Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Звіт з виконання лабораторних робіт

з дисципліни: «Базовий курс Java»

Виконав:

Студент гр. 125-20-2

Зубков Михайло Юрійович

Перевірив:

Мінєєв О.С.

Дніпро

2024

Лабораторна робота номер 0. Hello world

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

Хід роботи

З офіційного сайту JetBrains було завантажено IDE «IntelliJ IDEA Community Edition 2023.3.3». Створено новий Maven-проект.

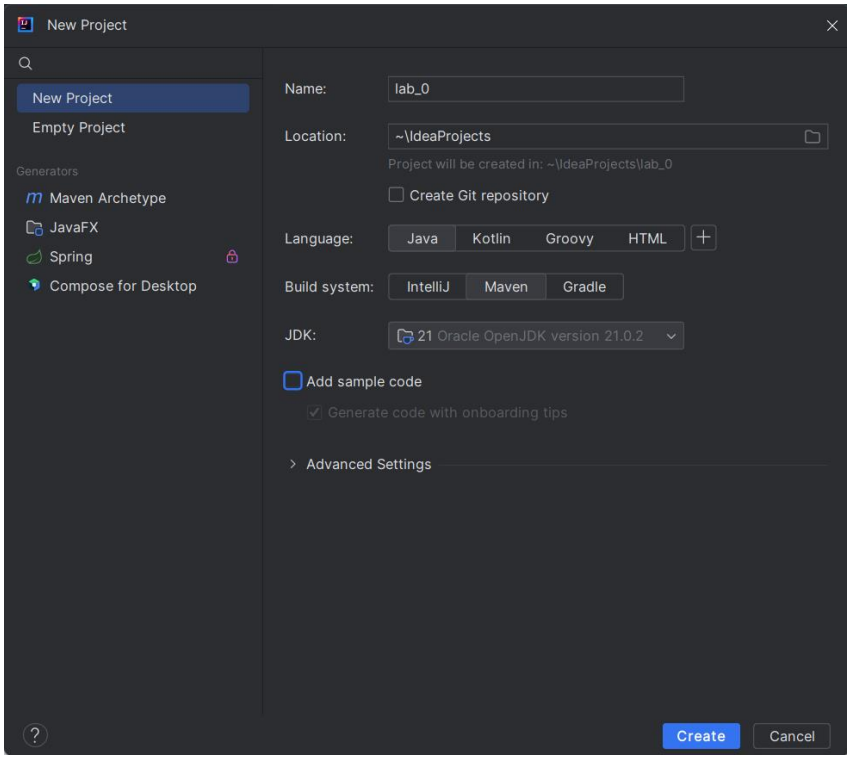
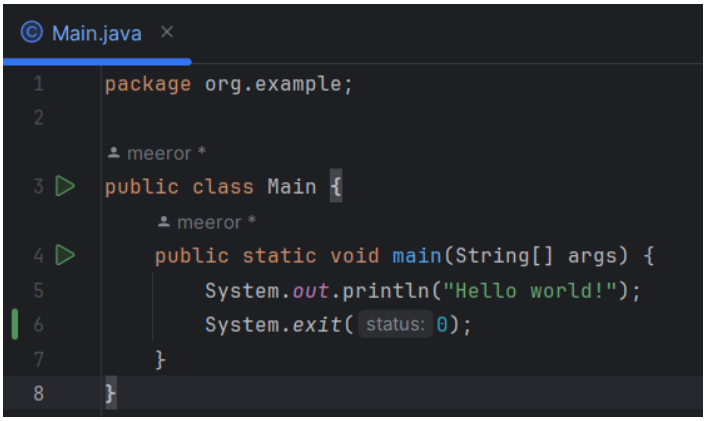


Рис.1 – створення проекту в IntelliJ IDEA

У новому проекті написано програму Hello, World!



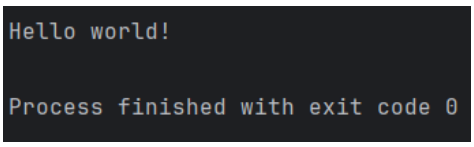


Рис. 2, 3 – програма Hello, World!

Код програми

```java
package org.example;

public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
        System.exit(0);
    }
}
```

Лабораторна робота номер 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.

Обов'язково перевірте, щоб ваш проект лабораторних робіт на GitHub мав вільний доступ усіх бажаючих до вашого коду. Обмеження доступу до ваших лабораторних робіт буде розцінено, як помилка при виконанні лабораторної роботи

Хід роботи

Виконано вхід до облікового запису сервісу Github. На сайті під'єднано віддалений репозиторій до папки на ПК зі всіма лабораторними. Створено гілку **LR_1** та зафіксовано до неї зміни у папці з 0 лабораторною роботою.

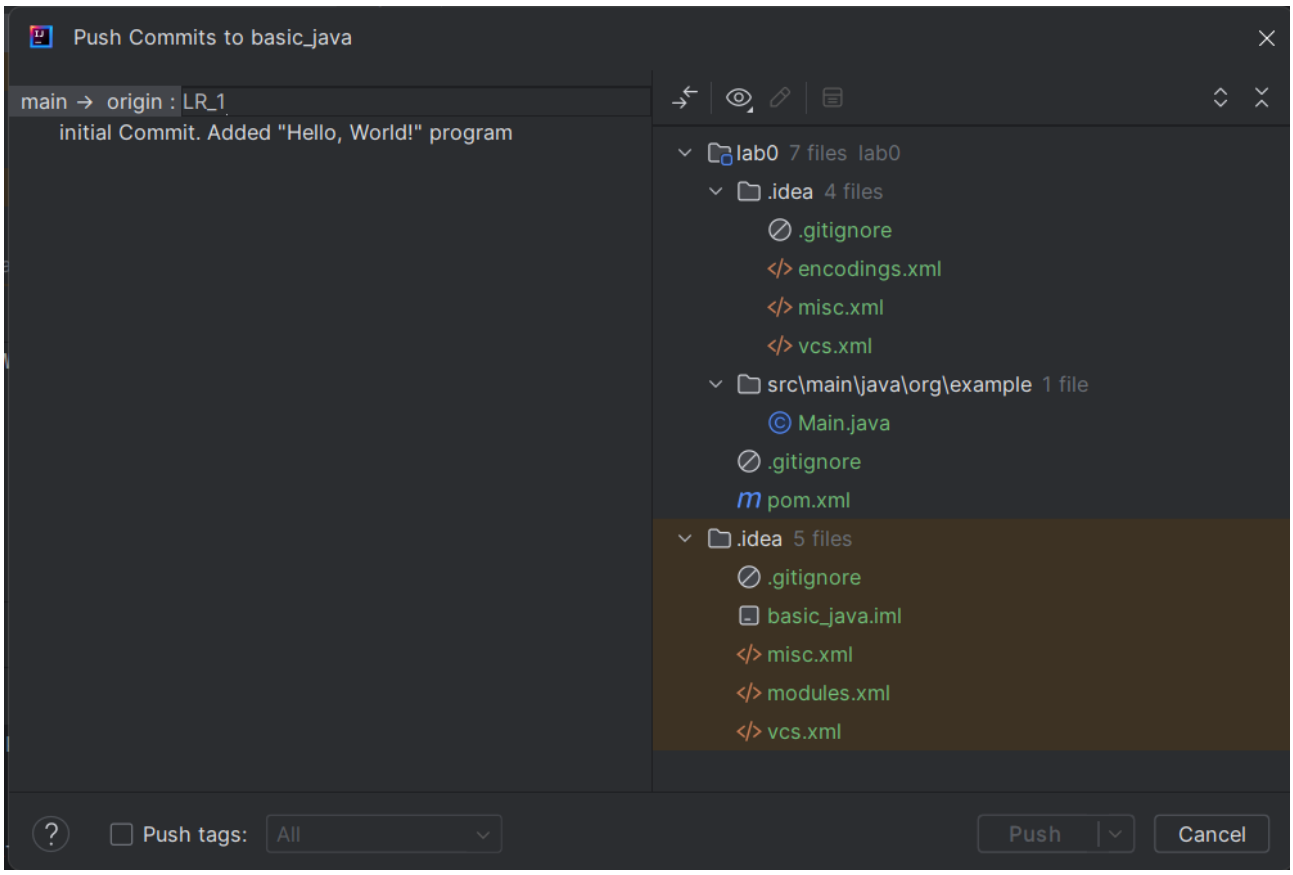


Рис. 4 – надсилання змін на хмарний сервіс Github.

Перейшовши до сервісу, можна побачити дві гілки – **main** та **LR_1**.

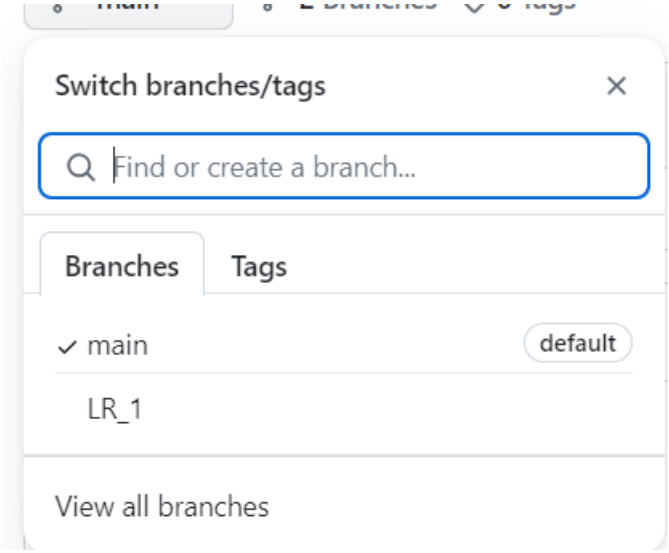


Рис.5 - гілки проекту

Після підтвердження пул реквесту можна спокійно замерджити гілки.

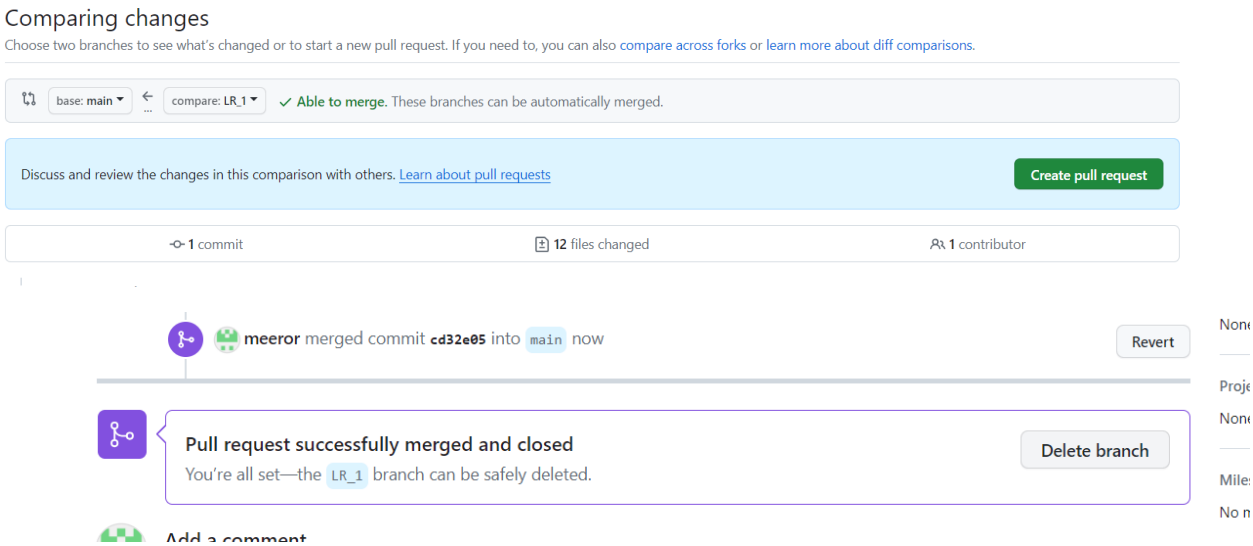


Рис. 6, 7 – створення та підтвердження pool request

Лабораторна робота номер 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.

Хід роботи

Методи для роботи з матрицями винесено в файл MatrixOperations.java. У файлі main проводиться виклик програми.

Рис 8. – main

Код програми

```
package org.example;

public class Main {
    public static void main(String[] args) {
        MatrixOperations matrixOperations = new MatrixOperations();
        matrixOperations.run();
    }
}
```

Програма по роботі з матрицею починається з імпортування бібліотек по роботі зі вводом з клавіатури та створенню випадкових чисел. Після цього оголошуються глобальні константи з максимальним розміром матриці та границі допустимих випадкових чисел.

```
1   package org.example;
2   import java.util.Scanner;
3   import java.util.Random;
4
    2 usages   meeror *
5   public class MatrixOperations {
6       // Максимальний розмір матриці
        4 usages
7       private static final int maxSize = 20;
8       // Діапазон рандомних чисел
        2 usages
9       private static final int randomMin = 1;
        1 usage
10      private static final int randomMax = 100;
11
```

Рис. 9 – імпортування бібліотек та var

У методі main викликаються усі інші методи, що потрібні для введення та обчислення шуканих даних.

```
18      public void run() {
19          readMatrixSize();
20          initializeMatrix();
21          printMatrix();
22          findMinMax();
23          calculateArithmeticAverage();
24          calculateGeometricAverage();
25      }
26
```

Рис. 10 – виклик методів

Метод readMatrixSize реалізує введення розміру матриці та обробку помилки у разі введення завеликого розміру.

```
27      public void readMatrixSize() {
28          Scanner scanner = new Scanner(System.in);
29          System.out.println("Введіть ширину матриці (не більше " + maxSize + "):");
30          width = scanner.nextInt();
31          System.out.println("Введіть висоту матриці (не більше " + maxSize + "):");
32          height = scanner.nextInt();
33          // Перевірка на максимальний розмір
34          if (width > maxSize || height > maxSize) {
35              System.out.println("Розмір матриці перевищує максимально допустимий розмір.");
36              System.exit( status: 1);
37          }
38      }
39
```

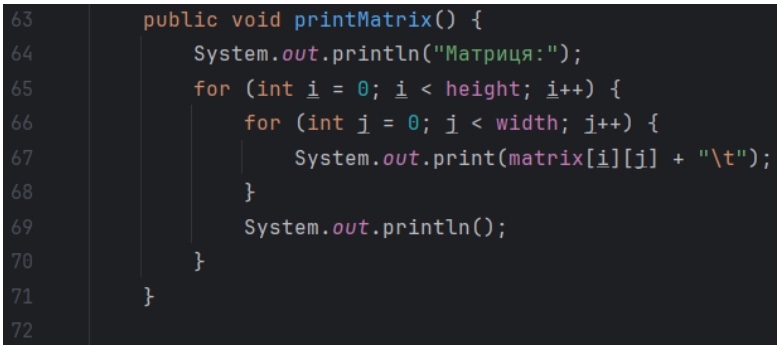
Рис. 11 – метод введення розмірів матриці

Метод initializeMatrix потрібен для введення значень елементів матриці. Варто зазначити, що у користувача є можливість згенерувати випадкові значення.

```
40      public void initializeMatrix() {
41          matrix = new int[height][width];
42          Scanner scanner = new Scanner(System.in);
43          Random random = new Random();
44          System.out.println("Бажаєте ввести матрицю вручну? (Y/N)");
45          String choice = scanner.nextLine();
46          if (choice.equalsIgnoreCase( anotherString: "Y")) {
47              System.out.println("Введіть елементи матриці:");
48              for (int i = 0; i < height; i++) {
49                  for (int j = 0; j < width; j++) {
50                      matrix[i][j] = scanner.nextInt();
51                  }
52              }
53          } else {
54              // Генерування рандомних чисел для матриці
55              for (int i = 0; i < height; i++) {
56                  for (int j = 0; j < width; j++) {
57                      matrix[i][j] = random.nextInt( bound: randomMax - randomMin + 1) + randomMin;
58                  }
59              }
60          }
61      }
62
```

Рис. 12 – метод введення елементів матриці

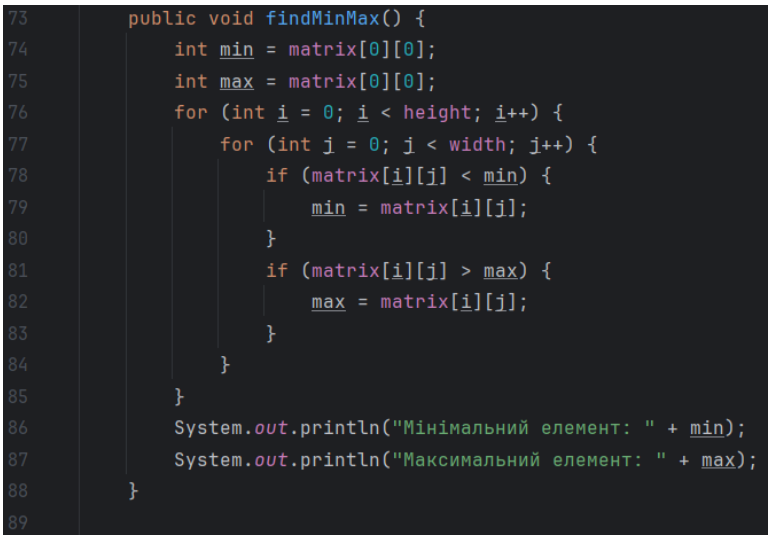
Метод printMatrix виводить построково елементи матриці.



```
63      public void printMatrix() {
64          System.out.println("Матриця:");
65          for (int i = 0; i < height; i++) {
66              for (int j = 0; j < width; j++) {
67                  System.out.print(matrix[i][j] + "\t");
68              }
69              System.out.println();
70          }
71      }
72
```

Рис. 13 – метод виведення матриці у консоль

Метод findMinMax потрібен для пошуку мінімального та максимального елементів методом порівняння усіх елементів матриці з першим елементом.



```
73      public void findMinMax() {
74          int min = matrix[0][0];
75          int max = matrix[0][0];
76          for (int i = 0; i < height; i++) {
77              for (int j = 0; j < width; j++) {
78                  if (matrix[i][j] < min) {
79                      min = matrix[i][j];
80                  }
81                  if (matrix[i][j] > max) {
82                      max = matrix[i][j];
83                  }
84              }
85          }
86          System.out.println("Мінімальний елемент: " + min);
87          System.out.println("Максимальний елемент: " + max);
88      }
89
```

Рис. 14 – метод знаходження найменшого та найбільшого елементу матриці

Методи calculateArithmeticAverage та calculateGeometricAverage розраховують значення середнього арифметичного та геометричного з елементів матриці.

```
90        public void calculateArithmeticAverage() {
91            double sum = 0;
92            for (int i = 0; i < height; i++) {
93                for (int j = 0; j < width; j++) {
94                    sum += matrix[i][j];
95                }
96            }
97            double average = sum / (width * height);
98            System.out.println("Середнє арифметичне: " + average);
99        }
100
```

```
101       public void calculateGeometricAverage() {
102           double product = 1.0;
103           for (int i = 0; i < height; i++) {
104               for (int j = 0; j < width; j++) {
105                   product *= matrix[i][j];
106               }
107           }
108           double geometricMean = Math.pow(product, 1.0 / (width * height));
109           System.out.println("Середнє геометричне: " + geometricMean);
110       }
111   }
112
```

Рис. 15,16 – метод розрахунку середнього арифметичного та геометричного з елементів матриці

Код програми

```
package org.example;
import java.util.Scanner;
import java.util.Random;

public class MatrixOperations {
    // Максимальний розмір матриці
    private static final int maxSize = 20;
    // Діапазон рандомних чисел
    private static final int randomMin = 1;
    private static final int randomMax = 100;

    private int[][] matrix;
    private int width;
    private int height;



    public void run() {
        readMatrixSize();
        initializeMatrix();
        printMatrix();
        findMinMax();
        calculateArithmeticAverage();
        calculateGeometricAverage();
    }

    public void readMatrixSize() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введіть ширину матриці (не більше " + maxSize +
"):");
        width = scanner.nextInt();
        System.out.println("Введіть висоту матриці (не більше " + maxSize +
"):");
        height = scanner.nextInt();
        // Перевірка на максимальний розмір
        if (width > maxSize || height > maxSize) {
            System.out.println("Розмір матриці перевищує максимально допустимий
розмір.");
            System.exit(1);
        }
    }

    public void initializeMatrix() {
        matrix = new int[height][width];
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
        System.out.println("Бажаєте ввести матрицю вручну? (Y/N)");
        String choice = scanner.nextLine();
        if (choice.equalsIgnoreCase("Y")) {
            System.out.println("Введіть елементи матриці:");
            for (int i = 0; i < height; i++) {
                for (int j = 0; j < width; j++) {
                    matrix[i][j] = scanner.nextInt();
                }
            }
        } else {
            // Генерування рандомних чисел для матриці
            for (int i = 0; i < height; i++) {
                for (int j = 0; j < width; j++) {
                    matrix[i][j] = random.nextInt(randomMax - randomMin + 1) +
randomMin;
                }
            }
        }
```

```
    }

    public void printMatrix() {
        System.out.println("Матриця:");
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
    }

    public void findMinMax() {
        int min = matrix[0][0];
        int max = matrix[0][0];
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                if (matrix[i][j] < min) {
                    min = matrix[i][j];
                }
                if (matrix[i][j] > max) {
                    max = matrix[i][j];
                }
            }
        }
        System.out.println("Мінімальний елемент: " + min);
        System.out.println("Максимальний елемент: " + max);
    }

    public void calculateArithmeticAverage() {
        double sum = 0;
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                sum += matrix[i][j];
            }
        }
        double average = sum / (width * height);
        System.out.println("Середнє арифметичне: " + average);
    }

    public void calculateGeometricAverage() {
        double product = 1.0;
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                product *= matrix[i][j];
            }
        }
        double geometricMean = Math.pow(product, 1.0 / (width * height));
        System.out.println("Середнє геометричне: " + geometricMean);
    }
}
```

Лабораторна робота номер 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

Хід роботи

Створено клас University.java, що являє собою комплексний об’єкт що має такі властивості як name, head, facultyList. Для реалізації підходу ООП, до програми додано get() та set() методи для кожної змінної класа.

```
public class University {

    private String name;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    private Human head;

    public Human getHead() {
        return head;
    }
    public void setHead(Human head) {
        this.head = head;
    }

    private List<Faculty> facultyList;
    public List<Faculty> getFaculties() {
        return facultyList;
    }
    public void setFaculties(List<Faculty> listFaculties) {
        this.facultyList = listFaculties;
    }
}
```

Також додано типові класи для факультетів, кафедр та груп. Далі буде додано лістинг для факультету, однак він буде приблизно однаковим для кожного зі структурних підрозділів.

```
public class Faculty {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    private Human head;

    public Human getHead() {
        return head;
    }

    public void setHead(Human head) {
        this.head = head;
    }

    private List<Department> departmentList;

    public List<Department> getDepartmentList() {
        return departmentList;
    }
    public void setDepartments(List<Department> listDepartments) {
        this.departmentList = listDepartments;
    }
}
```

Так як клас Student наслідує клас Human, було створено відповідний клас.

```
public class Human {
    protected String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    protected String surname;
    public String getSurname() {
        return surname;
    }
    public void setSurname(String surname) {
        this.surname = surname;
    }

    protected String patronymic;
    public String getPatronymic() {
        return patronymic;
    }
    public void setPatronymic(String patronymic) {
        this.patronymic = patronymic;
    }

    protected Sex sex;
    public Sex getSex() {
        return sex;
    }
    public void setSex(Sex sex) {
        this.sex = sex;
    }
}
```

Так як людина це типовий клас, і, в теорії, людина не обов'язково є студентом. Наприклад, навіть в цій програмі є голови кафедр, факультетів та університету, що не можуть бути студентами. Для обліку саме студентів класу студент додано поле Id.

```
public class Student extends Human {
    private int Id;
    public int Id() {
        return Id;
    }
    public void Id(int Id) {
        this.Id = Id;
    }
```

У кожної людини є поле стать. Його реалізовано за допомогою класу типу Enum

```
public enum Sex {
    MALE,
    FEMALE,
}
```

Для кожного класу створено інтерфейс. Усі інтерфейси знаходяться у package controller, на відміну від класів, що знаходились у package під назвою model.

Далі наведено типовий для всіх класів інтерфейс.

```
public class DepartmentCreator {

    public Department createDepartment(int countGroups) {

        Department department = new Department();
        GroupCreator groupCreator = new GroupCreator();

        List<Group> listGroups = new ArrayList<>();

        for (int i = 0; i < countGroups; i++) {
            listGroups.add(groupCreator.typicalGroup());
        }

        department.setGroupList(listGroups);

        return department;
    }

    public Department typicalDepartment() {

        return createDepartment(2);
    }
}
```

Клас Run знаходиться у кореневій папці проекту та запускає безпосередньо програму за допомогою створенню об’єкта класу University та визову методу createTypicalUniversity(). Далі можна побачити лістинг класу та результат компіляції проекту.

```
public class Run {

    public static void main(String[] args) {

        UniversityCreator universityCreator = new UniversityCreator();

        University university = universityCreator.createTypicalUniversity();

        System.out.println(university.toString());
    }

}
```

```
Run
C:\Users\admin\.jdks\openjdk-22\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\lib\idea_rt.jar=51675:C:\Program
 Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath
 C:\Users\admin\IdeaProjects\basic_java\lab3\out\production\_0;C:\Users\admin\.m2\repository\junit\junit\4.13.1\junit-4.13.1.jar;C:\Users\admin\
 .m2\repository\org\hamcrest\hamcrest-core\1.3\hamcrest-core-1.3.jar;C:\Users\admin\Downloads\gson-2.10.1.jar Run
University{name='myUniversity', head=Human{name='John', surname='Doe', patronymic='Johnson', sex=MALE}, faculties=[Faculty{head=Human{name='John', surname='Doe',
 patronymic='Johnson', sex=MALE}, departments=[Department{head=null, groups=[Group{head=null, students=[Student{recordBookNumber=1, name='John', surname='Doe',
 patronymic='Johnson', sex=MALE}, Student{recordBookNumber=1, name='John', surname='Doe', patronymic='Johnson', sex=MALE}]}, Group{head=null,
 students=[Student{recordBookNumber=1, name='John', surname='Doe', patronymic='Johnson', sex=MALE}, Student{recordBookNumber=1, name='John', surname='Doe',
 patronymic='Johnson', sex=MALE}]}]}, Department{head=null, groups=[Group{head=null, students=[Student{recordBookNumber=1, name='John', surname='Doe', patronymic='Johnson',
  sex=MALE}, Student{recordBookNumber=1, name='John', surname='Doe', patronymic='Johnson', sex=MALE}]}, Group{head=null, students=[Student{recordBookNumber=1, name='John',
 surname='Doe', patronymic='Johnson', sex=MALE}, Student{recordBookNumber=1, name='John', surname='Doe', patronymic='Johnson', sex=MALE}]}]}]}]}

Process finished with exit code 0
```

Рис. 17 – результат виконання програми.

Лабораторна робота номер 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.

Хід роботи

Для виконнаня роботи, довелось модифікувати класи групи model. Для перевірки збіжностей було додано методи що порівнюють об’єкти. Наданий лістинг є модифікацієї классу Department, але є типовим для всіх структурних підрозділів.

```
    @Override
    public boolean equals(Object o) {

        if (this == o) return true;

        if (!(o instanceof Department that)) return false;

        return groupList.equals(that.groupList);
    }

    @Override
    public int hashCode() {

        return Objects.hash(groupList);
    }
}
```

Для реалізації методів по роботі з json файлами було завантажено та додано до проекту сторонню бібліотеку gson від Google.

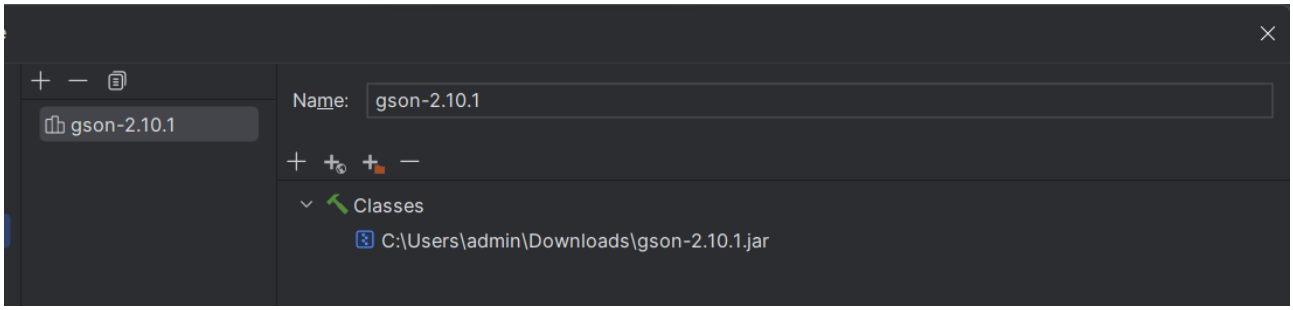


Рис. 18 – підключена зовнішня бібліотека

Що стосується саме Unit тесту, для цього було створено окремий package JUnitTest, у якому було створено 2 класи. Перший з них це JsonManager з методами по запису до json файлів та відповідно читання з цих файлів.

```
public class JsonManager {

    public void writeToJson(Object o, String filePath) {

        Gson gson = new Gson();

        String jsonFile = gson.toJson(o);

        try {
            FileWriter myWriter = new FileWriter(filePath);

            myWriter.write(jsonFile);
            myWriter.close();

            System.out.println("Written to Json File!");

        } catch (IOException e) {

            System.out.println(e.getMessage());

        }
    }

    public Object readFromJson(String filePath) {

        Gson gson = new Gson();

        try {
            System.out.println("Read from Json File!");
            return gson.fromJson(new FileReader(filePath), University.class);
        } catch (FileNotFoundException e) {

            System.out.println(e.getMessage());

            return null;
        }

    }
}
```

Клас JsonTest є класом, що запускає програму по тестуванню правильності відтворення інформації з json файлу, що вона була туда раніше записана.

```java
public class JsonTest {
    @Test
    public void testFileWriteJsonToFile() {

        String filePath = "./src/myUniversity.json";

        JsonManager jsonManager = new JsonManager();
        UniversityCreator universityCreator = new UniversityCreator();

        University oldUniversity = universityCreator.typicalUniversity();
        System.out.println(oldUniversity.toString());

        jsonManager.writeToJson(oldUniversity, filePath);
        University newUniversity = (University)
jsonManager.readFromJson(filePath);
        System.out.println(newUniversity.toString());

        Assert.assertEquals(newUniversity, oldUniversity);
    }
}
```

Як можна бачити з рисунку нижче, мітки про успішний запис та читання інформації були отримані в результаті виконання тесту. В консоль виведені об’єкти oldUniveristy newUniveristy для того, щоб можна було переконатись у правильності виконання коду. Всі операції виконувались за посередництвом файлу myUniversity.json що також є у кореневій папці проекту.

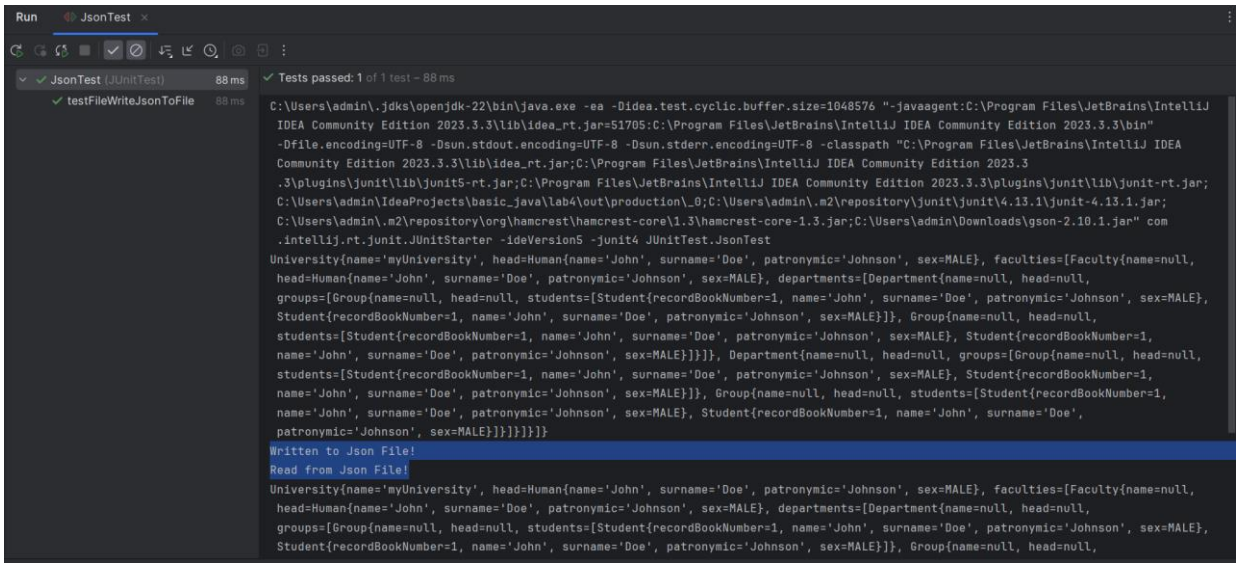


Рис. 19 – результат виконання програми

Рис. 20 – вміст json файлу

Лабораторна робота номер 5. Jdbc

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 6 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.

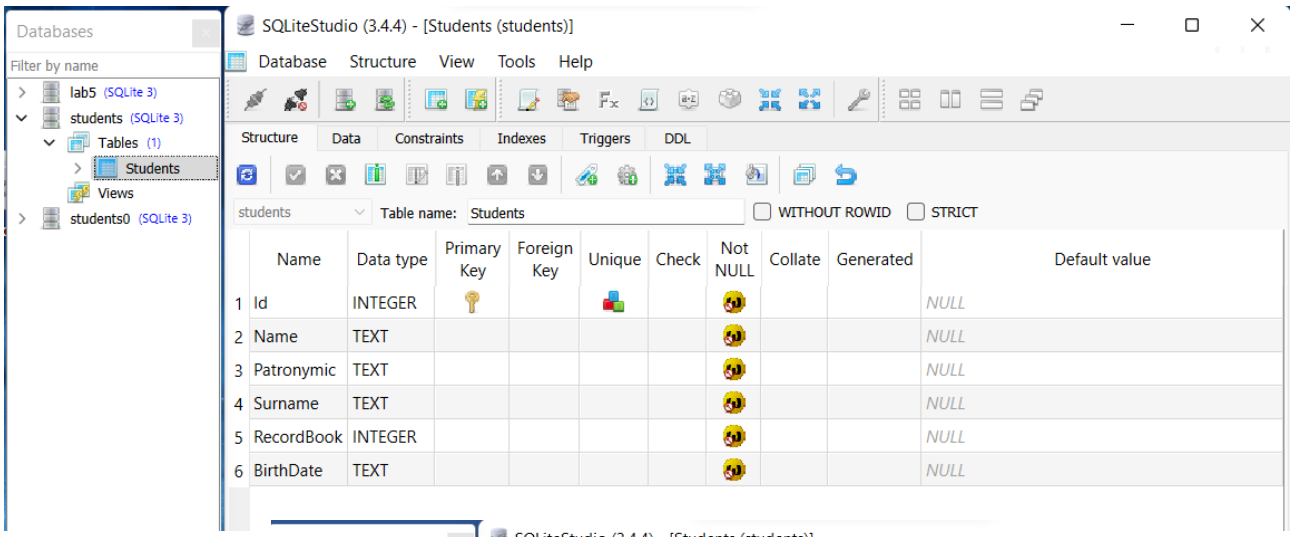
Хід роботи

Для підключення та роботи з базою даних завантажено та додано до проекту сторонню бібліотеку по роботі з базами даних SQLite під назвою sqlite_jdbc. Також для потреб цієї лабораторної роботи довелось завантажити сторонню бібліотеку slf4j_api. Під час компіляції проекту виникала помилка з текстом *java.lang.NoClassDefFoundError: org.slf4j.LoggerFactory* і для її вирішення цієї помилки потрібно було завантажити цей API.

У якості серверу для створення таблиці бази даних було обрано додаток SQLite Studio відповідно на основі SQLite. Було обрано саме цю СУБД

та додаток для її реалізації вибрано в першу чергу через попередню ознайомленість та зручності у користуванні. Із однозначних переваг саме цього рішення можна виділити те, що ця СУБД є мінімалістичною, простою та швидкою у користуванні, що є дуже корисним під час розробки невеликих проектів на кшталт даного. Із недоліків можна виділити частково урізаний функціонал. Наприклад, у ході виконання роботи виявилось, що без сторонніх «милиць» неможливо створити поле формату DATE.

Саме через цей недолік було обрано підхід, при якому до бази даних надходить запит на вибірку усього списку студентів, а на стороні Java програми вже відбувається фільтрація за ознакою. Нижче можна побачити структуру таблиці та її наповнення.



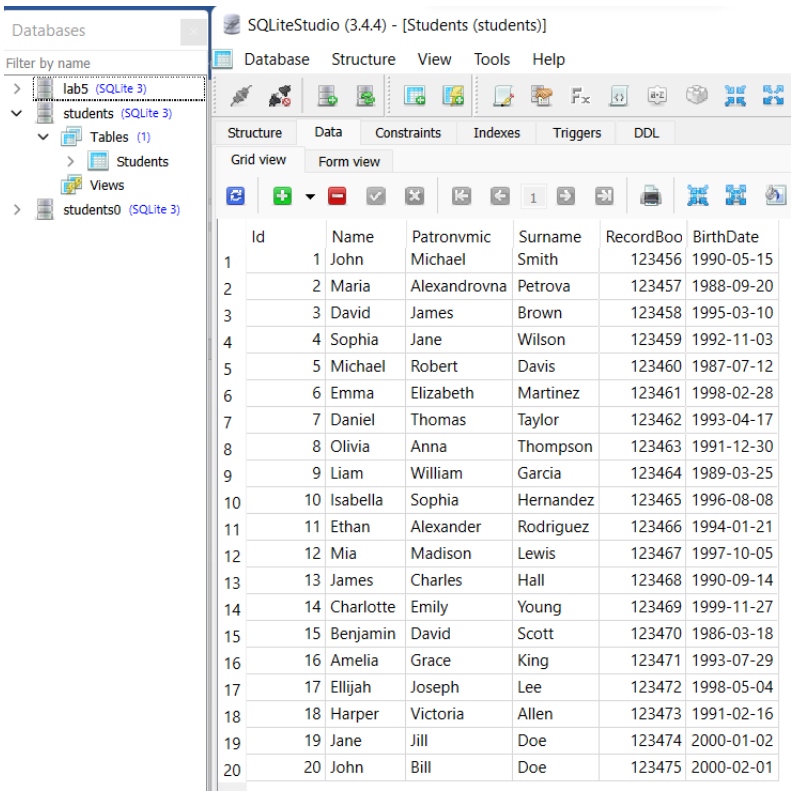


Рис. 21, 22 – структура та вміст БД

Далі наведено лістинг класу, що реалізує метод підключення бази даних до Java програми.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DbConnection {
    public static Connection connect(){
        Connection con = null;
        try{
            Class.forName("org.sqlite.JDBC");
            con = DriverManager.getConnection("jdbc:sqlite:students.db");

        } catch (ClassNotFoundException | SQLException e){
            System.out.println(e.getMessage());

        }
        return con;
    }
}
```

Після цього була написана програма, що реалізує метод з попереднього класу для вибірки даних з бази даних.

```
import java.util.Scanner;
import java.sql.*;

public class SQLRequest {
    public static void getStudentsByMonth(String month) {

        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
            connection = DbConnection.connect();
            statement = connection.createStatement();

            String sqlQuery = "SELECT * FROM Students WHERE substr(BirthDate, 6,
2) = '" + month + "'";
            resultSet = statement.executeQuery(sqlQuery);

            int count = 0;
            while (true)
            {
                if(!resultSet.next()) {
                    if(count == 0)
                    {
                        System.out.println("No students born this month!");
                    }return;
                }

                count++;

                int id = resultSet.getInt("id");
                String name = resultSet.getString("Name");
                String surname = resultSet.getString("Surname");
                String patronymic = resultSet.getString("Patronymic");
                String birthDate = resultSet.getString("BirthDate");
                String recordBook = resultSet.getString("RecordBook");
```

```
                System.out.println("ID: " + id);
                System.out.println("Name: " + name);
                System.out.println("Patronymic: " + patronymic);
                System.out.println("Surname: " + surname);
                System.out.println("BirthDate: " + birthDate);
                System.out.println("Record book number: " + recordBook);
                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();
            }
        }
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter number of month: ");

        String month = scanner.nextLine();
        String newMonth = month.length() > 1 ? month : '0'+month;

        getStudentsByMonth(newMonth);

    }
}
```

Розбираючи програму по частинам:

Після підключення бібліотек, оголошення класу та методу, йде створення об’єктів для підключення бази даних до проекту. Було б доречно створити їх під час реалізації, однак програма не може їх бачити у внутрішньому try, тому вони створюються у голові методу.

```
import java.util.Scanner;
import java.sql.*;

public class SQLRequest {
    public static void getStudentsByMonth(String month) {

        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
```

Далі виконується безпосередньо підключення БД, створення SQL запиту та реалізації вибірки дописів студентів саме з указаним місяцем народження. Далі оголошується змінна count та умова, що використовується для того, щоб до перебору усього масиву записів можна було вихнаичти чи існують в БД

студенти з указаним місяцем народження. Після цього можна приступати до створення вибірки студентів та формування структурних дописів для виводу у консоль та обробка помилок при не існуванні підключення.

```
try {
    connection = DbConnection.connect();
    statement = connection.createStatement();

    String sqlQuery = "SELECT * FROM Students WHERE substr(BirthDate, 6, 2) = '"
+ month + "'";
    resultSet = statement.executeQuery(sqlQuery);

    int count = 0;
    while (true)
    {
        if(!resultSet.next()) {
            if(count == 0)
            {
                System.out.println("No students born this month!");
            }return;
        }

        count++;

        int id = resultSet.getInt("id");
        String name = resultSet.getString("Name");
        String surname = resultSet.getString("Surname");
        String patronymic = resultSet.getString("Patronymic");
        String birthDate = resultSet.getString("BirthDate");
        String recordBook = resultSet.getString("RecordBook");

        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Patronymic: " + patronymic);
        System.out.println("Surname: " + surname);
        System.out.println("BirthDate: " + birthDate);
        System.out.println("Record book number: " + recordBook);
        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();
        }
    }
}
```

Нарешті, метод main що зчитує з консолі указаний місяць народження та приводить його до формату поля БД з датою народження, та виклик методу по створенню вибірки. Нижче приведено скріншоти виконання програми. Зі скриншотів №1 та №2 можна побачити, що програма виконується при введені місяців як одним числом, так і двома. На скриншоті №3 видно, що програма

коректно працює при введені червня, тобто 6 місяця, у якому за умовою завдання не народився ні один студент з групи.

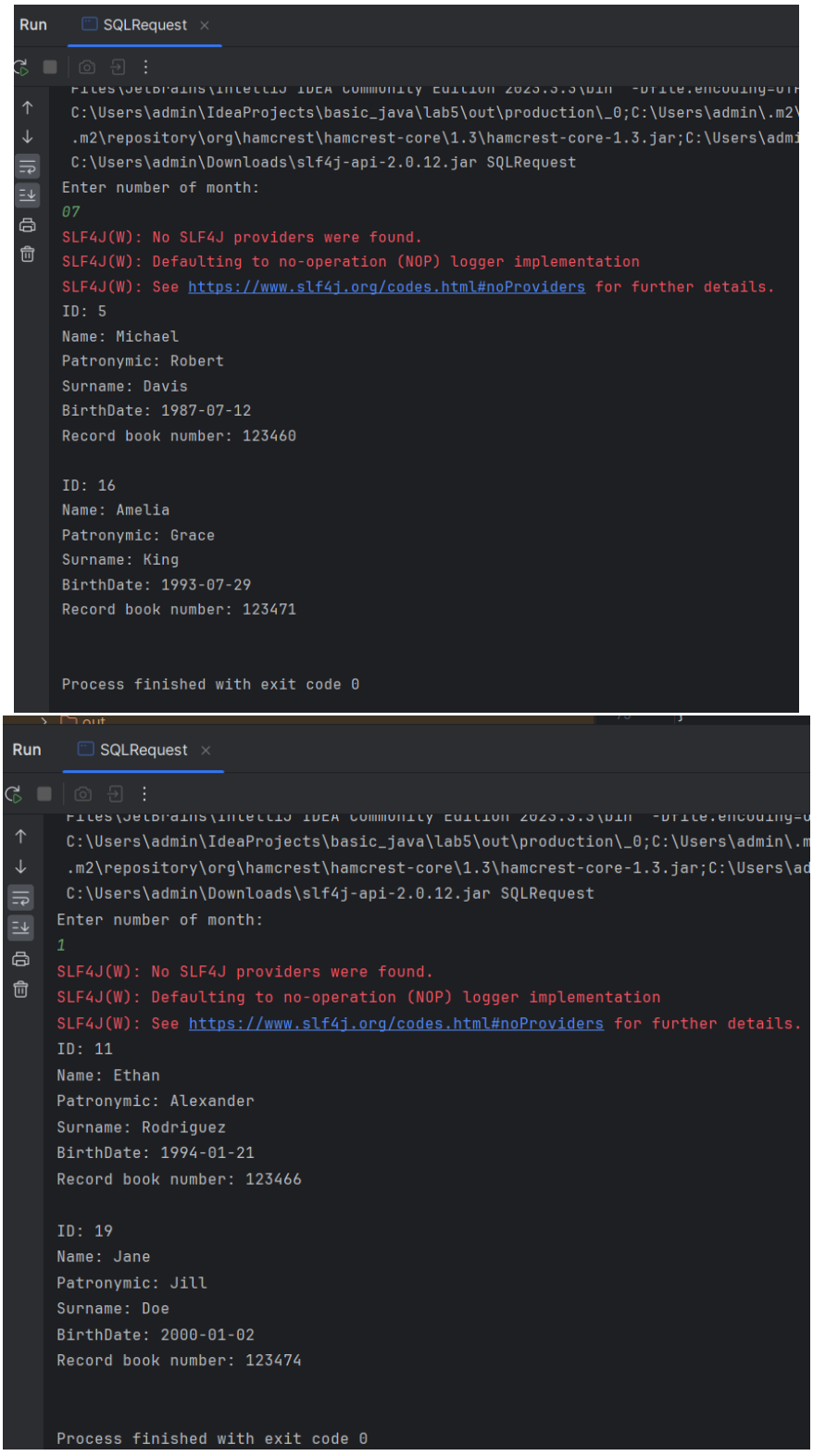


Рис. 23, 24 – результат виконання програми при різних умовах.

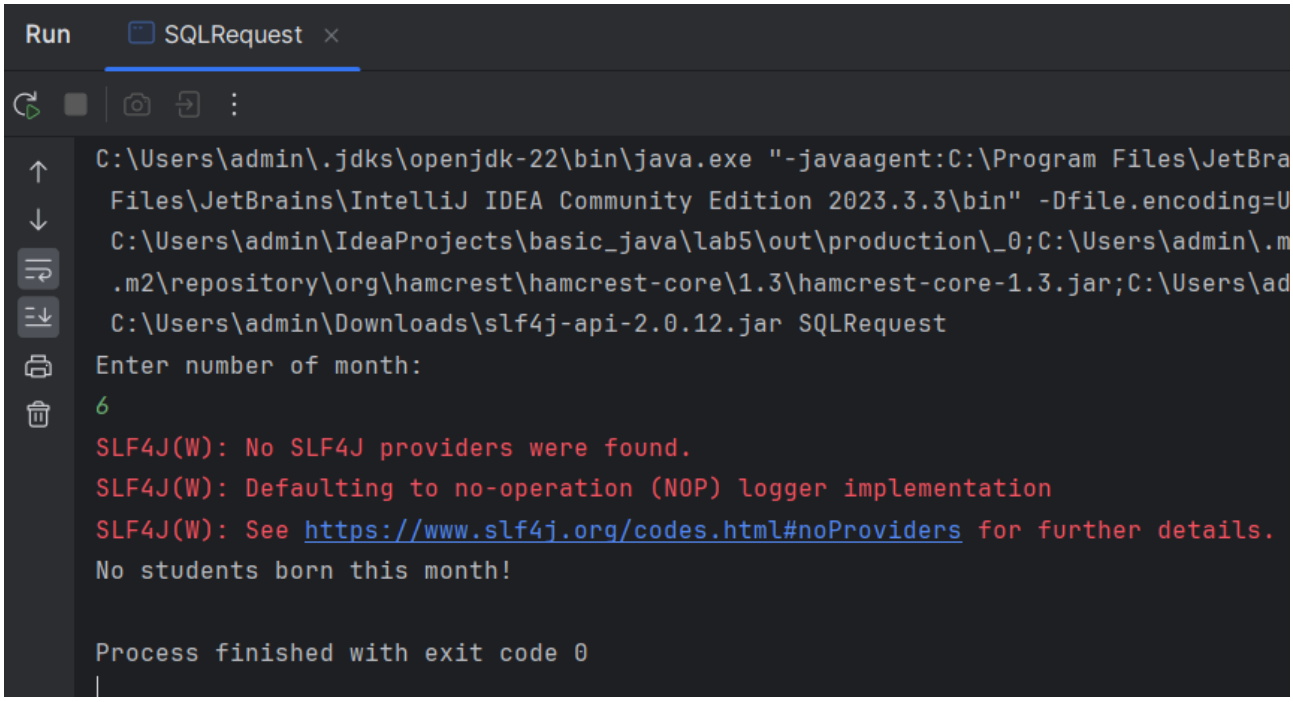


Рис. 25 – результат роботи програми при указані «порожнього» місяцю

Висновки: виконуючи лабораторні роботи з курсу Базове програмування на мові Java було здобуто знання про основи програмування на Java, про принципи методології ООП, про роботу з файлами типу json та про роботу з базами даних на мові SQL. Під час виконнаня робіт було здобуто навички про написання коду для невеликих програм, призначених для виконання вузького спектру задач.