

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

**Курсова робота**

**з дисципліни «Об'єктно-орієнтоване програмування»**

**на тему: «Інформаційна база навчального закладу»**

Виконав:  
студент гр.ПЗ1911  
Сафонов Д. Є.  
Прийняла:  
Демидович І. М.

Дніпро, 2021

## **Завдання на курсову роботу**

## Зміст

1. Опис предметної області та розробка специфікацій.....	5
1.1. Постановка задачі.....	5
1.2. Вимоги до програми.....	7
1.3. Методологія програмування.....	7
1.4. Середовище розробки і мова програмування.....	7
1.5. Вимоги до вхідних даних.....	7
1.6. Вимоги до вихідних даних.....	7
1.7. Вимоги до функціональності.....	7
2. Основна частина.....	8
2.1. Розробка об'єктно-орієнтованої моделі.....	8
2.2. Розробка та опис інтерфейсної частини класів.....	9
2.3. Розробка файлів реалізації класів.....	10
2.4. Тестування програми.....	11
2.5. Приклад роботи програми.....	12
2.6. Аналіз результатів.....	13
3. Висновки.....	14
4. Література.....	15
5. Додатки.....	16

## Вступ

Об'єктно-орієнтоване програмування(ООП) — парадигма програмування, яка базується на концепті “об'єктів”, які можуть складатися із коду та даних(стану): даних у формі полів(атрибутів), та коду у вигляді процедур(методів).

Одна з особливостей об'єктів — можливість змінювати свій стан у процесі виконання методів(mutability), це виконується за допомогою неявної передачі посилання на об'єкт через аргумент this або self(в залежності від використаної мови програмування). Насправді коли ми пишемо

```
var object = new Class();  
object.method();
```

Це те саме що:

```
Class::method(this=object);
```

Програми створені із допомогою об'єктно-орієнтованих мов програмування розроблюються як об'єкти, які взаємодіють між собою. Також ОО мови можуть бути базовані на прототипах, класах, акторах. Більшість мов базовані на класах, але існують популярні мови базовані на прототипах(наприклад ECMAScript(JavaScript)).

Більшість популярних об'єктно-орієнтованих мов програмування не є чистими, а складаються із декількох парадигм, найчастіше з: імперативною, процедурною, функціональною.

Об'єктно-орієнтовані мови програмування мають спільні риси із не ОО мовами: змінні, які можуть зберігати примітивні типи даних або об'єкти, процедури, які називаються методами, але виконують ту саму функцію(з цієї точки зору можна вважати що програма написана на імперативній мові програмування є класом, а якщо запустити її — отримаємо об'єкт, викликавши його процедури ми можемо змінити його стан, або отримати його). Інкапсуляція також не є рисою тільки об'єктно-орієнтованих мов, вона є мова програмування, які мають концепт модуля(декілька процедур, констант тощо можна об'єднати в один модуль та зробити деякі з цих об'єктів приватними(не у тому самому сенсі що в ООП)). Поліморфізм також існує в інших парадигмах, і не тільки у тому вигляді що в ООП(поліморфізм підтипів).

Через дуже велику популярність об'єктно-орієнтованого підходу у розробці програм були виділені практики, які можна використовувати у дуже різних випадках для покращення коду(з точки зору читабельності, розширюваності тощо), річ про патерни розробки, у 1994 навіть вийшла книга “Шаблони проектування: Елементи повторно використовованого об'єктно-орієнтованого програмного забезпечення”, автори книги: Еріх Гамма (англ. Erich Gamma), Річард Хелм (англ. Richard Helm), Ральф Джонсон (англ. Ralph Johnson), Джон Вліссідес (англ. John Vlissides). Колектив авторів також відомий як «Банда чотирьох» (англ. Gang of Four; GoF). Книга описує 23 популярних проблеми та патерни, які допомагають їх вирішити.

# 1. Опис предметної області та розробка специфікацій

## 1.1. Постановка задачі

Основною задачею курсового проекту є розробка програми, яка допомогла б з організацією навчального процесу, а саме програма має надавати:

- розклад занять
- інформацію про предмети
- інформацію про групи/класи, студентів, викладачів

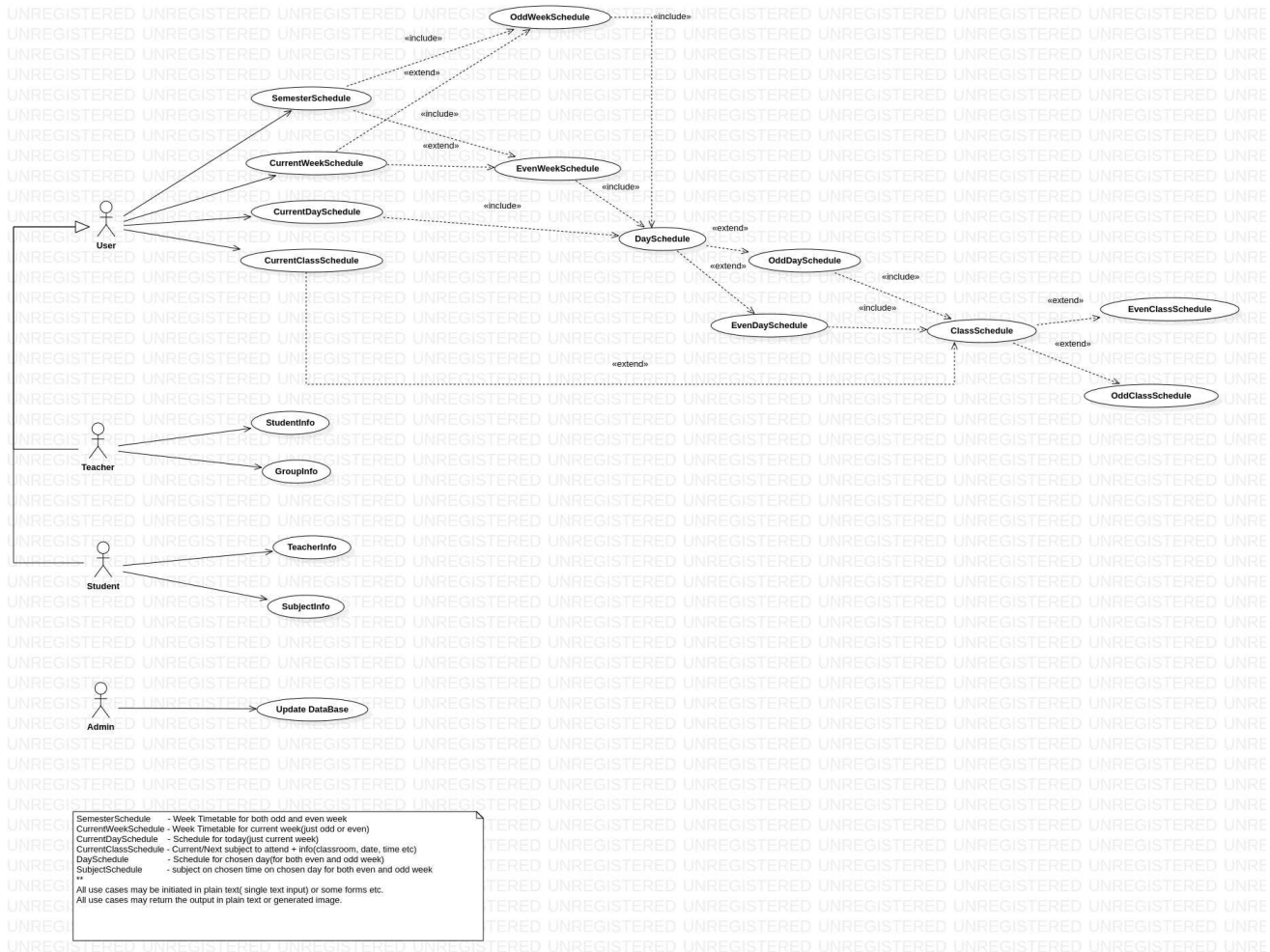


Рисунок 1: Діаграма usecase

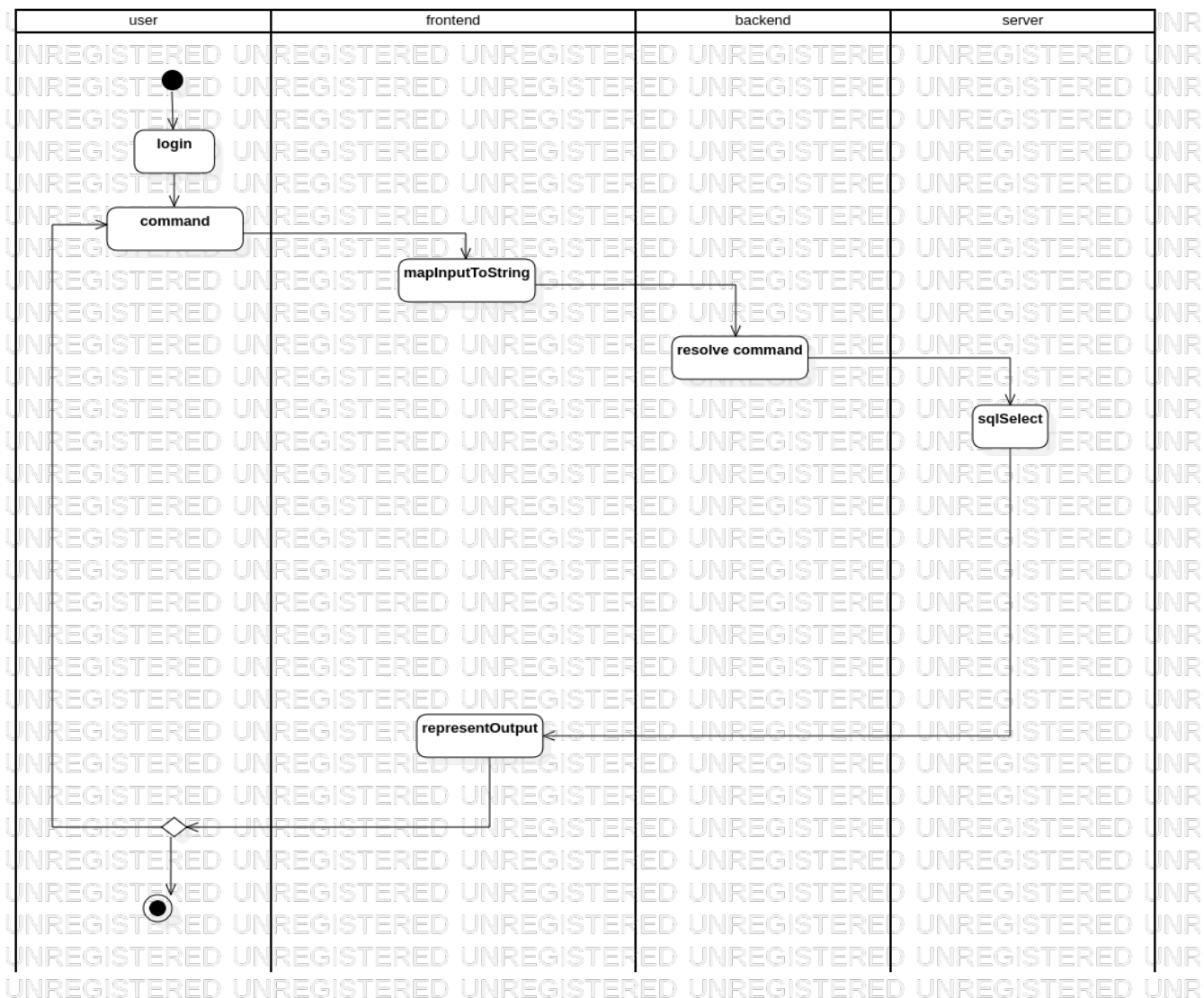


Рисунок 2: Діаграма activity

## **1.2. Вимоги до програми**

Програма має задовільняти наступні вимоги:

- Стійкість до некоректних команд
- Підтримка різних фронтендів
- Система адміністрації

## **1.3. Методологія програмування**

Об'єктно-орієнтоване програмування(ООП) — підхід у програмуванні, заснований на поєднанні даних та коду у об'єктах, і створенні програми із таких об'єктів.

## **1.4. Середовище розробки і мова програмування**

Курсовий проект був написаний із використанням мов TBD(to be determined), SQL у текстовому редакторі vscode(Visual Studio Code).

Ця мова була обрана....

Вибір середовища був з двох текстових редакторів: vim та vscode, і хоч перший досить швидкий, він є дещо складнішим у роботі з відносно великими проєктами.

## **1.5. Вимоги до вхідних даних**

Вхідні дані мають подаватися у вигляді текстових команд, але із допомогою фронтенду кінцевий користувач не обов'язково має вводити їх у такому вигляді.

## **1.6. Вимоги до вихідних даних**

Вихідні дані будуть подаватися у вигляді об'єктів, які будуть трансформовані у текст або картинки.

## **1.7. Вимоги до функціональності**

Програма має дозволяти:

- Подивитися розклад занять:
  - поточний(яка пара/урок наступний)
  - на сьогодні(має змінюватися щодня та щотижня(чисельник/знаменник))
  - на тиждень(можна обрати чисельник/знаменник або обидва)
- Подивитися інформацію про предмет:
  - Викладачів
  - Групи або класи, які вивчають цей предмети
- Подивитися інформацію про викладачів та студентів

## 2. Основна частина

### 2.1. Розробка об'єктно-орієнтованої моделі.

Опис відповідальності класів - для кожного класу треба чітко описати його відповідальність.

Backend – відповідає за розпізнавання текстових команд та повернення відповідних об'єктів-відповідей.

SqlDb – відповідає за маніпуляції над базою даних.

Frontend – відповідає за отримання вхідних даних від користувача та перетворення вихідних у відповідний вигляд.

Опис відношень між класами – описати як класи взаємодіють один з одним, вказати типи зв'язків між класами.

class diagram

sequence diagram



## **2.2. Розробка та опис інтерфейсної частини класів.**

В дані частині для кожного класу треба описати всі його поля (вказати назву, тип, діапазон можливих значень, призначення поля), методи (призначення методу, тип значення, що повертається, параметри з вказанням назви, типу та діапазону можливих значень).

### **2.3. Розробка файлів реалізації класів.**

Проектування основних алгоритмів. Методи класів реалізують певні алгоритми. В даному розділі треба навести основні алгоритми, які використовувались при розробці методів. Алгоритм можна представити у вигляді схеми Нассі або блок-схеми. Бажано використовувати деталізацію при представлення алгоритмів, а також коментувати основні змінні які в них використовуються (призначення). Бажано користуватись графічним редактором для побудови блок-схем чи схем Нассі алгоритмів.

## **2.4. Тестування програми.**

В даному розділі необхідно показати як виконувалось тестування програми на прикладі декількох методів класу. Обов'язково виконати тестування методами і чорної, і білої скриньки. Необхідно обґрунтувати методи тестування, які були обрані, представити тести та таблиці покриття(для методів тестування білою скринькою) та навести результати тестування.

## **2.5. Приклад роботи програми.**

Необхідно навести основні моменти роботи програми - введення вхідних даних, обробка та представити результат роботи програми.

## **2.6. Аналіз результатів.**

Переваги програми за рахунок застосування об'єктно-орієнтованої парадигми.

### **3. Висновки**

окрема сторінка

#### **4. Література**

На кожен пункт списку літератури має бути посилання в тексті курсової роботи

## **5. Додатки**

В додатках розміщують довідкові матеріали, деталі розрахунків, лістинги програм