



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8
З дисципліни «Технології розроблення програмного
забезпечення»
Тема: «ШАБЛОНИ «COMPOSITE», «FLYWEIGHT»,
«INTERPRETER», «VISITOR»»
Варіант №6

Виконав
студент групи ІА–13:
Костенко П.С.

Перевірив:
Мягкий М. Ю.

Київ 2023

Тема:

..6 Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p)

Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) - переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Завдання:

1. Реалізувати не менше 3х класів згідно з вибраною темою.
2. Реалізувати один з розглянутих шаблонів по вибраній темі.

Хід роботи:

На даній лабораторній роботі мені необхідно було реалізувати шаблон Visitor. Його суть полягає в тому, що при роботі з різними елементами виникає потреба у схожих діях, але щоб не доповнювати класи різних елементів створюється інтерфейс Visitor, який визначає необхідні методи для роботи, та конкретні візитори, які виконують конкретну реалізацію функціональності з елементами в залежності від того, що необхідно зробити з елементами.

У процесі виконання лабораторної роботи було створено новий інтерфейс Visitor, який описує, з якими елементами буде робота та створює шаблони цих методів:

```
from abc import ABC, abstractmethod


2 usages
class Visitor(ABC):
    1 usage (1 dynamic)
    @abstractmethod
    def visit_tag(self, tag):
        pass

    1 usage (1 dynamic)
    @abstractmethod
    def visit_script(self, script):
        pass


    1 usage (1 dynamic)
    @abstractmethod
    def visit_image(self, image):
        pass
```

а також нові класи з методами виклику візителя:


```
class TagElement:
    def accept(self, visitor):
        visitor.visit_tag(self)
```

 #Клас елемента тег

```
class ScriptElement:
    def accept(self, visitor):
        visitor.visit_script(self)
```

 #Клас елемента скрипт

```
class ImageElement:
    def accept(self, visitor):
        visitor.visit_image(self)
```

 #Клас елемента зображення

та клас конкретного візителя:

```
from model.Visitor import Visitor
```

3 usages

```
class HTMLVisitor(Visitor):
```

2 usages (1 dynamic)

```
def visit_tag(self, tag):
```

```
    # Логіка обробки тегів
```

```
    pass
```

2 usages (1 dynamic)

```
def visit_script(self, script):
```

```
    # Логіка обробки скриптів
```

```
    pass
```

2 usages (1 dynamic)

```
def visit_image(self, image):
```

```
    # Логіка обробки зображень
```

```
    pass
```

у якому конкретно описується логіка обробки елементів.

Також у класі MainWindow було створено метод

```
def visit_html(self, html_content):
```

```
    search_tag = self.html_content(html_content)
```

```
    search_image = self.image_content(html_content)
```

```
    search_script = self.js_content(html_content)
```

```
    HTMLVisitor.HTMLVisitor.visit_tag(search_tag)
```

```
    HTMLVisitor.HTMLVisitor.visit_image(search_image)
```

```
    HTMLVisitor.HTMLVisitor.visit_script(search_script)
```

який реалізується тим, що при передачі в нього контенту викликаються методи для повертання конкретного контенту та вони ж передаються у методи візита для роботи з даними(тегами, зображеннями, скриптами).

Висновок: на цій лабораторній роботі я познайомився з шаблоном visitor method, засвоїв знання на практиці, продовжив розробку проекту.