

**Міністерство освіти і науки України**  
**Національний технічний університет**  
**«Дніпровська політехніка»**



**ЗВІТ**  
**Практичні роботи № 0-5**  
**з дисципліни**  
**«Поглиблене програмування в середовищі Java»**

**Виконала:**

ст. гр. 122-21-3  
Філіппова Х. С.

**Прийняв:**

Доцент каф. САіУ  
Мінєєв О. С.

**м. Дніпро**  
**2025 рік**

# ПОСИЛАННЯ НА РЕПОЗИТОРІЙ:

[https://github.com/philiipka/Java\\_Basic](https://github.com/philiipka/Java_Basic)

## Практична робота №0

### Hello World

**Завдання:** Встановити IntelliJ 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

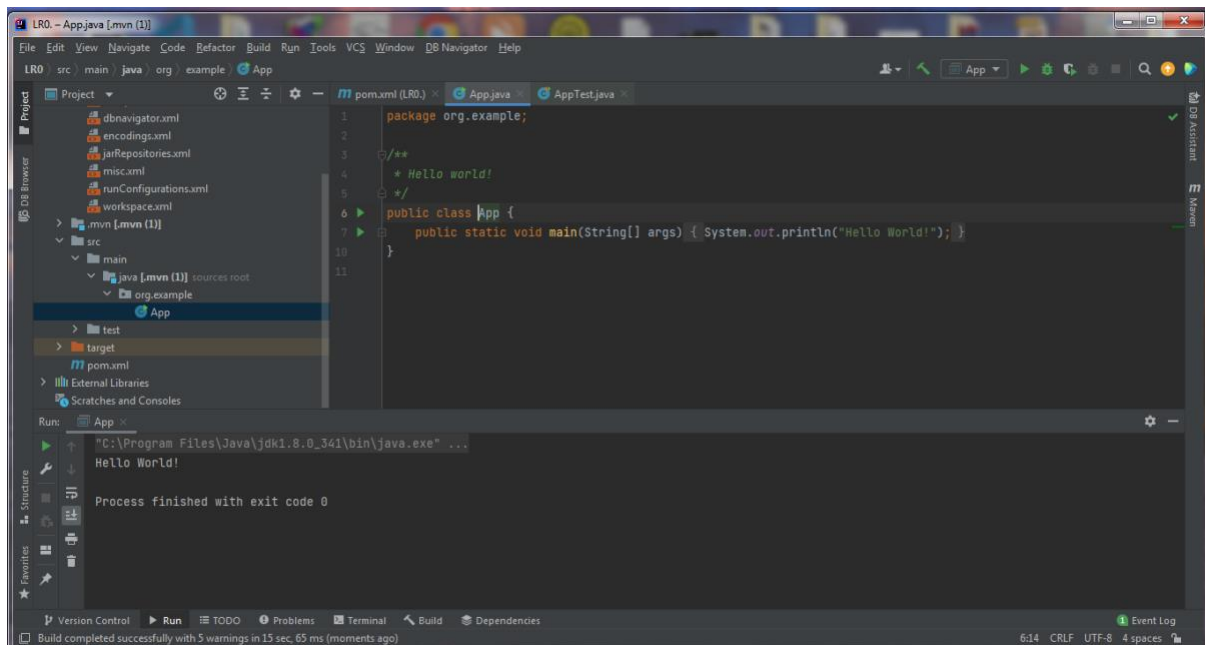


Рисунок 1.1 – Результат виконання завдання

**Висновок:** у ході лабораторної роботи вивчено процес налаштування середовища розробки IntelliJ IDEA та JDK, а також створення та виконання базової Java-програми.

## Практична робота №1

### CVS. GIT

**Завдання:** 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 окремій 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

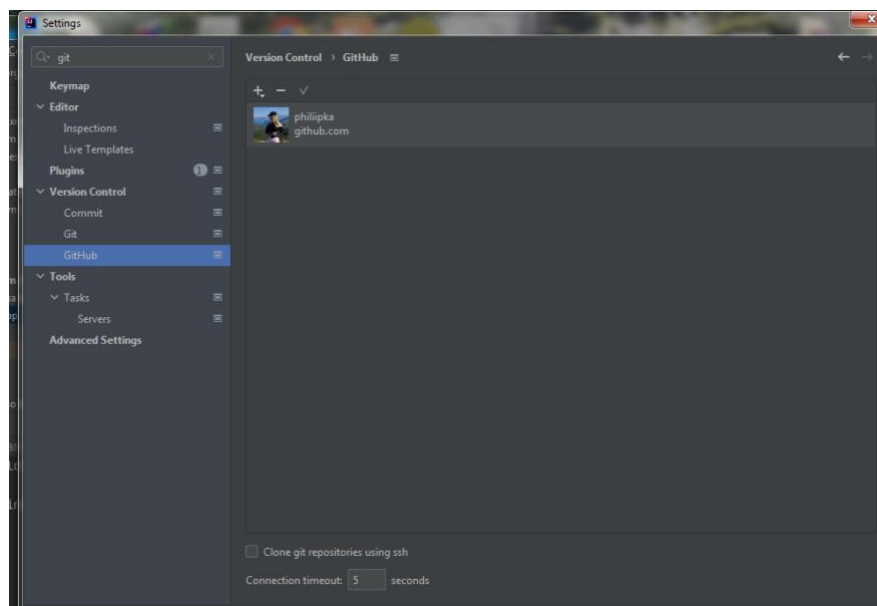


Рисунок 1.2 – Результат виконання завдання

**Висновок:** у ході лабораторної роботи вивчено підключення Git до IntelliJ IDEA, створення репозиторію на GitHub, управління гілками та робота з системою контролю версій.

## Практична робота №2

### Основи

**Завдання:** 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. Діапазон рандомних чисел для створення елементів матриці повинен зверігатись в спеціальних константах.

Як завдання підвищеної складності додати розрахунок середнього геометричного елементів матриці.

### Хід роботи

Код програми:

```
import java.util.Random;
import java.util.Scanner;

public class MatrixOperations {
    private static final int MAX_SIZE = 20;
    private static final int MIN_RANDOM = -100;
    private static final int MAX_RANDOM = 100;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter matrix width (max 20): ");
        int width = scanner.nextInt();
        System.out.print("Enter matrix height (max 20): ");
        int height = scanner.nextInt();

        if (width <= 0 || width > MAX_SIZE || height <= 0 || height > MAX_SIZE) {
            System.out.println("Invalid matrix size.");
            return;
        }
    }
}
```

```

        System.out.print("Enter 1 for manual input, 2 for random fill: ");
        int choice = scanner.nextInt();
        int[][] matrix = (choice == 1) ? createMatrixManually(scanner, width, height) :
createMatrixRandomly(width, height);

        printMatrix(matrix);

        int min = findMin(matrix);
        int max = findMax(matrix);
        double average = calculateAverage(matrix);
        double geometricMean = calculateGeometricMean(matrix);

        System.out.println("Min element: " + min);
        System.out.println("Max element: " + max);
        System.out.println("Arithmetic mean: " + average);
        System.out.println("Geometric mean: " + geometricMean);
    }

    private static int[][] createMatrixManually(Scanner scanner, int width, int height) {
        int[][] matrix = new int[height][width];
        System.out.println("Enter matrix elements:");
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
        return matrix;
    }

    private static int[][] createMatrixRandomly(int width, int height) {
        int[][] matrix = new int[height][width];
        Random random = new Random();
        for (int i = 0; i < height; i++) {
            for (int j = 0; j < width; j++) {
                matrix[i][j] = random.nextInt(MAX_RANDOM - MIN_RANDOM + 1) + MIN_RANDOM;
            }
        }
        return matrix;
    }

    private static void printMatrix(int[][] matrix) {
        System.out.println("Matrix:");
        for (int[] row : matrix) {
            for (int elem : row) {
                System.out.printf("%4d", elem);
            }
            System.out.println();
        }
    }
}

```

```

private static int findMin(int[][] matrix) {
    int min = Integer.MAX_VALUE;
    for (int[] row : matrix) {
        for (int elem : row) {
            if (elem < min) min = elem;
        }
    }
    return min;
}

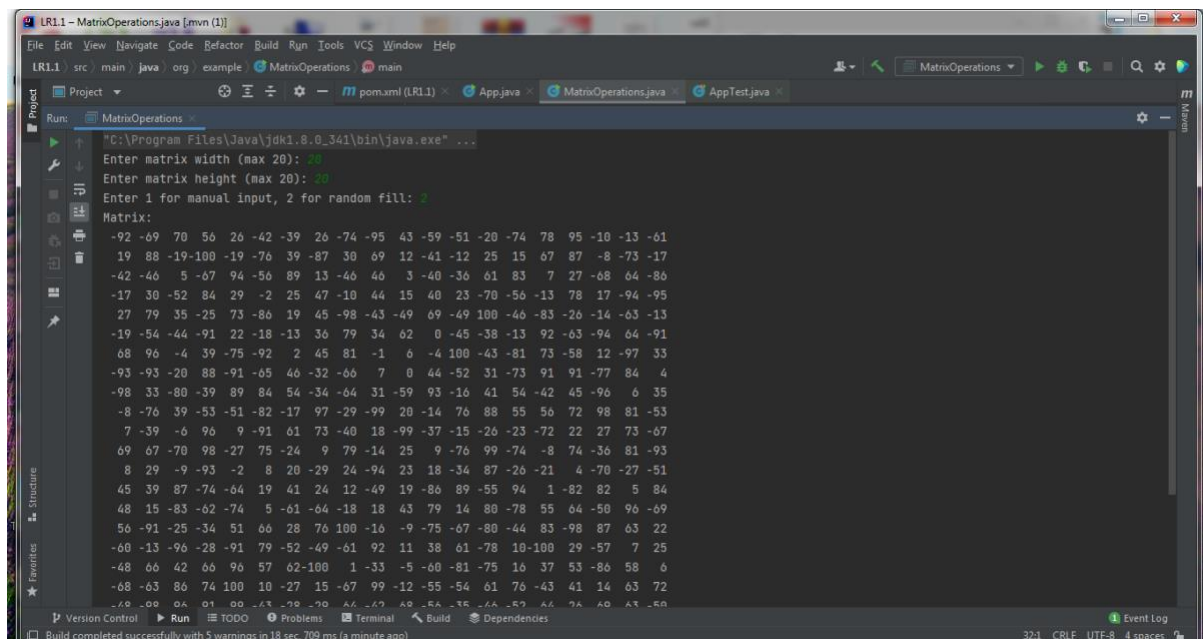
private static int findMax(int[][] matrix) {
    int max = Integer.MIN_VALUE;
    for (int[] row : matrix) {
        for (int elem : row) {
            if (elem > max) max = elem;
        }
    }
    return max;
}

private static double calculateAverage(int[][] matrix) {
    int sum = 0, count = 0;
    for (int[] row : matrix) {
        for (int elem : row) {
            sum += elem;
            count++;
        }
    }
    return (double) sum / count;
}

private static double calculateGeometricMean(int[][] matrix) {
    double product = 1.0;
    int count = 0;
    for (int[] row : matrix) {
        for (int elem : row) {
            product *= Math.abs(elem) + 1; // Avoiding zero values
            count++;
        }
    }
    return Math.pow(product, 1.0 / count) - 1;
}

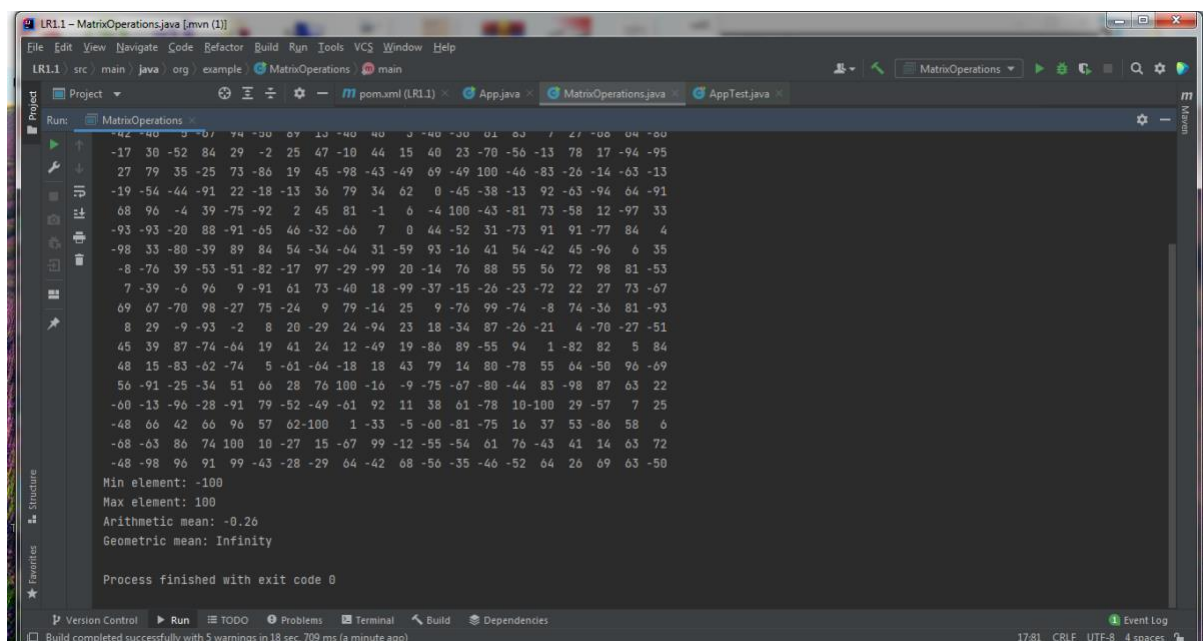
```

Результат програми:



```
LR1.1 - MatrixOperations.java [mnv (1)]
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
LR1.1 src / main / java / org / example / MatrixOperations main
Project Run: MatrixOperations
"C:\Program Files\Java\jdk1.8.0_341\bin\java.exe" ...
Enter matrix width (max 20): 20
Enter matrix height (max 20): 20
Enter 1 for manual input, 2 for random fill: 2
Matrix:
-92 -69 70 56 26 -42 -39 26 -74 -95 43 -59 -51 -20 -74 78 95 -10 -13 -61
19 88 -19 -100 -19 -76 39 -87 30 69 12 -41 -12 25 15 67 87 -8 -73 -17
-42 -46 5 -67 94 -56 89 13 -46 46 3 -40 -36 61 83 7 27 -68 64 -86
-17 30 -52 84 29 -2 25 47 -10 44 15 40 23 -70 -56 -13 78 17 -94 -95
27 79 35 -25 73 -86 19 45 -98 -43 -49 69 -49 100 -46 -83 -26 -14 -63 -13
-19 -54 -44 -91 22 -18 -13 36 79 34 62 0 -45 -38 -13 92 -63 -94 64 -91
68 96 -4 39 -75 -92 2 45 81 -1 6 -4 100 -43 -81 73 -58 12 -97 33
-93 -93 -20 88 -91 -65 46 -32 -66 7 0 44 -52 31 -73 91 91 -77 84 4
-98 33 -80 -39 89 84 54 -34 -64 31 -59 93 -16 41 54 -42 45 -96 6 35
-8 -76 39 -53 -51 -82 -17 97 -29 -99 20 -14 76 88 55 56 72 98 81 -53
7 -39 -6 96 9 -91 61 73 -40 18 -99 -37 -15 -26 -23 -72 22 27 73 -67
69 67 -70 98 -27 75 -24 9 79 -14 25 9 -76 99 -74 -8 74 -36 81 -93
8 29 -9 -93 -2 8 20 -29 24 -94 23 18 -34 87 -26 -21 4 -70 -27 -51
45 39 87 -74 -64 19 41 24 12 -49 19 -86 89 -55 94 1 -82 82 5 84
48 15 -83 -62 -74 5 -61 -64 -18 18 43 79 14 80 -78 55 64 -50 96 -69
56 -91 -25 -34 51 66 28 76 100 -16 -9 -75 -67 -80 -44 83 -98 87 63 22
-60 -13 -96 -28 -91 79 -52 -49 -61 92 11 38 61 -78 10 -100 29 -57 7 25
-48 66 42 66 96 57 62 -100 1 -33 -5 -60 -81 -75 16 37 53 -86 58 6
-68 -63 86 74 100 10 -27 15 -67 99 -12 -55 -54 61 76 -43 41 14 63 72
-48 -98 96 91 99 -43 -28 -29 64 -42 68 -56 -35 -46 -52 64 26 69 63 -50
Process finished with exit code 0
Build completed successfully with 5 warnings in 18 sec, 709 ms (a minute ago)
```

Рисунок 1.3 – Результат виконання завдання



```
LR1.1 - MatrixOperations.java [mnv (1)]
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
LR1.1 src / main / java / org / example / MatrixOperations main
Project Run: MatrixOperations
-42 -46 5 -67 94 -56 89 13 -46 46 3 -40 -36 61 83 7 27 -68 64 -86
-17 30 -52 84 29 -2 25 47 -10 44 15 40 23 -70 -56 -13 78 17 -94 -95
27 79 35 -25 73 -86 19 45 -98 -43 -49 69 -49 100 -46 -83 -26 -14 -63 -13
-19 -54 -44 -91 22 -18 -13 36 79 34 62 0 -45 -38 -13 92 -63 -94 64 -91
68 96 -4 39 -75 -92 2 45 81 -1 6 -4 100 -43 -81 73 -58 12 -97 33
-93 -93 -20 88 -91 -65 46 -32 -66 7 0 44 -52 31 -73 91 91 -77 84 4
-98 33 -80 -39 89 84 54 -34 -64 31 -59 93 -16 41 54 -42 45 -96 6 35
-8 -76 39 -53 -51 -82 -17 97 -29 -99 20 -14 76 88 55 56 72 98 81 -53
7 -39 -6 96 9 -91 61 73 -40 18 -99 -37 -15 -26 -23 -72 22 27 73 -67
69 67 -70 98 -27 75 -24 9 79 -14 25 9 -76 99 -74 -8 74 -36 81 -93
8 29 -9 -93 -2 8 20 -29 24 -94 23 18 -34 87 -26 -21 4 -70 -27 -51
45 39 87 -74 -64 19 41 24 12 -49 19 -86 89 -55 94 1 -82 82 5 84
48 15 -83 -62 -74 5 -61 -64 -18 18 43 79 14 80 -78 55 64 -50 96 -69
56 -91 -25 -34 51 66 28 76 100 -16 -9 -75 -67 -80 -44 83 -98 87 63 22
-60 -13 -96 -28 -91 79 -52 -49 -61 92 11 38 61 -78 10 -100 29 -57 7 25
-48 66 42 66 96 57 62 -100 1 -33 -5 -60 -81 -75 16 37 53 -86 58 6
-68 -63 86 74 100 10 -27 15 -67 99 -12 -55 -54 61 76 -43 41 14 63 72
-48 -98 96 91 99 -43 -28 -29 64 -42 68 -56 -35 -46 -52 64 26 69 63 -50
Min element: -100
Max element: 100
Arithmetic mean: -0.26
Geometric mean: Infinity
Process finished with exit code 0
Build completed successfully with 5 warnings in 18 sec, 709 ms (a minute ago)
```

Рисунок 1.4 – Результат виконання завдання

**Висновок:** у ході лабораторної роботи вивчено створення та обробку двовимірних масивів у Java, роботу зі сканером введення та генерацію випадкових чисел. Реалізовано методи пошуку мінімального та максимального елементів, обчислення середнього арифметичного та середнього геометричного.

## Практична робота №3

### ООП

**Завдання:** 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 контролер (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



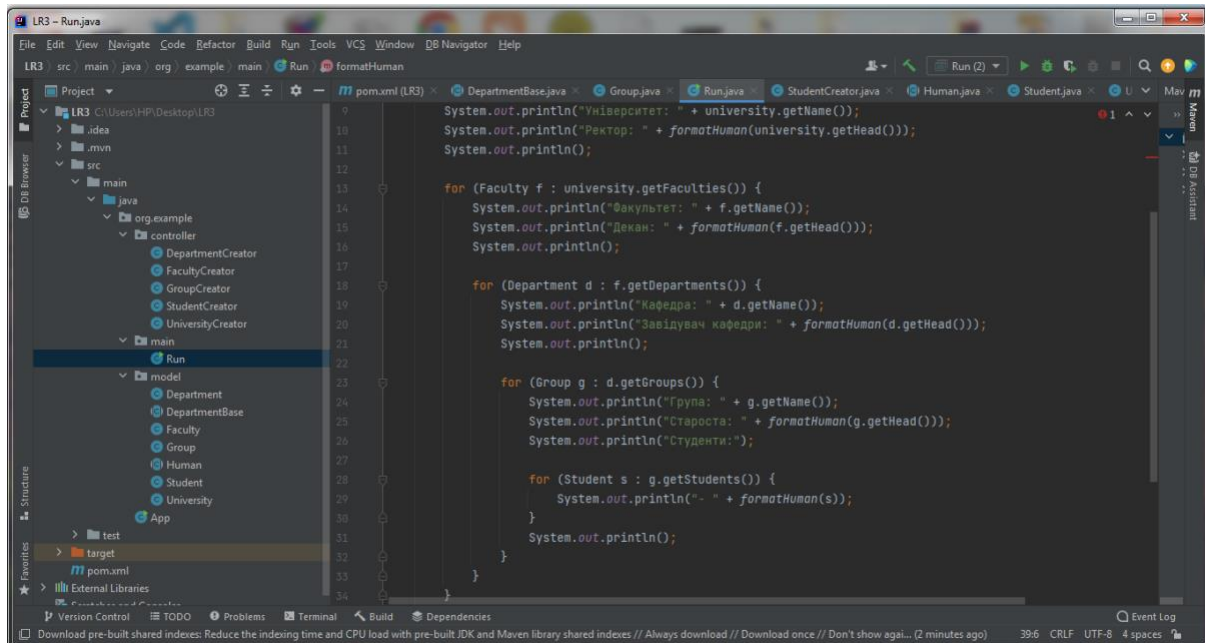


Рисунок 1.5 – Результат виконання завдання

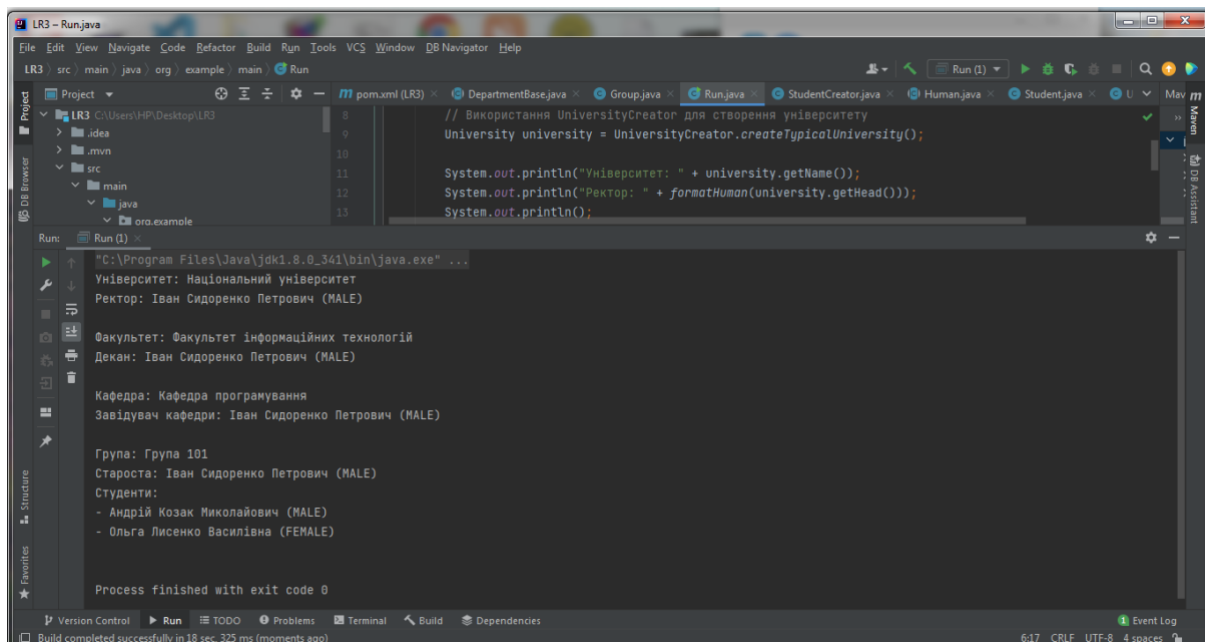


Рисунок 1.6 – Результат виконання завдання

**Висновок:** у ході лабораторної роботи досліджено концепцію об'єктно-орієнтованого програмування (ООП), створення класів, наслідування, абстрактних класів та інтерфейсів. Було розроблено структуру університету відповідно до патерну MVC.

## Практична 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 проєкті 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

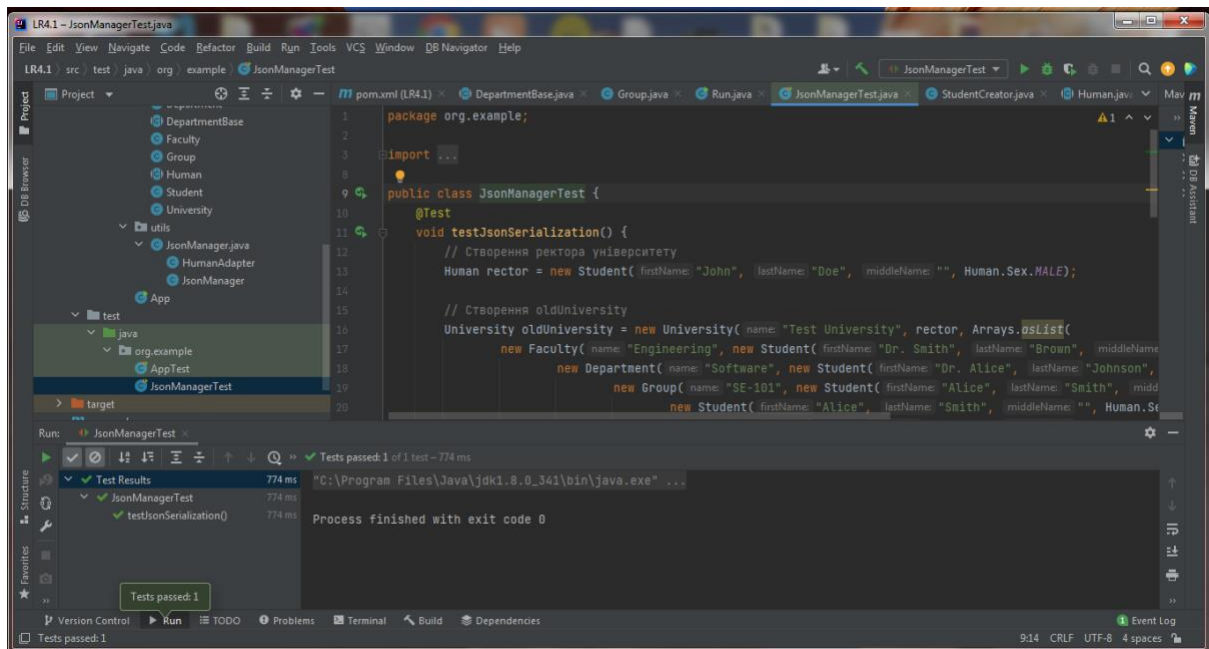


Рисунок 1.7 – Результат виконання завдання

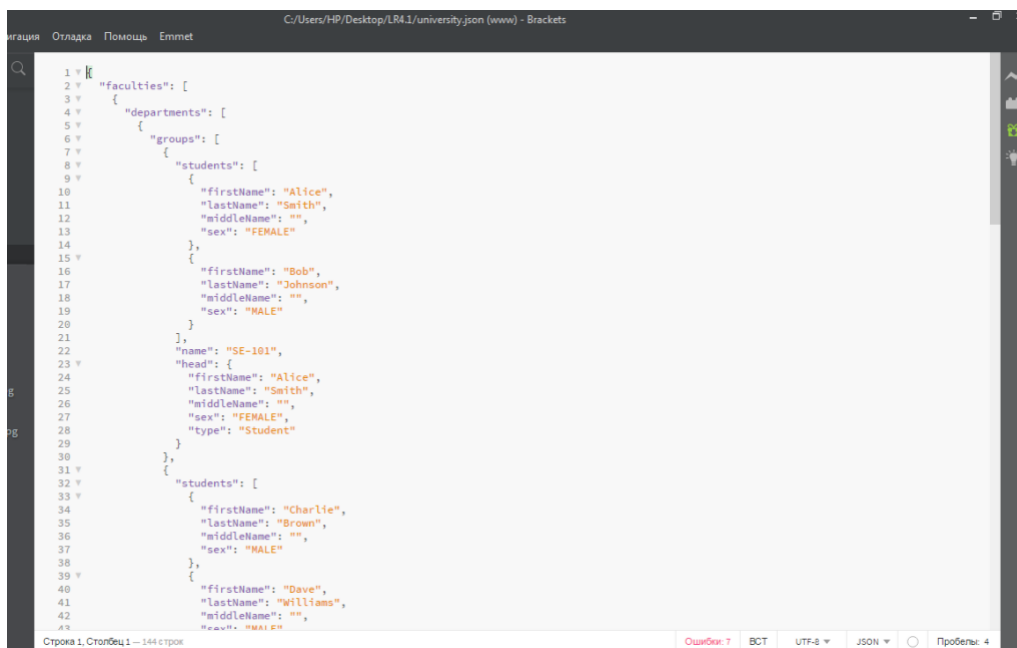


Рисунок 1.8 – Результат виконання завдання

**Висновок:** у ході лабораторної роботи вивчено серіалізацію та десеріалізацію об'єктів у форматі JSON, запис та зчитування даних у файли. Також освоєно тестування за допомогою JUnit та перевірку коректності обробки даних.

## Практична 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 запити. Вимог до розробки бази даних немає. Програма ж має бути написана за усіма стандартами ООП. Та може бути спроектована за двох принципів:

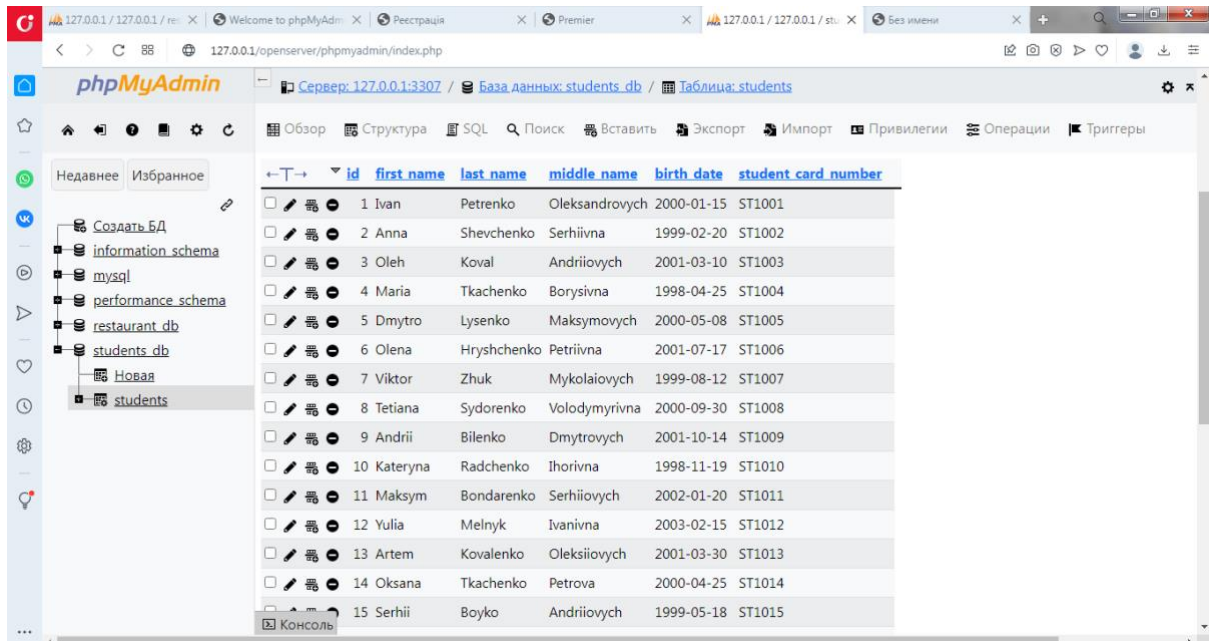
- при будь-якій ситуації буде забиратися весь перелік студентів, а вже на стороні java буде зроблено пошук необхідного
- SQL запит буде сформований згідно запиту який зробив користувач і вже сервер управління баз даних буде вирішувати, які самі студенти народилися в тому чи іншому місяці.

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 в проекті 6 лабораторної роботи в файлі database в папечці 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



|                          | id | first_name | last_name   | middle_name    | birth_date | student_card_number |
|--------------------------|----|------------|-------------|----------------|------------|---------------------|
| <input type="checkbox"/> | 1  | Ivan       | Petrenko    | Oleksandrovych | 2000-01-15 | ST1001              |
| <input type="checkbox"/> | 2  | Anna       | Shevchenko  | Serhiivna      | 1999-02-20 | ST1002              |
| <input type="checkbox"/> | 3  | Oleh       | Koval       | Andriiovych    | 2001-03-10 | ST1003              |
| <input type="checkbox"/> | 4  | Maria      | Tkachenko   | Borysivna      | 1998-04-25 | ST1004              |
| <input type="checkbox"/> | 5  | Dmytro     | Lysenko     | Maksymovych    | 2000-05-08 | ST1005              |
| <input type="checkbox"/> | 6  | Olena      | Hryshchenko | Petriivna      | 2001-07-17 | ST1006              |
| <input type="checkbox"/> | 7  | Viktor     | Zhuk        | Mykolaiovych   | 1999-08-12 | ST1007              |
| <input type="checkbox"/> | 8  | Tetiana    | Sydorenko   | Volodymyrivna  | 2000-09-30 | ST1008              |
| <input type="checkbox"/> | 9  | Andrii     | Bilenko     | Dmytrovych     | 2001-10-14 | ST1009              |
| <input type="checkbox"/> | 10 | Kateryna   | Radchenko   | Ihorivna       | 1998-11-19 | ST1010              |
| <input type="checkbox"/> | 11 | Maksym     | Bondarenko  | Serhiiovych    | 2002-01-20 | ST1011              |
| <input type="checkbox"/> | 12 | Yulia      | Melnyk      | Ivanivna       | 2003-02-15 | ST1012              |
| <input type="checkbox"/> | 13 | Artem      | Kovalenko   | Oleksiiiovych  | 2001-03-30 | ST1013              |
| <input type="checkbox"/> | 14 | Oksana     | Tkachenko   | Petrova        | 2000-04-25 | ST1014              |
| <input type="checkbox"/> | 15 | Serhii     | Boyko       | Andriiovych    | 1999-05-18 | ST1015              |

Рисунок 1.9 – Результат виконання завдання

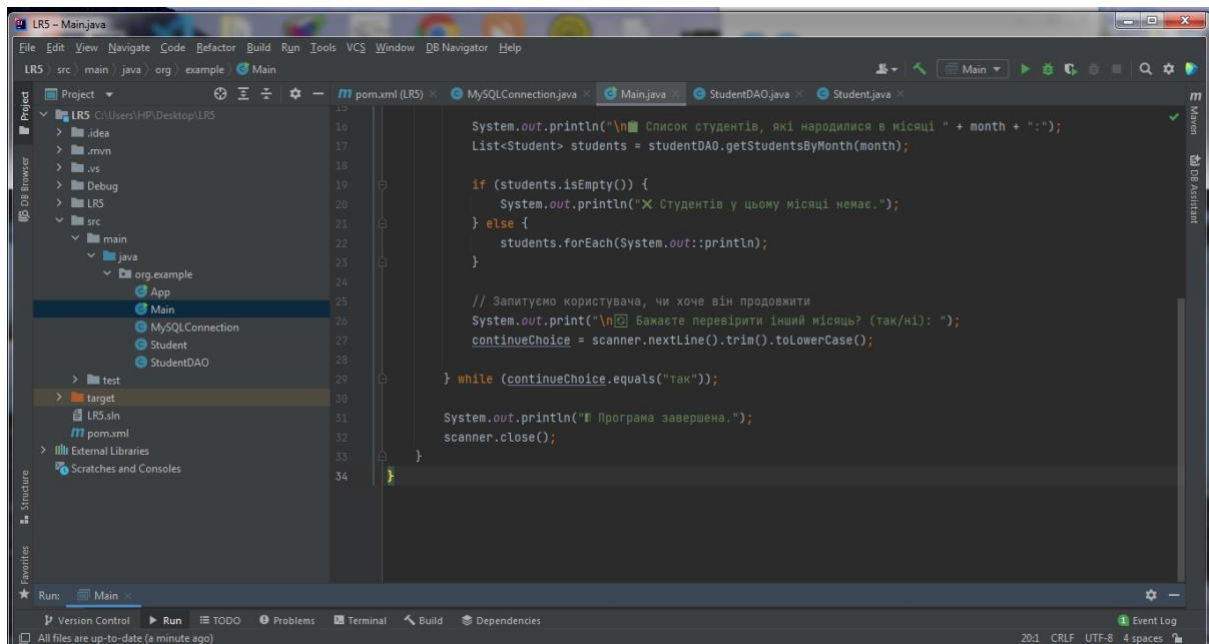


Рисунок 1.10 – Результат виконання завдання

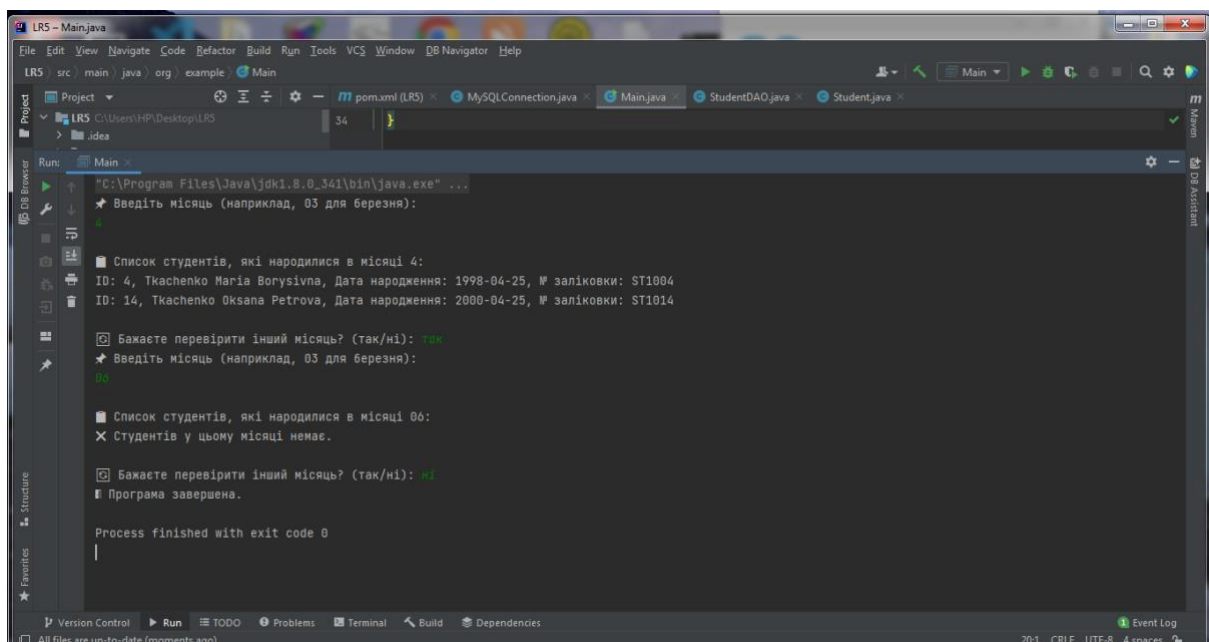


Рисунок 1.11 – Результат виконання завдання

Для реалізації завдання було створено базу даних students\_db у MySQL (phpMyAdmin). Вона містить таблицю students з такими полями:

id (INT, PRIMARY KEY, AUTO\_INCREMENT) – унікальний ідентифікатор студента

last\_name (VARCHAR) – прізвище

first\_name (VARCHAR) – ім'я

middle\_name (VARCHAR) – по батькові

birth\_date (DATE) – дата народження

student\_card\_number (VARCHAR) – номер залікової книжки

SQL-код створення таблиці розміщено у файлі database.sql в пакеті resources

У файлі MySqlConnection.java реалізовано підключення до бази даних через JDBC. Використано наступні параметри:

URL: jdbc:mysql://127.0.0.1:3307/students\_db

USER: root

PASSWORD: 1234

Клас містить статичний блок ініціалізації, що забезпечує підключення при завантаженні класу.

Реалізація класу Student

У файлі Student.java створено клас Student, який представляє модель студента. Він містить поля, конструктор та метод toString(), що виводить інформацію про студента.

Також є метод getBirthMonth(), який повертає місяць народження у вигляді рядка.

У файлі StudentDAO.java створено Data Access Object для взаємодії з базою даних. Реалізовано два методи:

`getAllStudents()` – отримує повний список студентів

`getStudentsByMonth(String month)` – повертає студентів, народжених у зазначеному місяці (параметр `month` – число від 1 до 12)

Методи використовують підключення через `MySQLConnection.getConnection()` і виконують `PreparedStatement` для безпечного виконання SQL-запитів.

Який принцип вибрано для реалізації пошуку студентів за місяцем?

Використано підхід: SQL-запит формується динамічно залежно від введеного користувачем місяця:

Цей підхід є більш ефективним, оскільки обробка даних відбувається на рівні сервера бази даних.

**Висновок:** у ході лабораторної роботи вивчено підключення Java-додатків до бази даних через JDBC, створення та виконання SQL-запитів. Було реалізовано пошук студентів за місяцем народження за допомогою SQL-запиту, що забезпечує ефективну вибірку даних на рівні сервера БД.