

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Інженерія програмного забезпечення

Лабораторна робота №6

«Шаблони поведінки. Шаблони strategy, Chain of
Responsibility, visitor»

Виконав:
студент групи ІО-32
Крадожон М.
Перевірів(-ла):
Васильєва М.

Київ – 2023

Лабораторна робота №5

Тема: «Шаблони поведінки. Шаблони strategy, Chain of Responsibility, visitor»

Мета: Вивчення шаблонів поведінки. Отримання базових навичок з застосування шаблонів Strategy, Chain of Responsibility, Visitor.

Варіант: 3215 % 10 = 5

Визначити специфікації класів, що реалізують обробку HTTP-запитів різних типів (наприклад GET та POST). Реалізувати можливість динамічної зміни кількості обробників. Забезпечити децентралізацію та слабку зв'язаність обробників.

Код (TypeScript):

```
1  /**
2  * Інтерфейс, що представляє HTTP-запит.
3  */
4  interface HttpRequest {
5  /** Метод HTTP-запиту, може бути або 'GET', або 'POST'. */
6  method: 'GET' | 'POST'
7  /** Необов'язковий параметр - тіло запиту, що містить дані у вигляді рядка. */
8  body?: string
9  }
10
11 /**
12 * Інтерфейс для обробки HTTP-запитів.
13 * Визначає методи для обробки запитів та встановлення наступного обробника в
ланцюжку.
14 */
15 interface HttpHandler {
16 /**
17  * Обробляє HTTP-запит.
18  * @param req - HTTP-запит, який потрібно обробити.
19  * @returns Булеве значення, що вказує, чи був запит оброблений.
20  */
21  handleRequest(req: HttpRequest): boolean
22
23 /**
24  * Встановлює наступний обробник у ланцюжку відповідальності.
25  * @param handler - Наступний обробник.
26  * @returns Екземпляр обробника для можливості ланцюгового виклику.
27  */
28  setNext(handler: HttpHandler): HttpHandler
29  }
30
31 /**
32 * Базовий клас, що реалізує інтерфейс HttpHandler.
33 * Дозволяє встановлювати посилання на наступний обробник у ланцюжку.
34 */
```

```

35 class BaseHandler implements HttpHandler {
36 /** Зберігає посилання на наступний обробник у ланцюжку. */
37 private nextHandler: HttpHandler | null = null
38
39 /**
40  * Встановлює наступний обробник у ланцюжку.
41  * @param handler - Наступний обробник.
42  * @returns Екземпляр обробника для можливості ланцюгового виклику.
43  */
44 setNext(handler: HttpHandler): HttpHandler {
45     this.nextHandler = handler
46     return handler
47 }
48
49 /**
50  * Намагається обробити запит, передаючи його далі, якщо доступний наступний
51  обробник.
52  * Виводить повідомлення, якщо обробник не знайдено.
53  * @param req - HTTP-запит для обробки.
54  * @returns Булеве значення, що вказує, чи був запит оброблений.
55  */
56 handleRequest(req: HttpRequest): boolean {
57     if (this.nextHandler) {
58         return this.nextHandler.handleRequest(req)
59     }
60     console.log('Запит не оброблено.')
61     return false
62 }
63
64 /**
65  * Клас обробника для обробки GET-запитів.
66  * Розширює BaseHandler і обробляє запити, коли метод - 'GET'.
67  */
68 class GetRequestHandler extends BaseHandler {
69 /**
70  * Обробляє GET-запит, якщо метод відповідає.
71  * @param req - HTTP-запит для обробки.
72  * @returns Булеве значення, що вказує, чи був запит оброблений.
73  */
74 handleRequest(req: HttpRequest): boolean {
75     if (req.method === 'GET') {
76         console.log(`Обробка GET-запиту з тілом: ${req.body}`)
77         return true
78     }
79     return super.handleRequest(req)
80 }
81 }
82
83 /**
84  * Клас обробника для обробки POST-запитів.
85  * Розширює BaseHandler і обробляє запити, коли метод - 'POST'.
86  */

```

```

87 class PostRequestHandler extends BaseHandler {
88 /**
89  * Обробляє POST-запит, якщо метод відповідає.
90  * @param req - HTTP-запит для обробки.
91  * @returns Булеве значення, що вказує, чи був запит оброблений.
92  */
93 handleRequest(req: HttpRequest): boolean {
94   if (req.method === 'POST') {
95     console.log(`Обробка POST-запиту з тілом: ${req.body}`)
96     return true
97   }
98   return super.handleRequest(req)
99 }
100 }
101
102 const getRequestHandler = new GetRequestHandler()
103 const postRequestHandler = new PostRequestHandler()
104 getRequestHandler.setNext(postRequestHandler)
105
106 const request: HttpRequest = { method: 'GET', body: 'Test data' }
107 getRequestHandler.handleRequest(request)
108
109

```

Результат:

```
> yarn run start
```

Handling GET request with body: Test data

Документація (за допомогою TypeDoc):

lab3 / BaseHandler /

Class BaseHandler

Базовий клас, що реалізує інтерфейс `HttpHandler`. Дозволяє встановлювати посилання на наступний обробник у ланцюжку.

Hierarchy (view full)

- BaseHandler
 - GetRequestHandler
 - PostRequestHandler

▼ Methods

handleRequest

`handleRequest(req): boolean`

Намагається обробити запит, передаючи його далі, якщо доступний наступний обробник. Виводить повідомлення, якщо обробник не знайдено.

Parameters

- `req: HttpRequest`

HTTP-запит для обробки.

Returns *boolean*

Булеве значення, що вказує, чи був запит оброблений.

Implementation of [HttpHandler.handleRequest](#)

Defined in `index.ts:55`

Висновок:

Для виконання лабораторної роботи було прочитано та засвоєно необхідний теоретичний матеріал. Виконано 5 варіант, створений проєкт зі інтерфейсом та необхідними класами, для реалізації проєкту було використано шаблон “Chain of Responsibility”. Також було створено коментарі у кожному файлі для TypeDoc. Мета лабораторної роботи виконана.