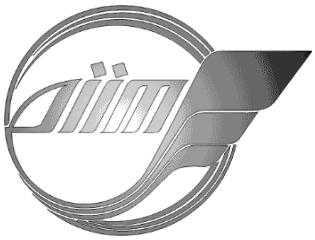
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Міністерство освіти і науки України
Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет: Комп’ютерні технології і системи
Кафедра: Комп’ютерні інформаційні технології
Спеціальність: 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Зав. кафедрою "КІТ"
проф. Шинкаренко В.І.
"____"_____2021р.

З А В Д А Н Н Я
до курсової роботи

студента Сафонова Д. Є.

1. Тема проекту: Інформаційна база навчального закладу
2. Дата видачі завдання _____
3. Перелік питань до розробки:
    1. Опис предметної області та розробка специфікацій (вимог до програми).
        1.1. Постановка задачі.
        1.2. Вимоги до програми.
    2. Розробка об’єктно-орієнтованої моделі.
        2.1. Опис відповідальності класів.
        2.2. Опис відношень між класами.
        2.3. Діаграма класів.
        2.4. Діаграма послідовностей.
    3. Розробка та опис інтерфейсної частини класів.
    4. Розробка файлів реалізації класів. Проектування основних алгоритмів.
    5. Тестування програми.
    6. Відлагодження програми.
    7. Приклад роботи програми.
    8. Аналіз результатів. Переваги програми за рахунок застосування об’єктно-орієнтованої парадигми.
    9. Текст програми.
4. Термін виконання курсового проекту 31 травня 2021 року

Керівник курсового проектування Демидович І. М.
Завдання прийняв до виконання Сафонов Д. Є.

Зміст

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 викликаємо метод об’єкту деякого класу, це те саме що викликати функцію із модуля, де один з аргументів буде структурою із набором атрибутів аналогічним цьому класу.

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 проєктування: Елементи повторно використовуваного 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

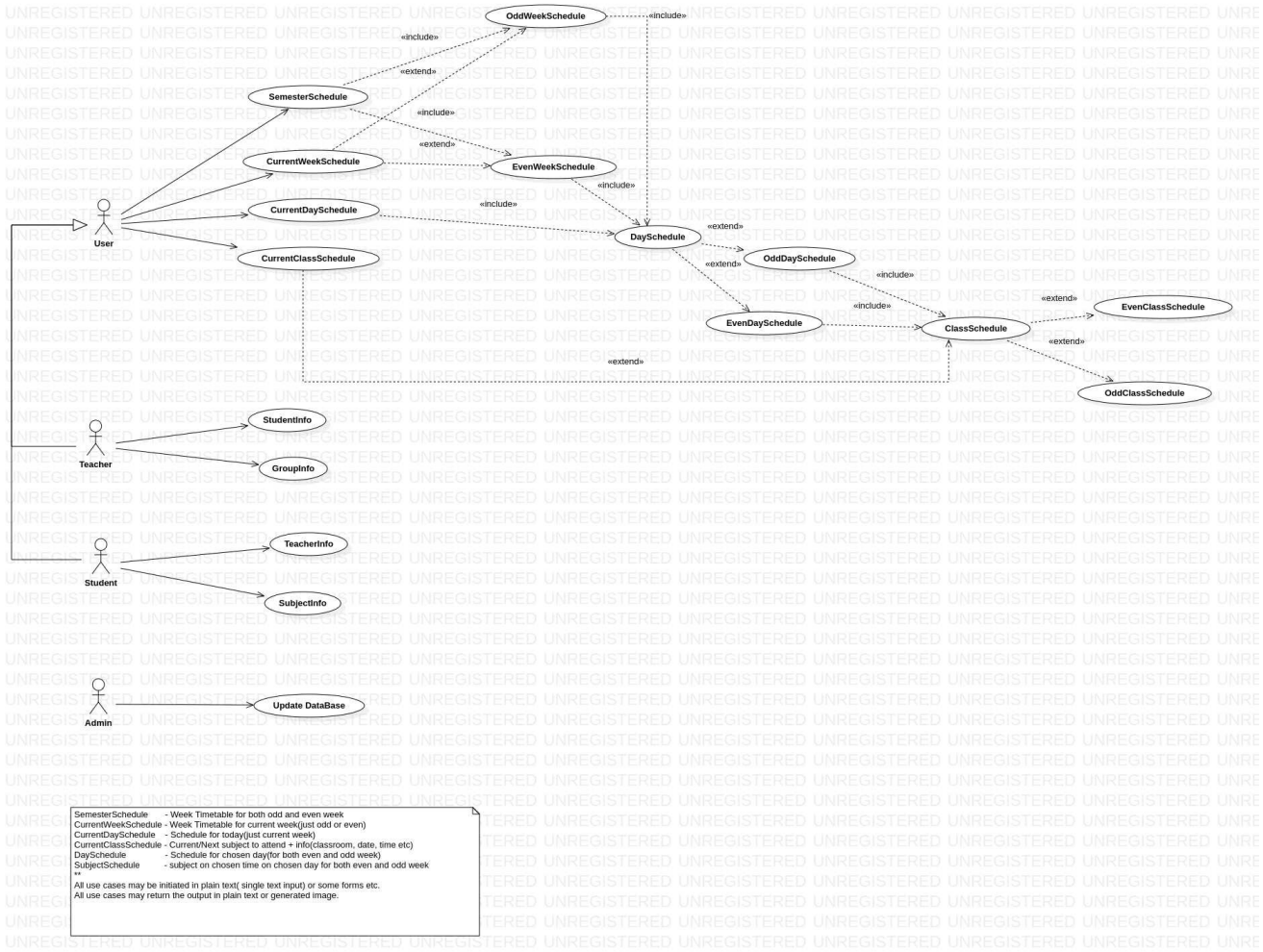


Рисунок 1: Діаграма usecase

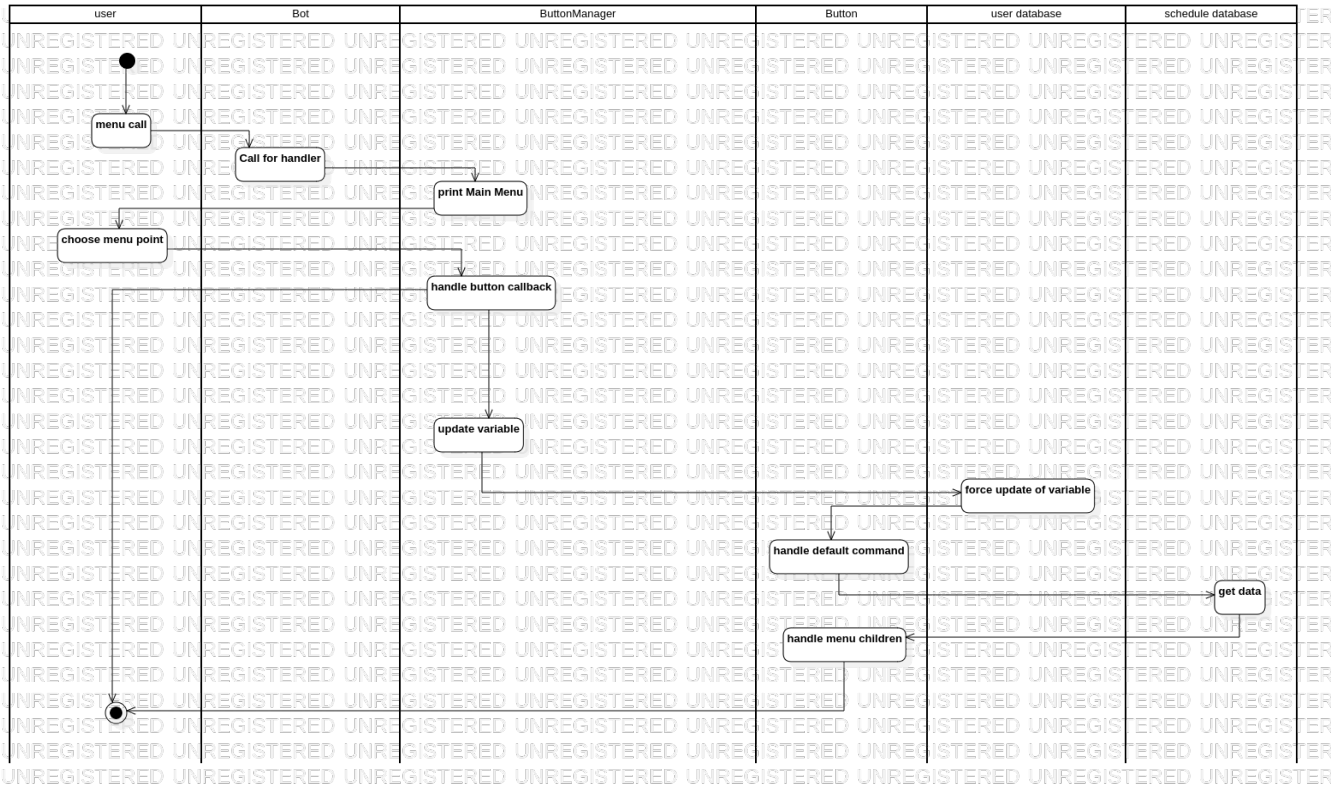


Рисунок 2: Діаграма activity(для одного циклу програми)

1.2. Вимоги до програми

Програма має задовільняти наступні вимоги:

- Стійкість до некоректних команд
- Модульність
- Стандартизація

1.3. Методологія програмування

Об’єктно-орієнтоване програмування(ООП) — підхід у програмуванні, заснований на поєднані 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 текстовик команд, але із допомогою фронт-енд 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. Розробка об’єктно-орієнтованої моделі

2.1. Опис розподілу відповідальностей між класами та зв’язків між ними

Клас	Відповідальність	Зв'язки
Button	Виконання дії після натискання на кнопку	Спадкування: **LeafButton**(Нащадок) **Menu**(Нащадок) Агрегація: **ButtonManager**(Button е складовою ButtonManager) Агрегація: **Menu**(Button е складовою Menu)
LeafButton	Виведення результату після натискання на кнопку	Спадкування: **Button**(Батьківський клас) Агрегація: **Menu**(Menu може мати LeafButton в собі)
Menu	Виведення нового меню після натискання на кнопку	Спадкування: **Button**(Батьківський клас) Агрегація: **LeafButton**(Menu може мати LeafButton в собі) **Menu**(Menu може мати Menu в собі)
ButtonManager	Складне меню на основі Menu, LeafButton, Button	Композиція: **Button**(ButtonManager може мати в собі LeafButton та/або Menu) **Schedule**(Schedule е складовою ButtonManager) **UserDbManager**(UserDbManager е складовою ButtonManager) Залежність: **Bot**(Bot залежить від ButtonManager)

Bot	Головний клас, який контролює взаємодію з користувачем	Залежність: **ButtonManager**(Bot залежить від ButtonManager) Агрегація: **Command**(Command є складовою Bot)
Command	Асоціація ключового слова із обробником та описом команди	Агрегація: **Bot**(Command є складовою Bot)
UserDbManager	Зручний інтерфейс до бази даних користувачів	Агрегація: **ButtonManager**(UserDbManager є складовою ButtonManager) Залежність: **User**(UserDbManager залежить від User)
User	Зберігання стану користувача	Залежність: **User**(UserDbManager залежить від User)
Schedule	Зручний інтерфейс до бази даних розкладу	Композиція: **Schedule**(Schedule є складовою ButtonManager)
MultiPageMenu	Багатосторінкове меню	Спадкування: **Menu**(Батьківський клас) **CalendarMenu**(Нащадок) **MultiPageListMenu**(Нащадок)
ListMenu	Односторінкове меню зі списку опцій	Спадкування: **Menu**(Батьківський клас)
MultiPageListMenu	Багатосторінкове меню зі списку опцій	Спадкування: **MultiPageMenu**(Батьківський клас)
CalendarMenu	Багатосторінкове меню-календар	Спадкування: **MultiPageMenu**(Батьківський клас)

2.2. Діаграма класів

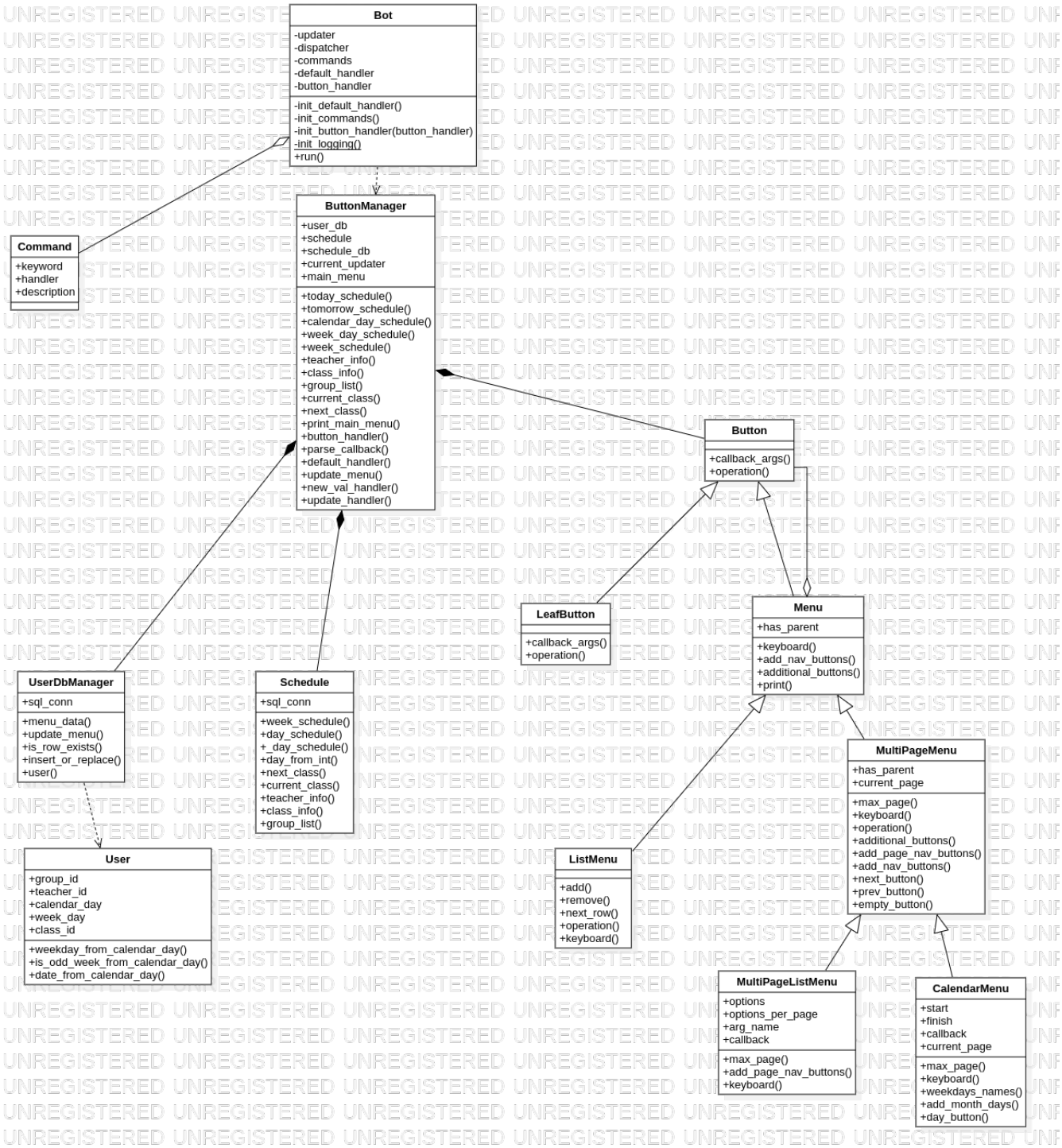


Рисунок 3: Діаграма класів

2.3. Діаграма послідовності

TODO

3. 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).

TODO

4. 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.

TODO

5. 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.

TODO

6. Приклад роботи програми.

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

TODO

7. Аналіз результатів.

Переваги програми за рахунок застосування об’єктно-орієнтованої парадигми.
TODO

8. Висновки

окрема сторінка
TODO

9. Література

На кожну позицію списку літератури має бути посилання в тексті курсової роботи
TODO

10. Додатки

В додатках розміщують довідкові матеріали, деталі розрахунків, лістинги програм
TODO