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 6
З дисципліни: «Теорія розробки програмного забезпечення»
Web-browser
Factory Method

Виконав:

студент групи ІА-14
Калінін Я.В

Перевірив:
Мягкий М.Ю

Київ 2023

Мета: Реалізувати шаблон проектування «FactoryMethod» до заданої теми : 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.

Теоретичні відомості : Фабричний метод — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів. Щоб ця система запрацювала, всі об’єкти, що повертаються, повинні мати спільний інтерфейс. Підкласи зможуть виготовляти об’єкти різних класів, що відповідають одному і тому самому інтерфейсу.

Код:

```
1  package com.example.demo.factory;
2
3  import com.example.demo.handler.ResponseHandler;
4
   3 implementations  Qwertua
5  public interface ErrorHandlerFactory {
       1 usage  3 implementations  Qwertua
6      ResponseHandler createHandler();
7  }
```

Це інтерфейс, який визначає метод createHandler(). Кожна фабрика обробників помилок має реалізувати цей метод, щоб створювати конкретні об'єкти обробників помилок.

class BadGatewayHandler

```
package com.example.demo.factory;

import com.example.demo.handler.BadGatewayHandler;
import com.example.demo.handler.ResponseHandler;

Qwertua
public class BadGatewayHandlerFactory implements ErrorHandlerFactory {
    1 usage  Qwertua
    @Override
    public ResponseHandler createHandler() {
        return new BadGatewayHandler();
    }
}
```

Ця фабрика відповідає за створення об'єктів BadGatewayHandler. Її реалізація методу createHandler() створює новий об'єкт BadGatewayHandler і повертає його.

class NotFoundHandlerFactory

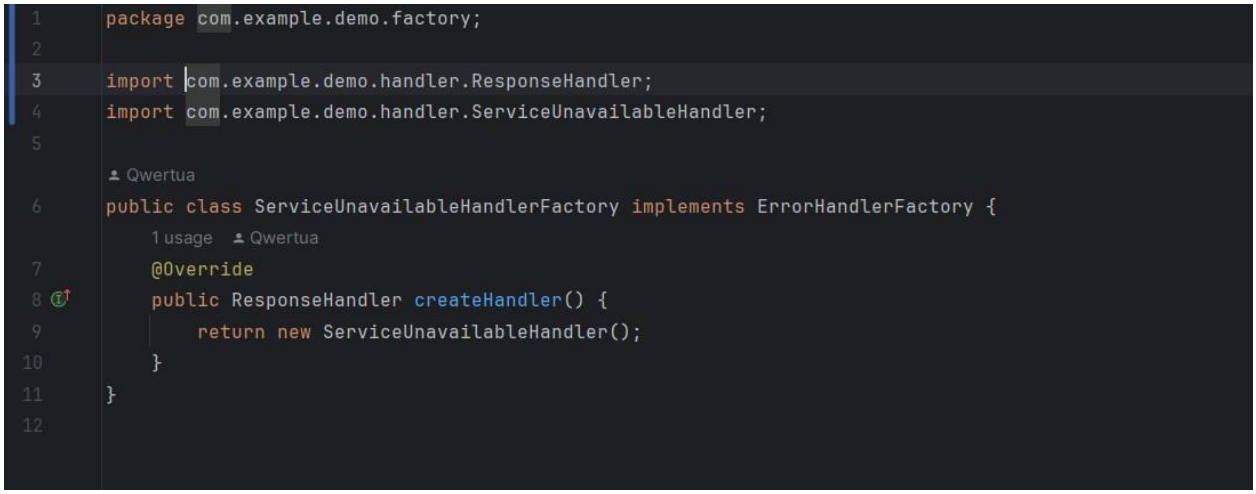
```
package com.example.demo.factory;

import com.example.demo.handler.NotFoundHandler;
import com.example.demo.handler.ResponseHandler;

Qwertua
public class NotFoundHandlerFactory implements ErrorHandlerFactory {
    1 usage  Qwertua
    @Override
    public ResponseHandler createHandler() {
        return new NotFoundHandler();
    }
}
```

createHandler(): Цей метод створює і повертає новий об'єкт NotFoundHandler, який є конкретною реалізацією інтерфейсу ResponseHandler.

class ServiceUnavailableHandlerFactory

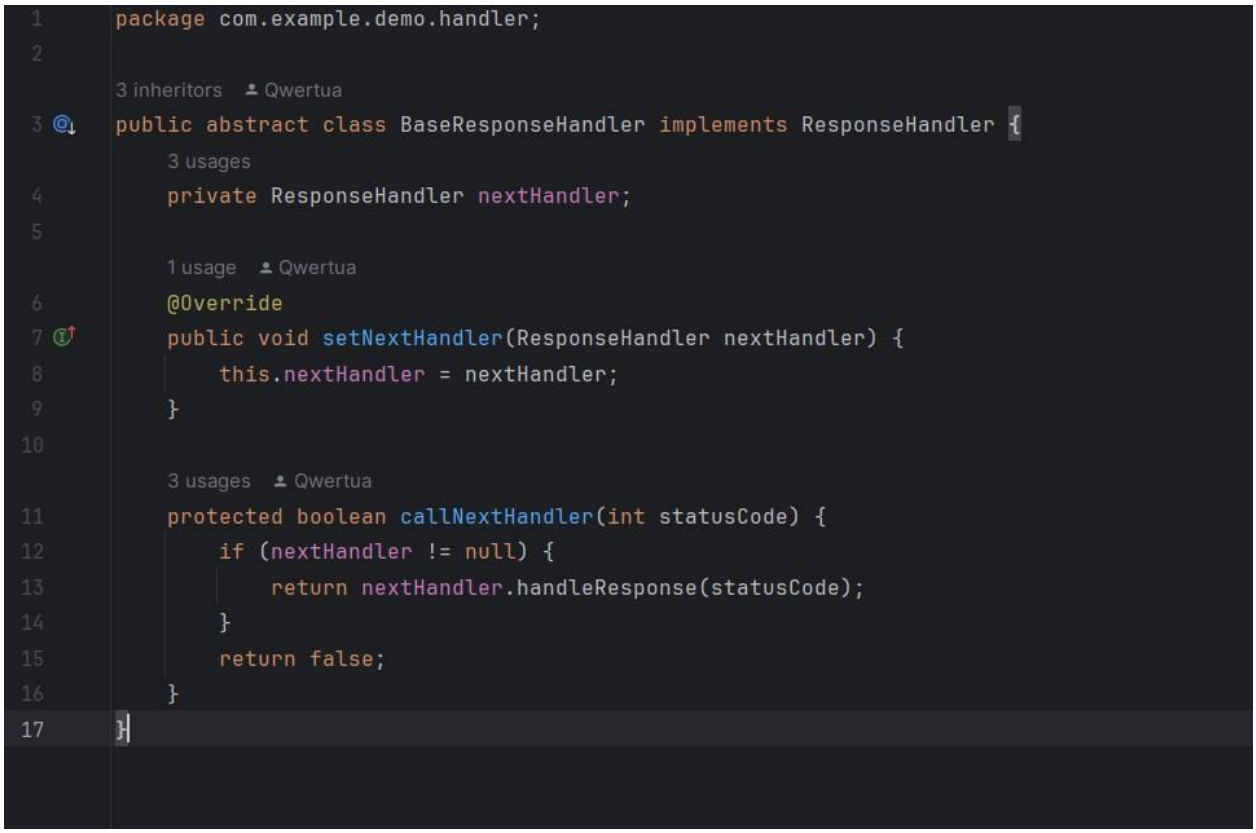


```
package com.example.demo.factory;

import com.example.demo.handler.ResponseHandler;
import com.example.demo.handler.ServiceUnavailableHandler;

Qwertua
public class ServiceUnavailableHandlerFactory implements ErrorHandlerFactory {
    1 usage  Qwertua
    @Override
    public ResponseHandler createHandler() {
        return new ServiceUnavailableHandler();
    }
}
```

class BaseResponseHandler



```
package com.example.demo.handler;

3 inheritors  Qwertua
public abstract class BaseResponseHandler implements ResponseHandler {
    3 usages
    private ResponseHandler nextHandler;

    1 usage  Qwertua
    @Override
    public void setNextHandler(ResponseHandler nextHandler) {
        this.nextHandler = nextHandler;
    }

    3 usages  Qwertua
    protected boolean callNextHandler(int statusCode) {
        if (nextHandler != null) {
            return nextHandler.handleResponse(statusCode);
        }
        return false;
    }
}
```

Цей абстрактний клас реалізує інтерфейс ResponseHandler і містить загальні методи для всіх об'єктів обробників помилок. Метод setNextHandler встановлює наступний обробник у ланцюжок, а callNextHandler викликає обробку наступного обробника у ланцюжку.

BadGatewayHandler, NotFoundHandler, ServiceUnavailableHandler:

Ці класи розширюють BaseResponseHandler і надають конкретні реалізації для обробки певних статусних кодів помилок.

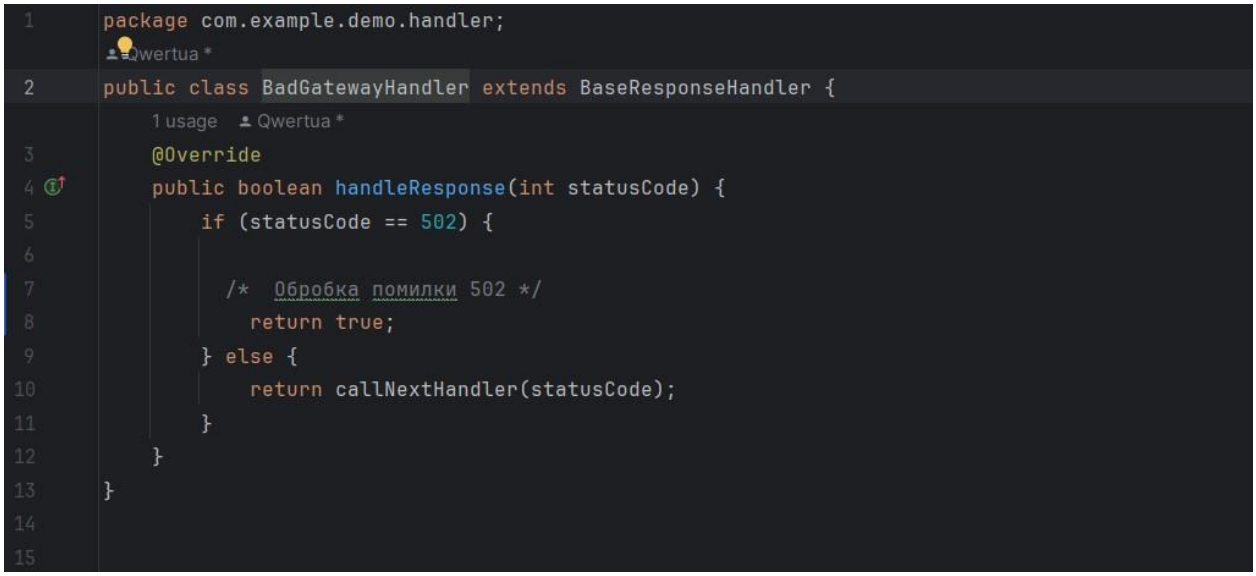
```java
package com.example.demo.handler;
Qwertua *
public class NotFoundHandler extends BaseResponseHandler {
    1 usage  Qwertua *
    @Override
    public boolean handleResponse(int statusCode) {
        if (statusCode == 404) {

          /*   Обробка помилки 404*/
            return true;
        } else {
            return callNextHandler(statusCode);
        }
    }
}
```

```java
package com.example.demo.handler;

Qwertua *
public class ServiceUnavailableHandler extends BaseResponseHandler {
    1 usage  Qwertua *
    @Override
    public boolean handleResponse(int statusCode) {
        if (statusCode == 503) {

           /* Обробка помилки 503*/
            return true;
        } else {
            return callNextHandler(statusCode);
        }
    }
}
```



```java
package com.example.demo.handler;
Qwertua *
public class BadGatewayHandler extends BaseResponseHandler {
    1 usage  Qwertua *
    @Override
    public boolean handleResponse(int statusCode) {
        if (statusCode == 502) {

          /*  Обробка помилки 502 */
            return true;
        } else {
            return callNextHandler(statusCode);
        }
    }
}
```

Висновок : У цій лабораторній роботі я детально ознайомився та реалізував відповідно до своєї теми шаблон проектування «ChainOfResponsibility»