МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Лабораторна робота №2

з дисципліни

«Системний аналіз»

Виконав:

Студент групи КН-214

Добрій Назарій

Викладач:

Зімоха.І.О.

Львів – 2021р.

Завдання :

Реалізувати патерн «Спостерігач» на основі предметної області.

Кроки реалізації

1. Розбийте вашу функціональність на дві частини: незалежне ядро та опціональні залежні частини. Незалежне ядро стане видавцем. Залежні частини стануть підписниками.
2. Створіть інтерфейс підписників. Зазвичай достатньо визначити в ньому лише один метод сповіщення.
3. Створіть інтерфейс видавців та опишіть у ньому операції керування підпискою. Пам’ятайте, що видавці повинні працювати з підписниками тільки через їхній загальний інтерфейс.
4. Вам потрібно вирішити, куди помістити код ведення підписки, адже він зазвичай буває однаковим для всіх типів видавців. Найочевидніший спосіб — це винесення коду до проміжного абстрактного класу, від якого будуть успадковуватися всі видавці. Якщо ж ви інтегруєте патерн до існуючих класів, то створити новий базовий клас може бути важко. У цьому випадку ви можете помістити логіку підписки в допоміжний об’єкт та делегувати йому роботу з видавцями.
5. Створіть класи конкретних видавців. Реалізуйте їх таким чином, щоб після кожної зміні стану вони слали сповіщення всім своїм підписникам.
6. Реалізуйте метод сповіщення в конкретних підписниках. Не забудьте передбачити параметри, через які видавець міг би відправляти якісь дані, пов’язані з подією, що відбулась

Предметна область:

№ вар	Предметна область
5	Видання книг

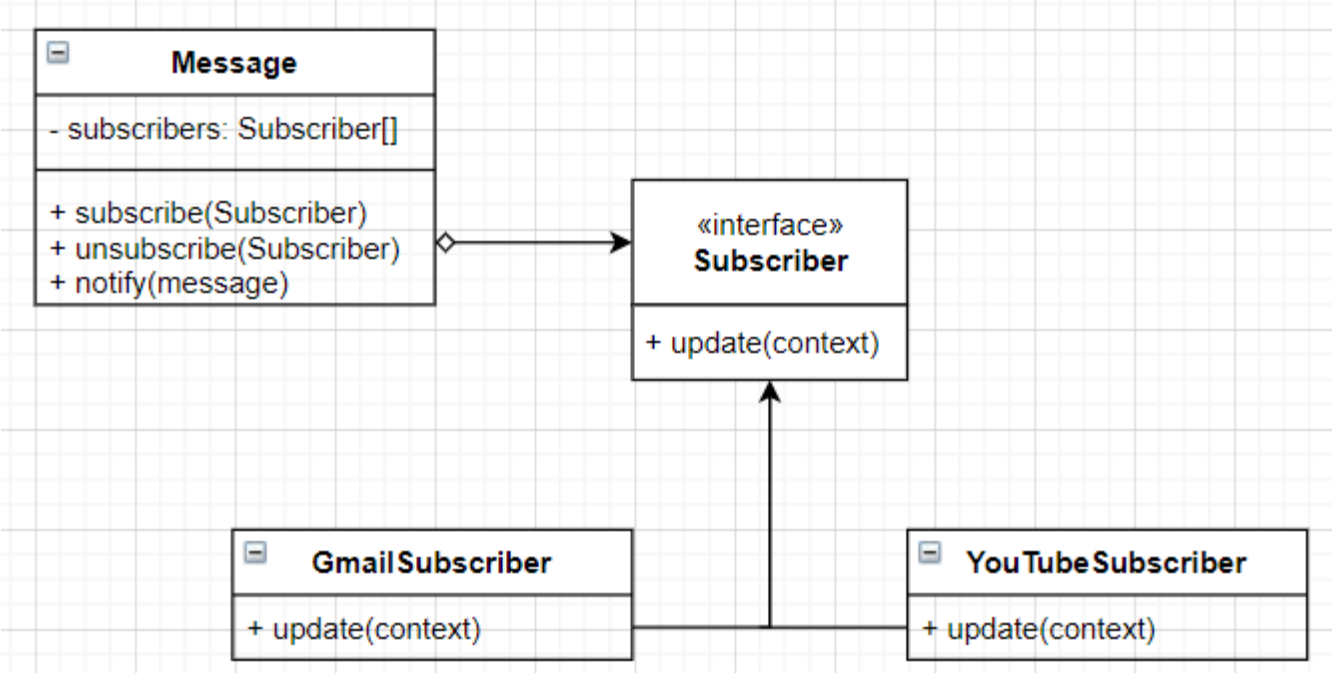

Рис. 1. UML-діаграма патерна “Спостерігач”

Код програми на мові Python:

```
from abc import ABC, abstractmethod


class Subscriber(ABC):
    @abstractmethod
    def update(self, message):
        pass


class YouTubeSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name

    def update(self, message):
        print('Користувач {} отримав повідомлення "{}" через
YouTube'.format(self.name, message))


class GmailSubscriber(Subscriber):
    def __init__(self, name):
        self.name = name

    def update(self, message):
        print('Користувач {} отримав повідомлення "{}" через пошту
gmail.com'.format(self.name, message))


class Notificator:
    def __init__(self):
        self.subscribers = set()

    def subscribe(self, sub: Subscriber):
        self.subscribers.add(sub)
        print('{} підписався'.format(sub.name))
```

```
    def unsubscribe(self, sub):
        self.subscribers.discard(sub)
        print('{} відписався'.format(sub.name))

    def notify(self, message):
        for subs in self.subscribers:
            subs.update(message)


notifiactor = Notificator()

sub1 = GmailSubscriber("nazardobriy@gmail.com")
sub2 = YouTubeSubscriber("Nazar Dobriy")

notifiactor.subscribe(sub1)
notifiactor.subscribe(sub2)

notifiactor.notify("Виконання підписки продовжено на місяць!")

notifiactor.unsubscribe(sub2)

notifiactor.notify("Успішна підписка на місяць!")
```

Рис. 2.1. Код реалізації програми.

```
nazardobriy@gmail.com підписався
Nazar Dobriy підписався
Користувач Nazar Dobriy отримав повідомлення "Виконання підписки продовжено на місяць!" через YouTube
Користувач nazardobriy@gmail.com отримав повідомлення "Виконання підписки продовжено на місяць!" через пошту gmail.com
Nazar Dobriy відписався
Користувач nazardobriy@gmail.com отримав повідомлення "Успішна підписка на місяць!" через пошту gmail.com

Process finished with exit code 0
```

Рис. 2.2. Виконання програми.

Висновок: я навчився реалізовувати патерн “Спостерігач” на основі предметної області видання книг, добавивши клієнтам підписку на пошту та youtube.