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.
Перевірив(-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):

```typescript
/**
* Інтерфейс, що представляє HTTP-запит.
*/
interface HttpRequest {
/** Метод HTTP-запиту, може бути або 'GET', або 'POST'. */
method: 'GET' | 'POST'
/** Необов'язковий параметр - тіло запиту, що містить дані у вигляді рядка. */
body?: string
}

/**
* Інтерфейс для обробки HTTP-запитів.
* Визначає методи для обробки запитів та встановлення наступного обробника в ланцюжку.
*/
interface HttpHandler {
/**
 * Обробляє HTTP-запит.
 * @param req - HTTP-запит, який потрібно обробити.
 * @returns Булеве значення, що вказує, чи був запит оброблений.
 */
handleRequest(req: HttpRequest): boolean

/**
 * Встановлює наступний обробник у ланцюжку відповідальності.
 * @param handler - Наступний обробник.
 * @returns Екземпляр обробника для можливості ланцюгового виклику.
 */
setNext(handler: HttpHandler): HttpHandler
}

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

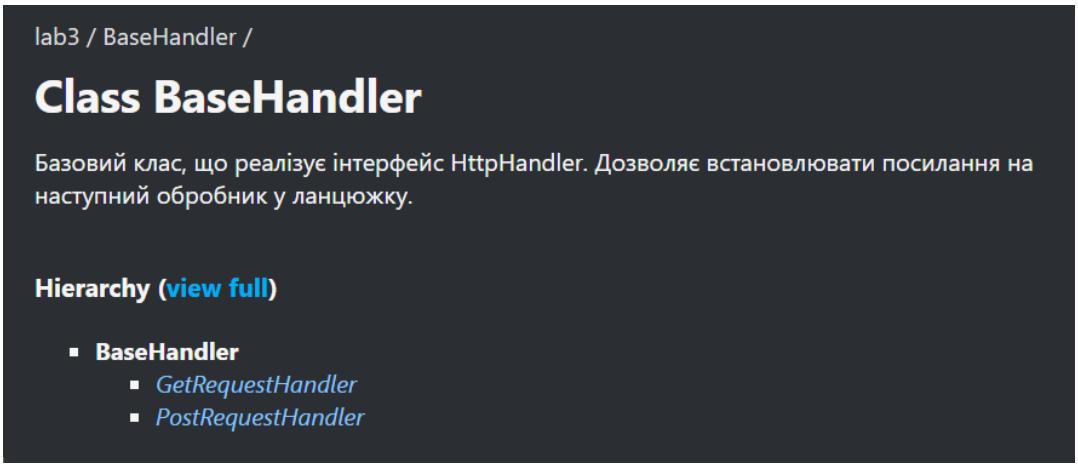
```
 class BaseHandler implements HttpHandler {
/** Зберігає посилання на наступний обробник у ланцюжку. */
private nextHandler: HttpHandler | null = null

/**
 * Встановлює наступний обробник у ланцюжку.
 * @param handler - Наступний обробник.
 * @returns Екземпляр обробника для можливості ланцюгового виклику.
 */
setNext(handler: HttpHandler): HttpHandler {
 this.nextHandler = handler
 return handler
}

/**
 * Намагається обробити запит, передаючи його далі, якщо доступний наступний
обробник.
 * Виводить повідомлення, якщо обробник не знайдено.
 * @param req - HTTP-запит для обробки.
 * @returns Булеве значення, що вказує, чи був запит оброблений.
 */
handleRequest(req: HttpRequest): boolean {
 if (this.nextHandler) {
  return this.nextHandler.handleRequest(req)
 }
 console.log('Запит не оброблено.')
 return false
}
 }

 /**
* Клас обробника для обробки GET-запитів.
* Розширює BaseHandler і обробляє запити, коли метод - 'GET'.
*/
 class GetRequestHandler extends BaseHandler {
/**
 * Обробляє GET-запит, якщо метод відповідає.
 * @param req - HTTP-запит для обробки.
 * @returns Булеве значення, що вказує, чи був запит оброблений.
 */
handleRequest(req: HttpRequest): boolean {
 if (req.method === 'GET') {
  console.log(`Обробка GET-запиту з тілом: ${req.body}`)
  return true
 }
 return super.handleRequest(req)
}
}

/**
* Клас обробника для обробки POST-запитів.
* Розширює BaseHandler і обробляє запити, коли метод - 'POST'.
*/
```

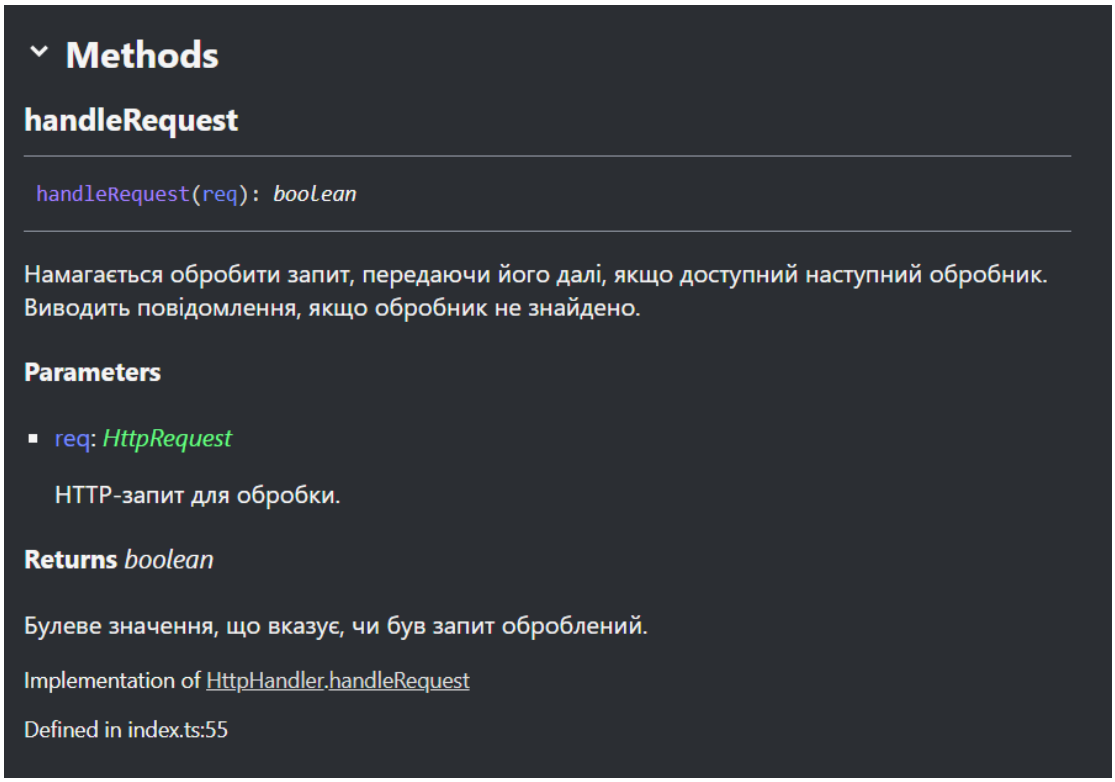
```
 class PostRequestHandler extends BaseHandler {
/**
 * Обробляє POST-запит, якщо метод відповідає.
 * @param req - HTTP-запит для обробки.
 * @returns Булеве значення, що вказує, чи був запит оброблений.
 */
handleRequest(req: HttpRequest): boolean {
 if (req.method === 'POST') {
  console.log(`Обробка POST-запиту з тілом: ${req.body}`)
  return true
 }
 return super.handleRequest(req)
}
 }

 const getRequestHandler = new GetRequestHandler()
 const postRequestHandler = new PostRequestHandler()
 getRequestHandler.setNext(postRequestHandler)

 const request: HttpRequest = { method: 'GET', body: 'Test data' }
 getRequestHandler.handleRequest(request)


```

Результат:

```
> yarn run start

Handling GET request with body: Test data
```

Документація (за допомогою TypeDoc):





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.