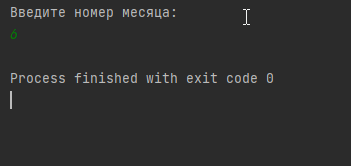
lab5

import java.util.Scanner;  
import RequestPackage.executeRequest;  
  
public class lab5 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Введите номер месяца: ");  
 int month = scanner.nextInt();  
  
 executeRequest.*executeRequest*(month);  
  
 scanner.close();  
 }  
}

Результат виконання:







Другий підхід, де SQL-запит формується на сервері управління баз даних, є кращим з кількох причин:

Ефективність: Виконання пошуку в базі даних є ефективнішим, оскільки база даних може використовувати оптимізовані алгоритми та індекси для швидкого виконання запиту. У порівнянні з обробкою пошуку на бікенді (Java), де можуть виникати затримки через велику кількість даних або неоптимальні алгоритми пошуку, такий підхід дає кращу продуктивність.

Масштабованість: Виконання запиту на рівні бази даних дозволяє легко масштабувати систему. База даних може бути налаштована для роботи з великою кількістю даних та великими навантаженнями, що забезпечує стійку та надійну роботу системи навіть при збільшенні обсягу даних та користувацьких запитів.

Безпека: Виконання SQL-запиту на рівні бази даних дозволяє забезпечити безпеку даних шляхом використання механізмів автентифікації, авторизації та контролю доступу. Це дозволяє захистити дані від несанкціонованого доступу та змін

Простота розробки та підтримки: Розміщення логіки обробки даних на рівні бази даних може спростити розробку та підтримку системи, оскільки це дозволяє розділити логіку додатку на частину, що відповідає за роботу з даними, та частину, що відповідає за бізнес-логіку.

Отже, використання SQL-запитів на рівні бази даних надає кращу продуктивність, масштабованість, безпеку та простоту розробки та підтримки порівняно з використанням пошуку даних на стороні додатку.