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

ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»

Виконала
студентка групи ІА – 13:
Майданюк Анастасія

Перевірив:
Мягкий М.Ю.

Київ 2023

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії

для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми

Варіант:

..11 Web crawler (proxy, chain of responsibility, memento, template method, composite, p2p)

Веб-сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

Хід роботи

Ланцюжок відповідальності - це патерн проектування, який спрощує процес передачі запитів через серію обробників. Кожен елемент у цій послідовності має можливість або обробити запит самостійно, або передати його далі по ланцюгу.

Цей патерн застосовується для вирішення різноманітних запитів, особливо у ситуаціях аналізу даних з веб-сайтів. Кожен елемент ланцюга, званий обробником, має свої специфічні обов'язки та можливість переадресації запиту наступному обробнику.

Головний клас, зазвичай іменований як Handler, є абстрактним і включає метод обробки запитів, як правило, під назвою handle_request. Різні спадкоємці цього класу, такі як SalaryHandler і DateAndViewHandler, імплементують цей метод для обробки певних видів запитів. У випадку, коли обробник не в змозі

впоратись із запитом, він перенаправляє його до наступного елементу ланцюжка.

Головний клас JobScraper використовує ці обробники для специфічної обробки запитів, наприклад, пов'язаних зі збором інформації про зарплату та дати публікацій.

```
class Handler:
    def __init__(self, successor=None):
        self.successor = successor

    def handle_request(self, soup):
        if self.successor:
            return self.successor.handle_request(soup)
        return None


class SalaryHandler(Handler):
    def handle_request(self, soup):
        salary_tag = soup.find('span', class_='public-salary-item')
        if salary_tag:
            return salary_tag.text.strip()
        return super().handle_request(soup)


class DateAndViewHandler(Handler):
    def handle_request(self, soup):
        info_tag = soup.find('p', class_='text-muted')
        if info_tag:
            date_match = re.search(r'опублікована\s+(\d+\s+\S+\s+\d+)', str(info_tag))
            date = date_match.group(1) if date_match else None

            views_match = re.search(r'(\d+)\s+перегляд', str(info_tag))
            views = views_match.group(1) if views_match else None

            responses_match = re.search(r'(\d+)\s+відгук', str(info_tag))
            responses = responses_match.group(1) if responses_match else None

            return {'date': date, 'views': views, 'responses': responses}
        return super().handle_request(soup)
```

Висновок:Під час виконання л/р я застосувала патерн "Ланцюжок відповідальності". Використання цього патерну дозволило створити чітку і гнучку структуру, де кожен обробник у ланцюжку виконує свої специфічні завдання, але при цьому може переадресовувати запит до наступного елемента ланцюжка, якщо він не в змозі його обробити. Це забезпечує високу адаптивність системи до різних сценаріїв використання.