Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 6

з дисципліни «Теорія розробки ПЗ»

Тема: ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Варіант №3

Виконав:
студент групи ІА-13
Губенко Єгор Олександрович

Київ, 2023

Тема: ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator».

Завдання.

- Ознайомитися з короткими теоретичними відомостями.
- Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.
- Застосування одного з даних шаблонів при реалізації програми.

Варіант.

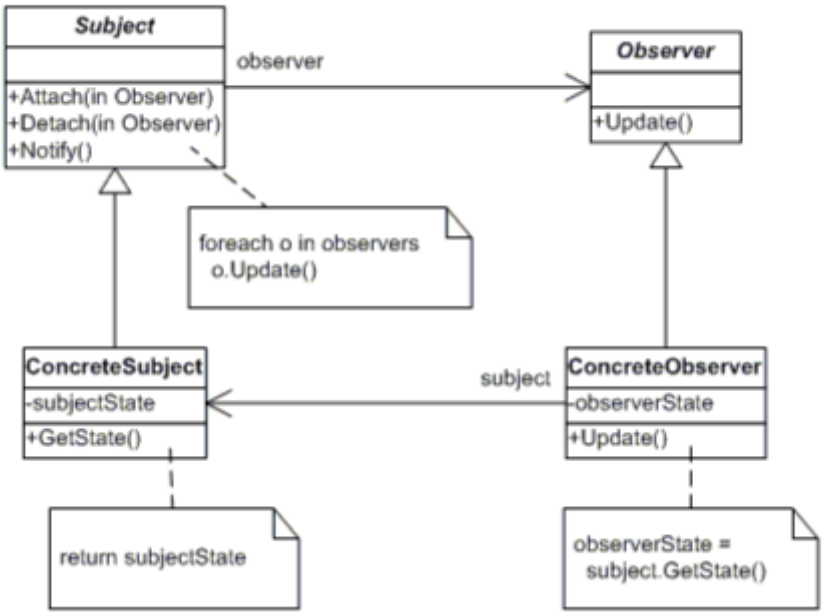
3. Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Хід роботи

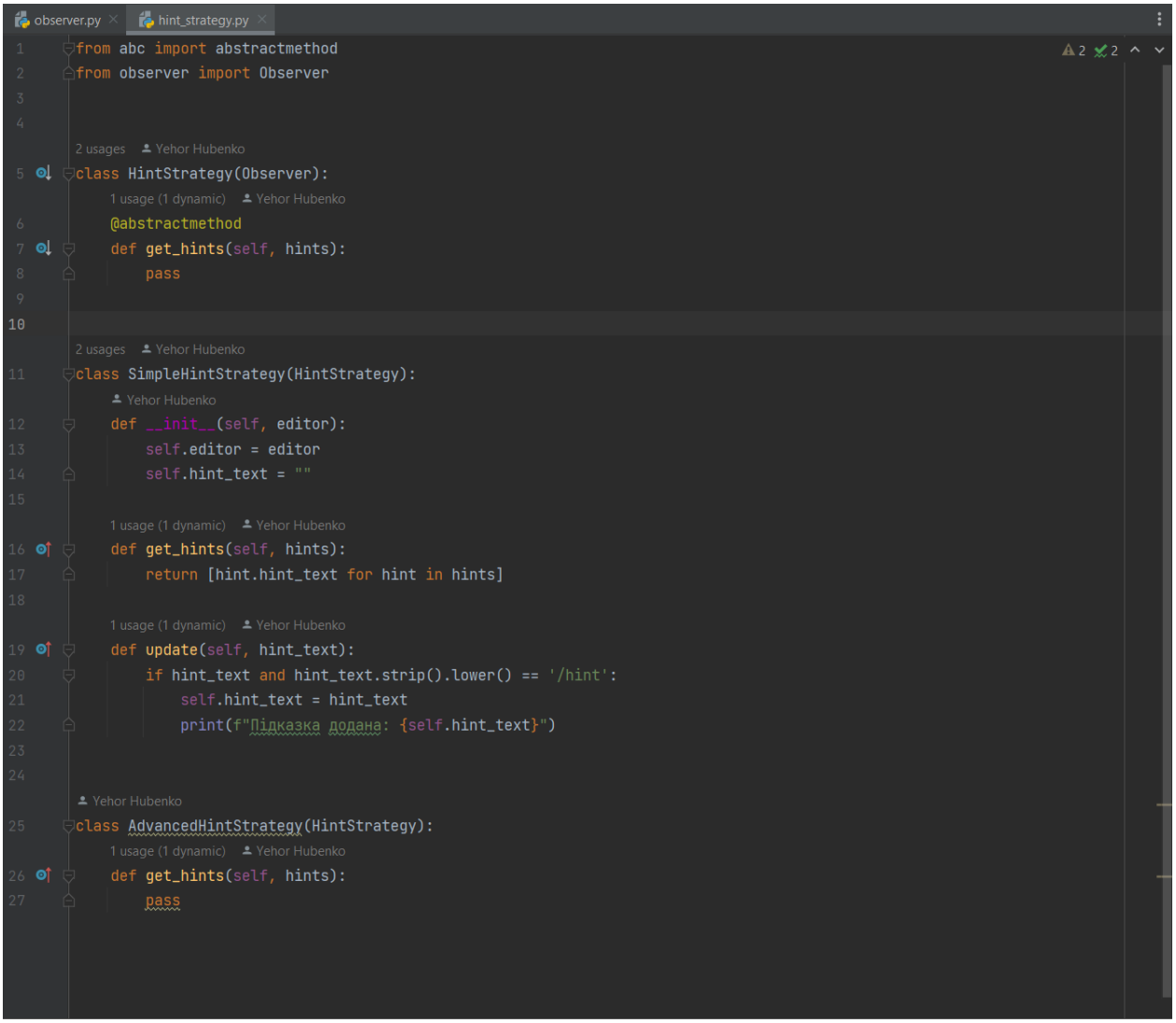
Шаблон «Observer»

Шаблон визначає залежність "один-до-багатьох" таким чином, що коли один об'єкт змінює власний стан, усі інші об'єкти отримують про це сповіщення і мають можливість змінити власний стан також.



observer.py

```python
import os
from abc import abstractmethod, ABC


4 usages  Yehor Hubenko
class Observer(ABC):
    1 usage (1 dynamic)  Yehor Hubenko
    @abstractmethod
    def update(self, hint_text):
        pass


Yehor Hubenko
class PathSubject:
    Yehor Hubenko
    def __init__(self):
        self.observers = []

    Yehor Hubenko
    def add_observer(self, observer):
        self.observers.append(observer)

    Yehor Hubenko
    def remove_observer(self, observer):
        self.observers.remove(observer)

    Yehor Hubenko
    def notify_observers(self, file_path):
        for observer in self.observers:
            observer.update(file_path)


3 usages  Yehor Hubenko
class EditorObserver(Observer):
    2 usages (1 dynamic)  Yehor Hubenko
    def update(self, changes):
        print(f"Editor received an update: {changes}")

    Yehor Hubenko
    def start_edit_mode(self):
        print("Entering edit mode. Type '/finish' to exit editing.")

    Yehor Hubenko
    def end_edit_mode(self):
        print("Exiting edit mode.")

    1 usage (1 dynamic)  Yehor Hubenko
    def open_file_by_path(self, file_path):
        print(f"Opening file by path: {file_path}")


2 usages  Yehor Hubenko
class PathObserver(Observer):
    Yehor Hubenko
    def __init__(self, editor):
        self.editor = editor

    1 usage (1 dynamic)  Yehor Hubenko
    def update(self, file_path):
        if file_path and os.path.exists(file_path):
            self.editor.open_file_by_path(file_path)
```



Висновок: у даній лабораторній я додав декілька класів до свого проєкту, використовуючи паттерн Observer.