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
«Використання шаблону Observer, тема: Download manager»
Варіант 26

Виконала
студентка групи ІА-13
Танасієнко Анастасія Вячеславівна

Перевірив:

Київ 2023р.

Мета: дослідження та реалізація шаблонів проектування «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Шаблон «Observer»

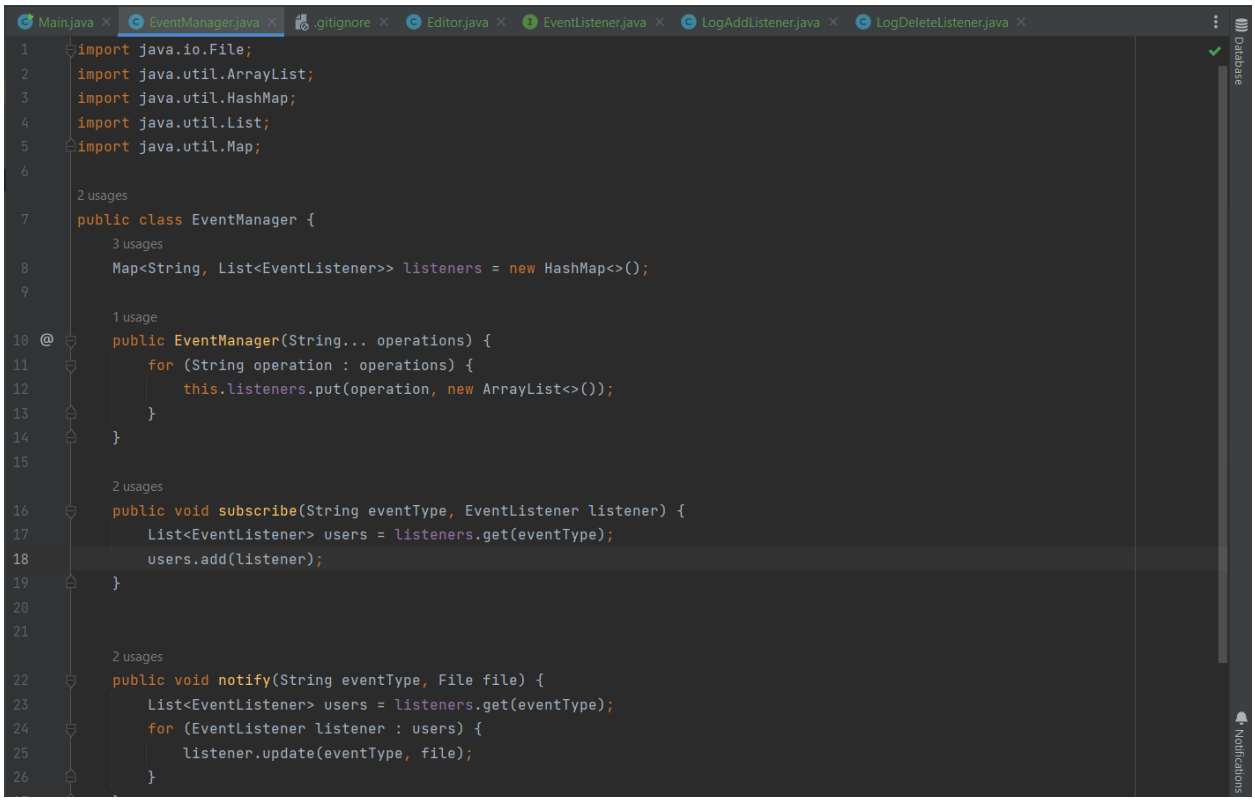
Editor - це простий текстовий редактор на Java з можливістю додавання та видалення файлів. Він використовує EventManager для відстеження подій "add" і "delete". Метод addFile додає новий файл і сповіщає про це, а метод deleteFile видаляє файл, якщо він був доданий раніше, інакше генерує виключення.

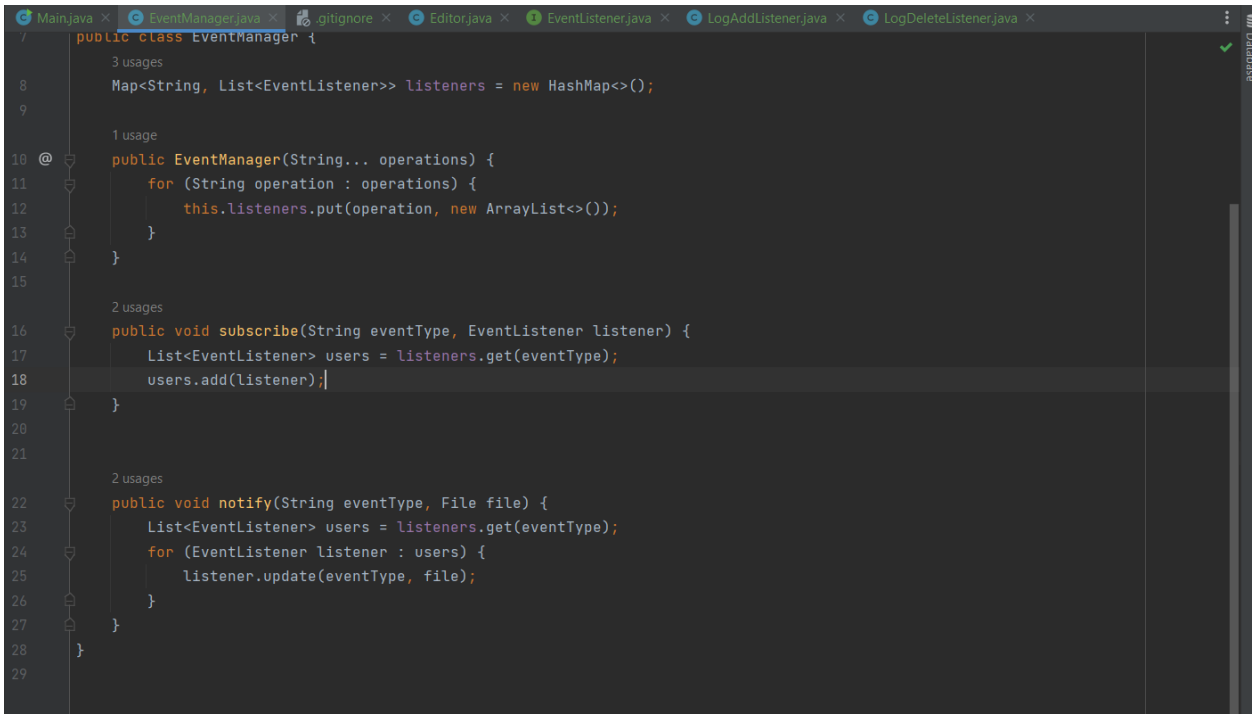
```java
import java.io.File;


public class Editor {

    public EventManager events;

    private File file;


    public Editor() {
        this.events = new EventManager( ...operations: "add", "delete");
    }


    public void addFile(String filePath) {
        this.file = new File(filePath);
        events.notify( eventType: "add", file);
    }


    public void deleteFile(String filePath) throws Exception {
        if (this.file != null) {
            events.notify( eventType: "delete", file);
        } else {
            throw new Exception("Please add a file first.");
        }
    }
}
```

Інтерфейс EventListener має метод update, який отримує тип події та об'єкт файлу для оновлення слухачів.

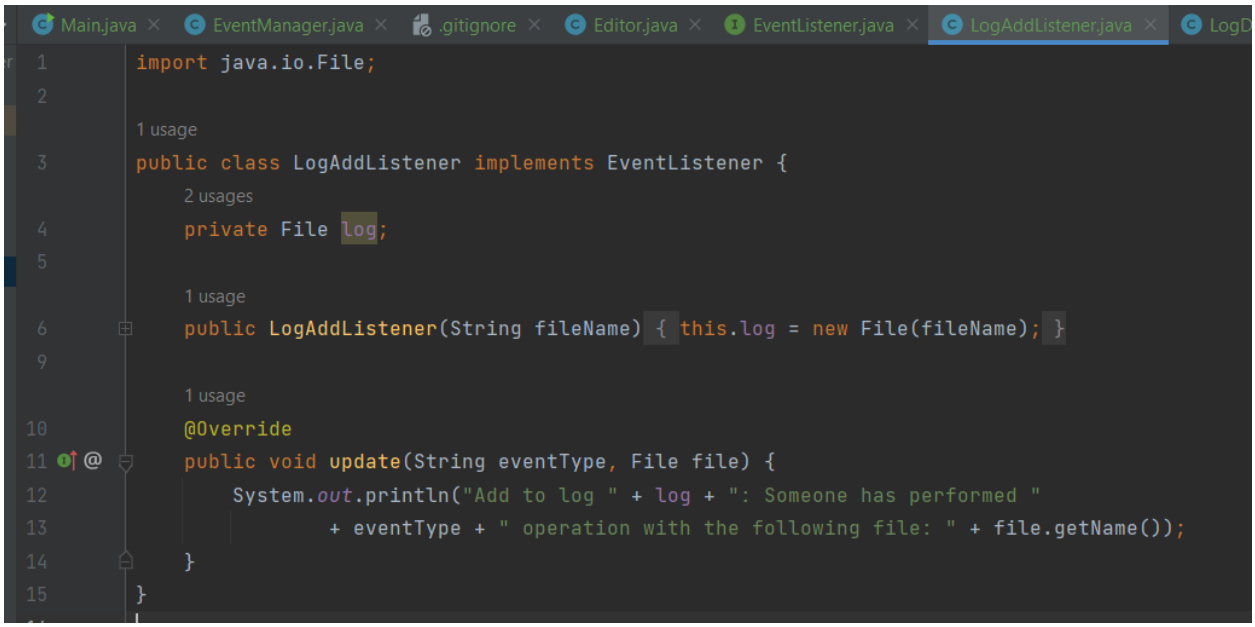
```java
import java.io.File;
9 usages  2 implementations
public interface EventListener {
    1 usage  2 implementations
    void update(String eventType, File file);
}
```

EventManager - це клас, який забезпечує механізм підписки та сповіщення про події. Він має методи subscribe для додавання слухачів і notify для повідомлення їх про зміни. Слухачі визначаються за типом події, який передається разом із файлом.

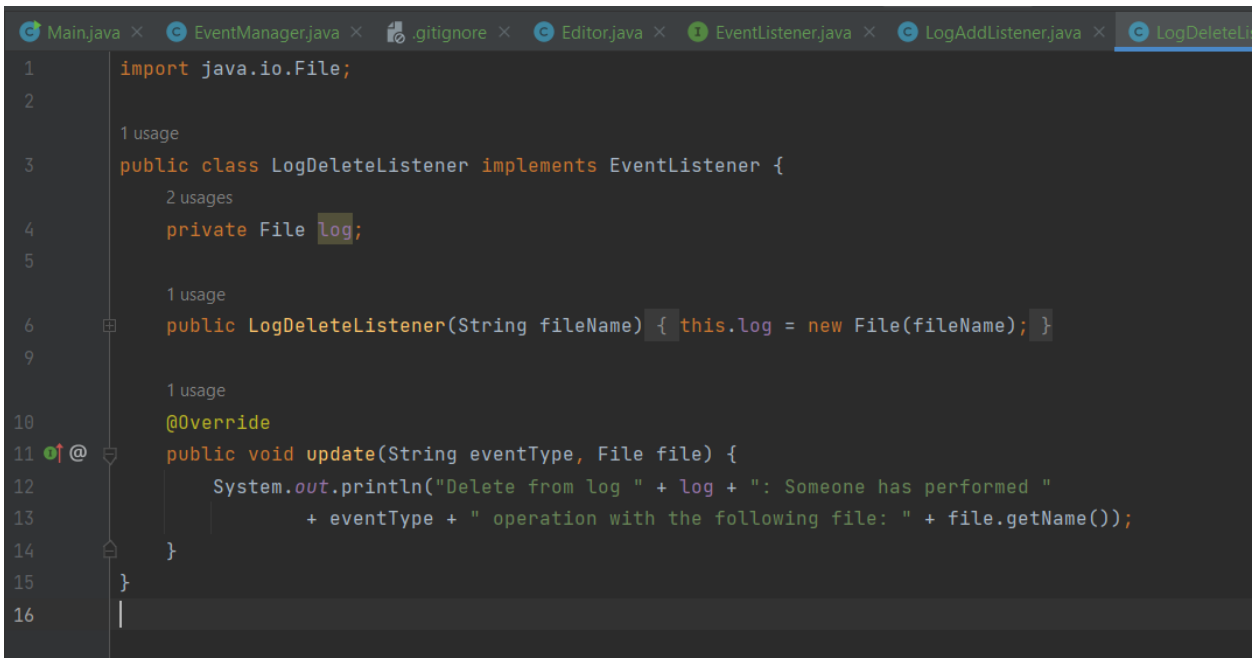




LogAddListener - клас, який слухає події, реалізує інтерфейс EventListener. При оновленні виводить повідомлення до лог-файлу про виконану операцію та файл.



LogDeleteListener - клас, який слухає події, реалізує інтерфейс EventListener. При оновленні виводить повідомлення до лог-файлу про виконану операцію видалення та файл.



Клас Main має метод main, який створює об'єкт редактора (Editor) та підписує два слухачі - LogAddListener та LogDeleteListener - на події "add" і "delete" відповідно. Потім викликає методи додавання та видалення файлу через редактор і виводить відповідні повідомлення до лог-файлу.

Висновок:

У цій лабораторній роботі були досліджені та реалізовані п'ять важливих шаблонів проектування: «Abstract Factory», «Factory Method», «Memento», «Observer» та «Decorator». Кожен із них має свою унікальну функціональність та може бути використаний для вирішення різноманітних завдань у програмній розробці.

- Шаблон "Abstract Factory" дозволяє створювати сімейства взаємодіючих об'єктів без прив'язки до конкретних класів. Він надає абстрактний інтерфейс для створення сімейств пов'язаних або взаємозалежних об'єктів.
- Шаблон "Factory Method" визначає загальний інтерфейс для створення об'єктів, але залишає вибір конкретного класу-продукту до підкласів. Він дозволяє створювати об'єкти, не вказуючи конкретний клас.
- Шаблон "Memento" дозволяє зберігати стан об'єкта так, щоб його можна було відновити в майбутньому без розкриття деталей його реалізації.
- Шаблон "Observer" визначає залежність одного об'єкта від змін у іншому об'єкті, гарантуючи, що при зміні стану одного об'єкта всі його залежності будуть автоматично сповіщені та оновлені.
- Шаблон "Decorator" дозволяє динамічно надавати об'єкту нові функції, обгортуючи його в інші класи, що реалізують однаковий інтерфейс.

Робота над цими шаблонами дозволила отримати глибше розуміння принципів об'єктно-орієнтованого програмування та паттернів проектування. Кожен з цих шаблонів має свої конкретні випадки застосування та може бути важливим інструментом для розробників у різних сценаріях розробки програмного забезпечення.