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 №5

З дисципліни «**Технології розроблення програмного забезпечення**»

Тема: «**ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»**»

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:

Для мого варіанту необхідно реалізувати паттерн Chain of Responsibility. Він полягає у тому, що створюються методи, які мають здійснювати поступову обробку, але якщо вони не можуть виконати її за наявними даними, або ж необхідні дії були виконані, але потребуються додаткові, вони передають дані наступникові.

```
    def showHistoryuser(self, user):
        repository = Repository
                            for       names       in
Repository.Repository.exec(repository.Repository(
), dbname='WebBrowser', user='postgres',

password='1234',

host='127.0.0.1', port='5432', query='SELECT name
from users'):
            if user.name == names:
                if user.administrator == False:

Repository.Repository.exec(repository.Repository(
), dbname='WebBrowser', user='postgres',
```

```
password='1234',

host='127.0.0.1', port='5432',

query='SELECT name from pages, users, histories
WHERE '

'histories.user_id = users.id AND '

'users.page_id = pages.id AND'

'users.name = ${user}')

                else:

MainWindow.showhistoryadmin(user=user)
            else:
                print("Ви не зареєстровані")


    # Метод для перевірки, чи юзер зареєстрований
    def showHistoryadmin(self, user):
        repository = Repository
                            for       names       in
Repository.Repository.exec(repository.Repository(
), dbname='WebBrowser', user='postgres',

password='1234',

host='127.0.0.1', port='5432', query='SELECT name
from users'):
                    if  user.name  ==  names  and
user.administrator == True:

Repository.Repository.exec(repository.Repository(
), dbname='WebBrowser', user='postgres',

password='1234',
```

```
host='127.0.0.1', port='5432',

query='SELECT *, name, name from histories,
users, pages WHERE '

'histories.id_user = users.id AND
histories.page_id ='

'pages.id')


    # Метод для перевірки, чи юзер -
адміністратор
```

Для реалізації завдання було обрано випадок перегляду історії. Було додано прапор на адміністратора для користувачів, переглядати історію можуть тільки зареєстровані користувачі або ж адміністратори, які можуть переглядати усі наявні історії.

Було додано два методи - showHistoryuser та showHistoryadmin.

При спробі перегляду історії буде викликатись showHistoryuser та перевіряти, чи взагалі існує цей юзер у базі даних, тобто чи він зареєстрований. Якщо такий юзер є - буде виконано перевірку, чи він звичайний юзер. Якщо так, йому буде показано його історію. Якщо такого користувача нема - це не зареєстрований користувач та історію показано не буде, буде виведено попередження, що це не зареєстрований користувач. Якщо ж такий користувач існує, але флажок адміністратора не False - відбудеться виклик showHistoryadmin та здійснеться перевірка, чи це адміністратор, якщо так - буде виведено історію.

Висновок: на цій лабораторній роботі я познайомився з паттерном Chain of Responsibility, засвоїв знання на практиці, продовжив розробку проєкту.