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»

Варіант 11

Виконала
студентка групи ІА – 14:
Літвін Юлія

Перевірив:
Мягкий М. Ю.

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Варіант:

фіксації графічно.

..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

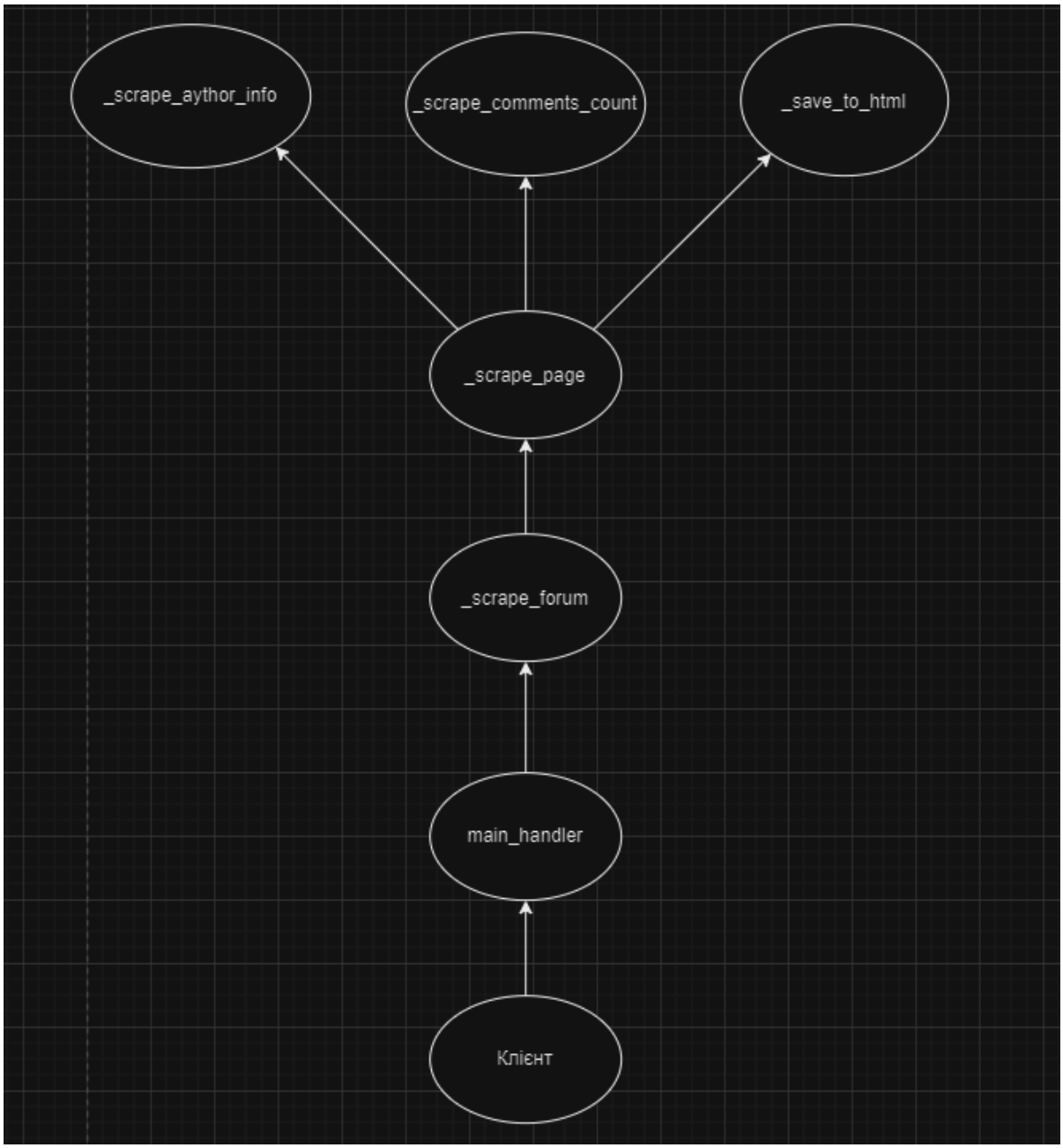
Паттерн "Chain of Responsibility" (ланцюг відповідальності) є паттерном проектування, який використовується для передачі запитів вздовж ланцюгу об'єктів обробки.

Обробники - це методи, які виконують конкретні завдання, пов'язані із веб-скрапінгом. Їх назви мають префікс _handler_, за яким слідує завдання, яке вони виконують.

Приклад обробників: _handler_scrape_comments_count, _handler_scrape_author_info, _handler_save_to_html, _handler_scrape_page і _handler_scrape_forum.

Обробник main_handler є вищорівневим, він викликає один із конкретних обробників на підсаві значення параметра handler.

Послідовність викликів в коді:



main_handler:

```
2 usages (1 dynamic)
def main_handler(self, handler, output_filename, tag, start_date=None, end_date=None):
    if handler == 'main':
        self._handler_scrape_forum( handler: '_scrape_forum', output_filename, tag, start_date=start_date, end_date=end_date)
```

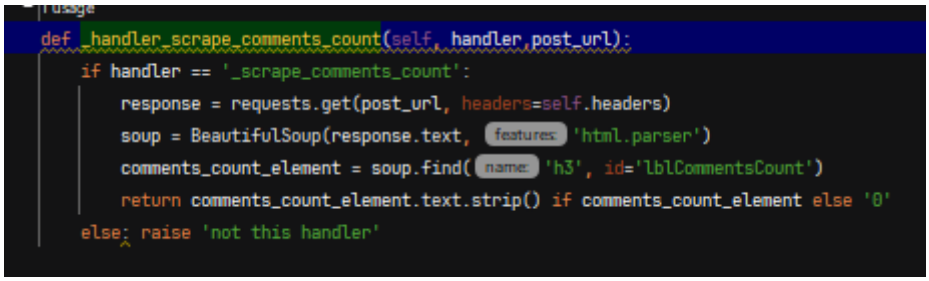
_handler_scrape_forum:

```python
def _handler_scrape_forum(self, handler,output_filename, tag, start_date=None, end_date=None):
    if handler == '_scrape_forum':
        result_file_path = os.path.join('../results', output_filename)
        with open(result_file_path, 'w', encoding='utf-8') as file:
            file.write('<html>\n<head>\n<title>Forum Data</title>\n</head>\n<body>\n')
        url = f'{self.base_url}/forums/tags/{tag}/'
        response = requests.get(url, headers=self.headers)
        soup = BeautifulSoup(response.text, features: 'html.parser')
        page_links = soup.select('span.page a')
        max_pages = int(page_links[-1].text) if page_links else 1

        for page_number in range(1, max_pages + 1):
            page_url = f'{self.base_url}/forums/tags/{tag}/page/{page_number}/'
            self.caretaker.add_memento(self.create_memento())
            self._handler_scrape_page( handler: '_scrape_page',output_filename, page_url, start_date, end_date)
            self.current_page += 1
        with open(result_file_path, 'a', encoding='utf-8') as file:
            file.write('</body>\n</html>')

        self._update_parser_stats(url)
    else: raise 'not this handler'
```

_handler_scrape_page:

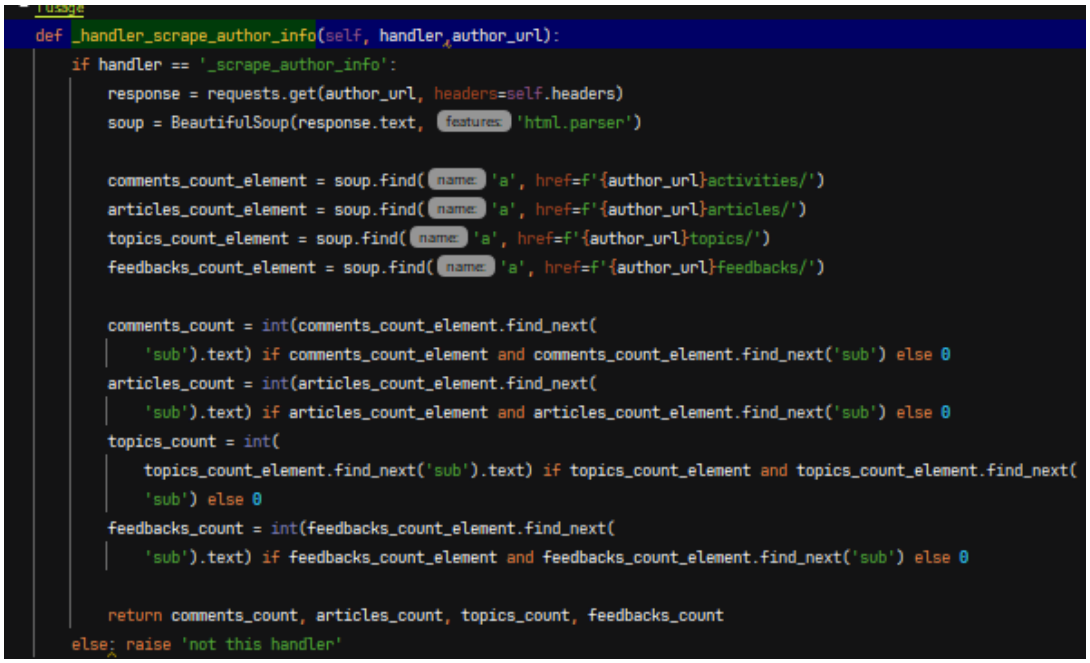
```python
def _handler_scrape_page(self, handler,output_filename, url, start_date=None, end_date=None):
    if handler == '_scrape_page':
        response = requests.get(url, headers=self.headers)
        soup = BeautifulSoup(response.text, features: 'html.parser')
        posts = soup.find_all('article')

        for post in posts:
            title = post.find('h2').find('a').text
            date_str = post.find('time', class_='date').text
            date = dateparser.parse(date_str, settings={'TIMEZONE': 'Europe/Kiev'})
            author_tag = post.find('div', class_='info').find('a', class_='author')
            author_name = author_tag.text.strip()
            author_profile_url = author_tag['href']
            link = post.find('h2').find('a')['href']
            if start_date and date < start_date: continue
            if end_date and date > end_date: continue
            comments_count = self._handler_scrape_comments_count( handler: '_scrape_comments_count',link)
            comments, articles, topics, feedbacks = self._handler_scrape_author_info( handler: '_scrape_author_info',author_profile_url)
            post_data = {
                'title': title,
                'date': date_str,
                'author_name': author_name,
                'link': link,
                'author_profile_url': author_profile_url,
                'comments_count': comments_count,
                'comments': comments,
                'articles': articles,
                'topics': topics,
                'feedbacks': feedbacks
            }

            self._handler_save_to_html( handler: '_save_to_html',output_filename, post_data)
    else: raise 'not this handler'
```

_handler_scrape_comments_count:



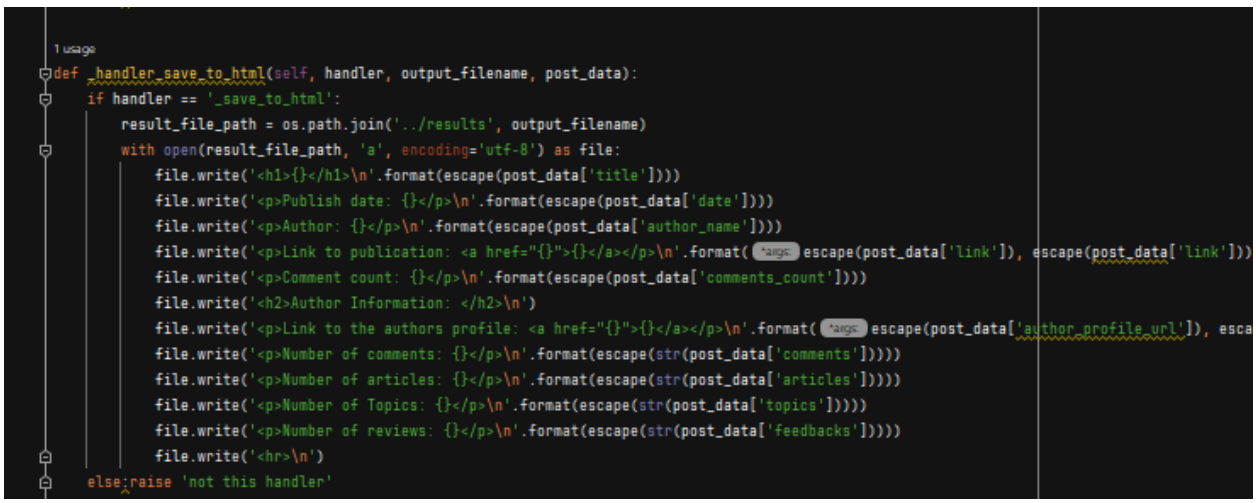
```
def _handler_scrape_comments_count(self, handler,post_url):
    if handler == '_scrape_comments_count':
        response = requests.get(post_url, headers=self.headers)
        soup = BeautifulSoup(response.text, features: 'html.parser')
        comments_count_element = soup.find( name: 'h3', id='lblCommentsCount')
        return comments_count_element.text.strip() if comments_count_element else '0'
    else: raise 'not this handler'
```

_handler_scrape_author_info:



```
def _handler_scrape_author_info(self, handler,author_url):
    if handler == '_scrape_author_info':
        response = requests.get(author_url, headers=self.headers)
        soup = BeautifulSoup(response.text, features: 'html.parser')

        comments_count_element = soup.find( name: 'a', href=f'{author_url}activities/')
        articles_count_element = soup.find( name: 'a', href=f'{author_url}articles/')
        topics_count_element = soup.find( name: 'a', href=f'{author_url}topics/')
        feedbacks_count_element = soup.find( name: 'a', href=f'{author_url}feedbacks/')

        comments_count = int(comments_count_element.find_next(
            'sub').text) if comments_count_element and comments_count_element.find_next('sub') else 0
        articles_count = int(articles_count_element.find_next(
            'sub').text) if articles_count_element and articles_count_element.find_next('sub') else 0
        topics_count = int(
            topics_count_element.find_next('sub').text) if topics_count_element and topics_count_element.find_next(
            'sub') else 0
        feedbacks_count = int(feedbacks_count_element.find_next(
            'sub').text) if feedbacks_count_element and feedbacks_count_element.find_next('sub') else 0

        return comments_count, articles_count, topics_count, feedbacks_count
    else: raise 'not this handler'
```

_handler_save_to_html:



```
1 usage
def _handler_save_to_html(self, handler, output_filename, post_data):
    if handler == '_save_to_html':
        result_file_path = os.path.join('../results', output_filename)
        with open(result_file_path, 'a', encoding='utf-8') as file:
            file.write('<h1>{}</h1>\n'.format(escape(post_data['title'])))
            file.write('<p>Publish date: {}</p>\n'.format(escape(post_data['date'])))
            file.write('<p>Author: {}</p>\n'.format(escape(post_data['author_name'])))
            file.write('<p>Link to publication: <a href="{}">{}</a></p>\n'.format( *args: escape(post_data['link']), escape(post_data['link'])))
            file.write('<p>Comment count: {}</p>\n'.format(escape(post_data['comments_count'])))
            file.write('<h2>Author Information: </h2>\n')
            file.write('<p>Link to the authors profile: <a href="{}">{}</a></p>\n'.format( *args: escape(post_data['author_profile_url']), esca
            file.write('<p>Number of comments: {}</p>\n'.format(escape(str(post_data['comments'])))))
            file.write('<p>Number of articles: {}</p>\n'.format(escape(str(post_data['articles'])))))
            file.write('<p>Number of Topics: {}</p>\n'.format(escape(str(post_data['topics'])))))
            file.write('<p>Number of reviews: {}</p>\n'.format(escape(str(post_data['feedbacks'])))))
            file.write('<hr>\n')
    else:raise 'not this handler'
```