МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет «Дніпровська політехніка»

Кафедра ПЗКС

ЗВІТ

з практичної роботи №5

дисципліни «Поглиблене програмування Java»

Виконав: ст. гр. 122-21-1

Сарібекян Андрій Арменович

Перевірив: доц. Мінєєв О.С.

ас. Шевченко Ю.О.

Дніпро

2024

Практична робота №5

Jdbc

Мета роботи: навчитися працювати з Jdbc

Завдання до виконання:

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.

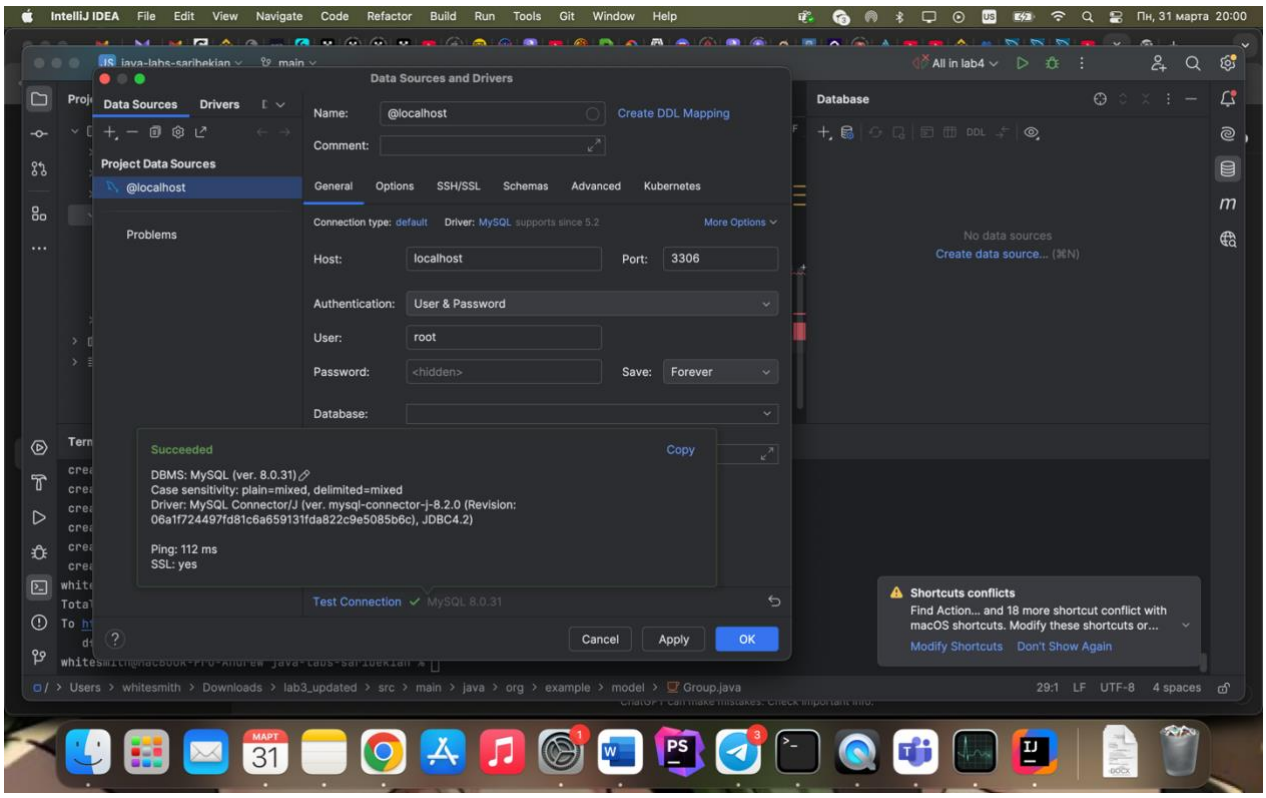
SQL код створення бази даних розмістити проекті 6 лабораторної роботи в файлі database в пекеджі resources. Для використання цієї лабораторної

роботи рекомендується активно використовувати знання отримані на дисципліні що стосуються розробки баз даних.

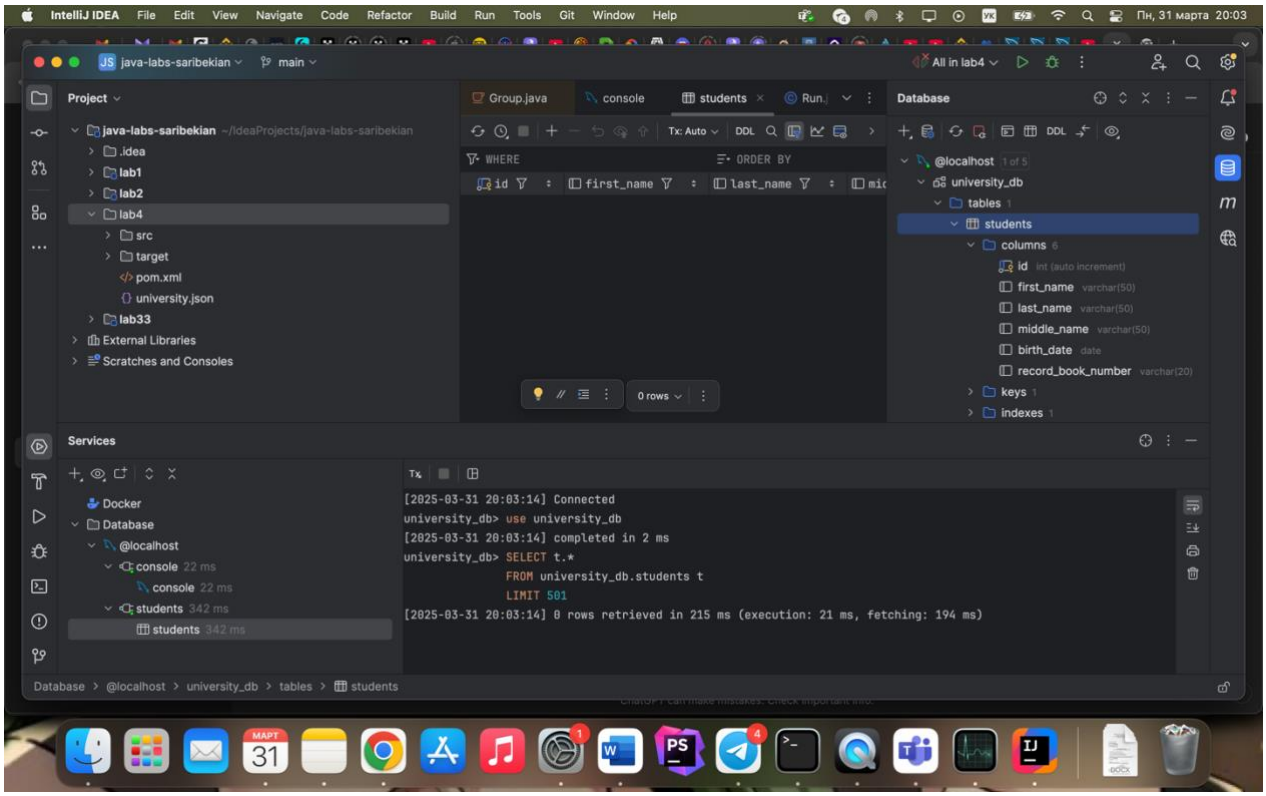
До паперового звіту обов'язково додати принтскрин з програми в якій ви дивитесь інформацію вашого сервера управління баз даних, де показати створену таблицю, її ім'я та загальні відомості бази даних, наприклад назва, ім'я, назва користувача адміністратора, пароль тощо. Для роботи з сервером управління 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

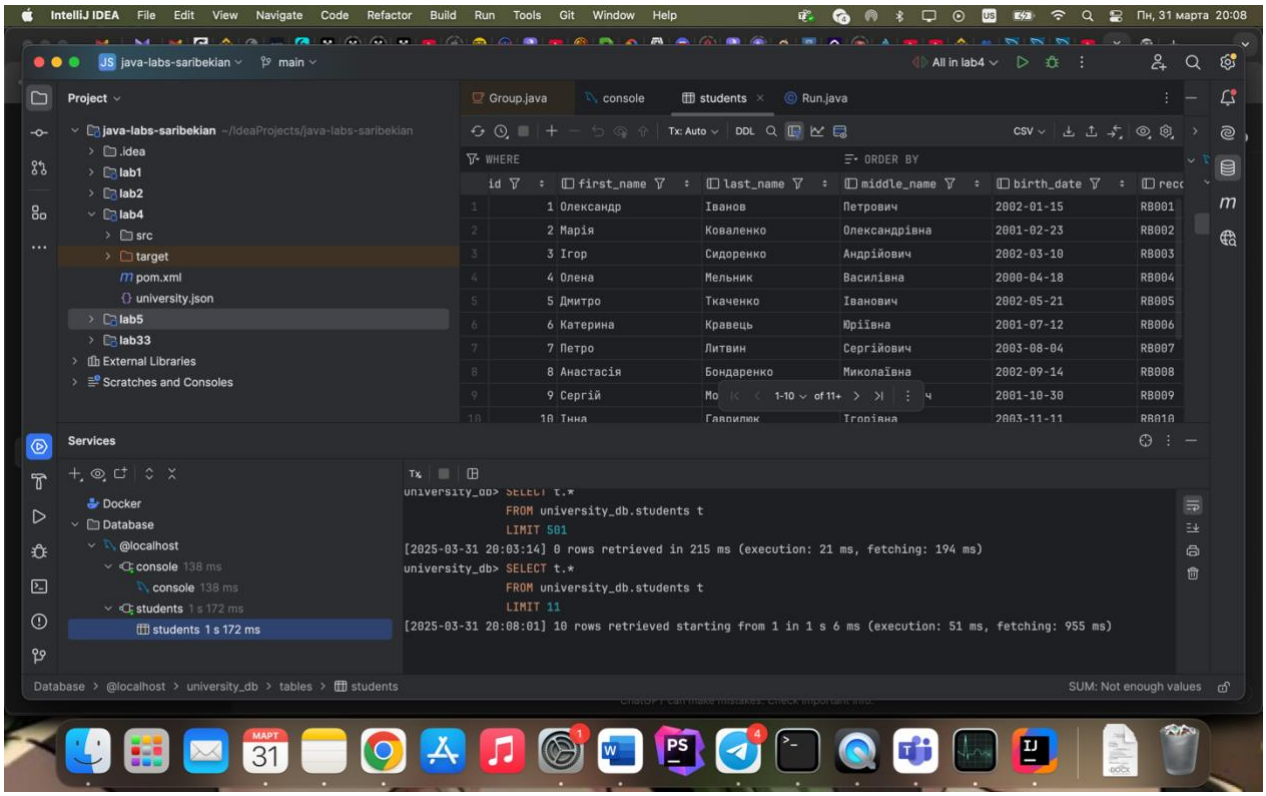
Для створення БД будемо використовувати вбудовану DataGrip у IDEA. Після запуску MySQL Server було створено нового користувача, базу даних та перевірено підключення.



Створюємо university_db:



Заповненна таблиця students:



Run.java

Запускає консольний інтерфейс, виводить всіх студентів, запитує у користувача номер місяця, і показує студентів, народжених у цьому місяці.

```java
package org.example;

import org.example.controller.StudentRepository;
import org.example.model.Student;

import java.util.List;
import java.util.Scanner;

public class Run {
    public static void main(String[] args) {
        StudentRepository repo = new StudentRepository();

        System.out.println("== Всі студенти ==");
        repo.getAllStudents().forEach(System.out::println);

        Scanner scanner = new Scanner(System.in);
        System.out.print("\nВведіть номер місяця (1-12), щоб вивести
студентів, народжених у цьому місяці: ");

        int month;
        while (true) {
            try {
                month = Integer.parseInt(scanner.nextLine());
                if (month >= 1 && month <= 12) {
                    break;
                } else {
                    System.out.print("Будь ласка, введіть число від 1 до 12:
");
                }
            } catch (NumberFormatException e) {
                System.out.print("Некоректне значення. Спробуйте ще раз: ");
            }
        }

        List<Student> studentsByMonth = repo.getStudentsByBirthMonth(month);

        if (studentsByMonth.isEmpty()) {
            System.out.println("\nСтудентів, народжених у " +
getMonthName(month) + ", не знайдено.");
        } else {
            System.out.println("\n== Студенти, народжені в " +
getMonthName(month) + " ==");
            studentsByMonth.forEach(System.out::println);
        }

        scanner.close();
    }

    private static String getMonthName(int month) {
        return switch (month) {
            case 1 -> "січні";
            case 2 -> "лютому";
            case 3 -> "березні";
            case 4 -> "квітні";
            case 5 -> "травні";
            case 6 -> "червні";
            case 7 -> "липні";
            case 8 -> "серпні";
            case 9 -> "вересні";
            case 10 -> "жовтні";
            case 11 -> "листопаді";
            case 12 -> "грудні";
            default -> "невідомому місяці";
```

```
        };
    }
}
```

Student.java

Представляє сутність студента з полями: ім’я, прізвище, по батькові, дата народження, номер залікової.

```
package org.example.model;

import java.time.LocalDate;

public class Student {
    private int id;
    private String firstName;
    private String lastName;
    private String middleName;
    private LocalDate birthDate;
    private String recordBookNumber;

    public Student(int id, String firstName, String lastName, String
middleName, LocalDate birthDate, String recordBookNumber) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.middleName = middleName;
        this.birthDate = birthDate;
        this.recordBookNumber = recordBookNumber;
    }

    @Override
    public String toString() {
        return firstName + " " + middleName + " " + lastName + " (" +
birthDate + ")";
    }

}
```

StudentRepository.java

Містить методи getAllStudents() та getStudentsByBirthMonth(int month) для отримання студентів з БД через JDBC.

```
package org.example.controller;

import org.example.model.Student;
import org.example.util.DatabaseConnection;

import java.sql.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class StudentRepository {

    public List<Student> getAllStudents() {
        List<Student> students = new ArrayList<>();
        String query = "SELECT * FROM students";
```

```
        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {

            while (rs.next()) {
                students.add(mapResultSetToStudent(rs));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return students;
    }

    public List<Student> getStudentsByBirthMonth(int month) {
        List<Student> students = new ArrayList<>();
        String query = "SELECT * FROM students WHERE MONTH(birth_date) = ?";

        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(query)) {

            stmt.setInt(1, month);
            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                students.add(mapResultSetToStudent(rs));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }

        return students;
    }

    private Student mapResultSetToStudent(ResultSet rs) throws SQLException {
        return new Student(
                rs.getInt("id"),
                rs.getString("first_name"),
                rs.getString("last_name"),
                rs.getString("middle_name"),
                rs.getDate("birth_date").toLocalDate(),
                rs.getString("record_book_number")
        );
    }
}
```

DatabaseConnection.java

Містить метод getConnection() для встановлення з’єднання з базою даних MySQL.

```
package org.example.util;

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

public class DatabaseConnection {
    private static final String URL =
"jdbc:mysql://localhost:3306/university_db";
    private static final String USER = "root";
    private static final String PASSWORD = "root1234";
```

```
    public static Connection getConnection() {
        try {
            // Явне завантаження драйвера
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

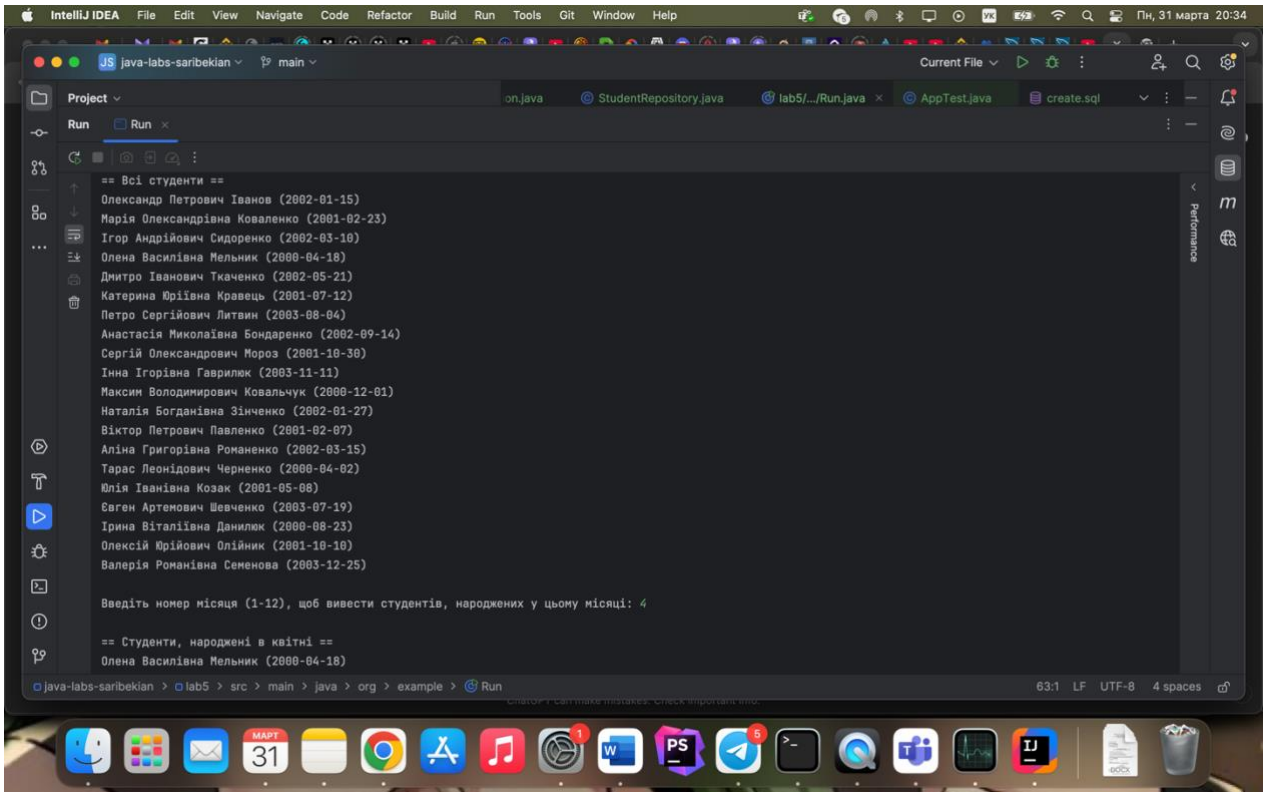
create.sql

Файл для ініціалізації бази даних: створення таблиці students з усіма необхідними полями.

```
CREATE DATABASE IF NOT EXISTS university_db;
USE university_db;

CREATE TABLE IF NOT EXISTS students (
                                    id INT PRIMARY KEY AUTO_INCREMENT,
                                    first_name VARCHAR(50),
                                    last_name VARCHAR(50),
                                    middle_name VARCHAR(50),
                                    birth_date DATE,
                                    record_book_number VARCHAR(20)
);
```

Результат програми:



Висновок

У цій лабораторній роботі було використано підхід, за якого SQL-запит формується відповідно до дій користувача — тобто пошук студентів, народжених у певному місяці, виконується безпосередньо на рівні бази даних.

Такий варіант був обраний з кількох причин. По-перше, це дозволяє не витягувати з бази всі дані, а працювати тільки з тими, що справді потрібні. Це зменшує навантаження на мережу (у випадку віддаленої БД) і знижує використання пам’яті в Java-програмі. По-друге, фільтрація на рівні БД зазвичай виконується швидше, ніж у коді, оскільки СУБД оптимізовані для таких операцій.

Альтернативний підхід — коли спочатку отримуються всі записи, а вже потім фільтруються в програмі — був би простішим у реалізації, але менш ефективним. Він міг би бути виправданим лише у випадках, коли потрібно обробити невелику кількість даних або коли немає можливості передати складні запити до БД.

Загалом, вибраний підхід краще підходить для подібних завдань, особливо якщо обсяг даних у майбутньому зросте.