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

Технологія розроблення програмного забезпечення

ШАБЛОНИ «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Варіант 19

Виконав
студент групи ІА-13
Павлюк Оскар Ігорович

Перевірив:
Мягкий М. Ю.

Київ 2023р.

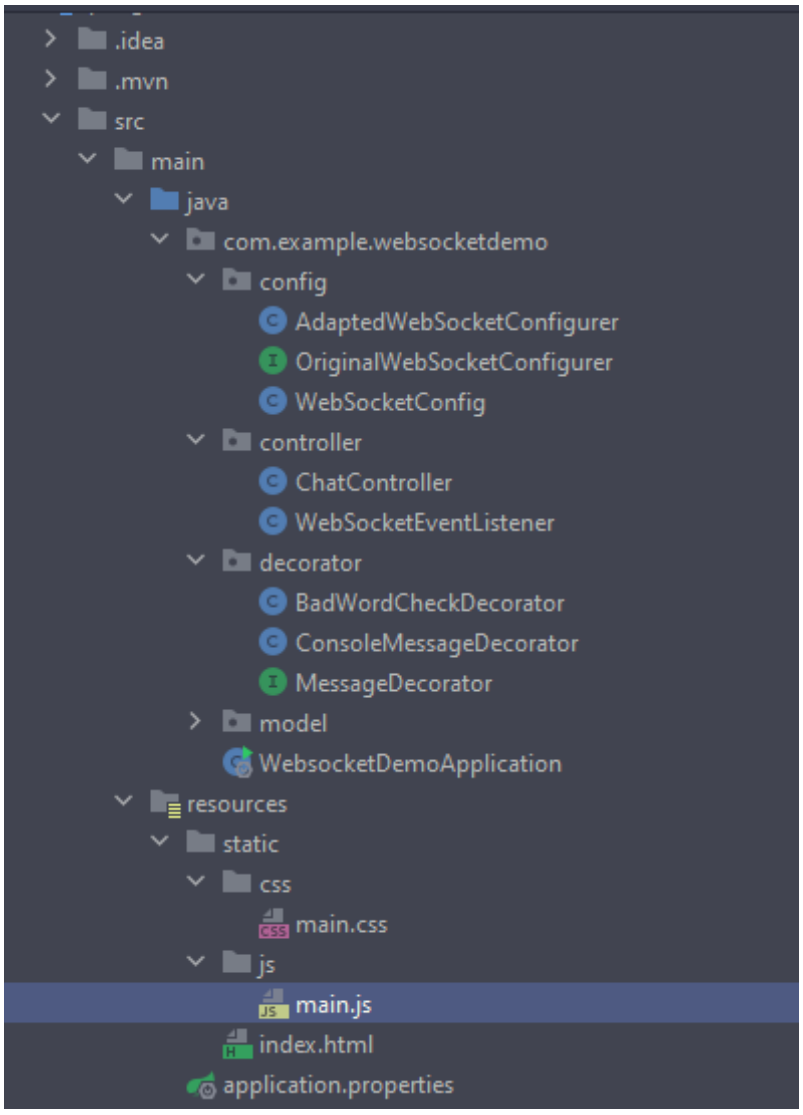
Мета: • Ознайомитися з короткими теоретичними відомостями. • Реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей. • Застосування одного з даних шаблонів при реалізації програми.

Хід роботи

..19 IRC client (singleton, builder, abstract factory, template method, composite, client-server)

Клієнт для IRC-чатів з можливістю вказівки порту і адреси з'єднання, підтримка базових команд (підключення до чату, створення чату, установка імені, реєстрація, допомога і т.д.), отримання метаданих про канал.

Структура проекту:



Вигляд роботи:

Нормальне повідомлення

Oskar joined

O **Oskar**
normal message

```
2023-12-13 00:58:47.212  INFO 26452 --- [tboundChannel-7] c.e.w.controller.WebSocketEventListener  : Received a new web socket connection
Received message: normal message
```

```
>>> SEND                                                    stomp.min.js:8
destination:/app/chat.sendMessage
content-length:59

{"sender":"Oskar","content":"normal message","type":"CHAT"}
```

```
<<< MESSAGE                                                  stomp.min.js:8
destination:/topic/public
content-type:application/json
subscription:sub-0
message-id:mwhs2enf-4
content-length:59

{"type":"CHAT","content":"normal message","sender":"Oskar"}
```

Заборонене повідомлення

Invalid message format

```
Помилка: Не можна використовувати погані слова.
```

```
>>> SEND                                                    stomp.min.js:8
destination:/app/chat.sendMessage
content-length:49

{"sender":"Oskar","content":"shot","type":"CHAT"}
```

```
<<< MESSAGE                                                  stomp.min.js:8
destination:/topic/public
content-type:application/json
subscription:sub-0
message-id:mwhs2enf-5
content-length:42

{"type":null,"content":null,"sender":null}
```

Реалізація шаблону “decorator”

У нас є інтерфейс **MessageDecorator**, який визначає метод **processMessage**, а два класи, **ConsoleMessageDecorator** і **BadWordCheckDecorator**, реалізують цей інтерфейс, додаючи свою функціональність до обробки повідомлень.

Ми також використовуємо композицію, де **BadWordCheckDecorator** містить посилання на інший об'єкт **MessageDecorator**, що дозволяє нам створювати ланцюжки декораторів і додавати різні функціональності до обробки повідомлень.

```
package com.example.websocketdemo.decorator;

import com.example.websocketdemo.model.ChatMessage;

import java.util.HashSet;
import java.util.Set;

public class BadWordCheckDecorator implements MessageDecorator {

    private final MessageDecorator messageDecorator;
    private final Set<String> badWords;

    public BadWordCheckDecorator(MessageDecorator messageDecorator) {
        this.messageDecorator = messageDecorator;

        // Додамо сюди слова, які ми хочемо перевіряти
        this.badWords = new HashSet<>();
        badWords.add("bad");
        badWords.add("shot");
        badWords.add("gun");
    }

    @Override
    public ChatMessage processMessage(ChatMessage chatMessage) {
        if (containsBadWords(chatMessage.getContent())) {
            System.out.println("Помилка: Не можна використовувати погані слова.");
            return null; // Повертаємо null, щоб показати, що повідомлення не було
оброблено
        } else {
            return messageDecorator.processMessage(chatMessage);
        }
    }

    private boolean containsBadWords(String str) {
        for (String badWord : badWords) {
            if (str != null && str.toLowerCase().contains(badWord)) {
                return true;
            }
        }
        return false;
    }
}
```

```java
package com.example.websocketdemo.decorator;


import com.example.websocketdemo.model.ChatMessage;

public class ConsoleMessageDecorator implements MessageDecorator {

    @Override
    public ChatMessage processMessage(ChatMessage chatMessage) {
        System.out.println("Received message: " + chatMessage.getContent());
        return chatMessage;
    }
}
```

```java
package com.example.websocketdemo.decorator;

import com.example.websocketdemo.model.ChatMessage;

public interface MessageDecorator {
    ChatMessage processMessage(ChatMessage chatMessage);
}
```

Висновок: Під час цієї лабораторної роботи було розроблено частково реалізацію IRC client і також decorator.