Práctica 7: Sistema de Observación Meteorológica usando el Patron Observador

Jorge Domínguez González (alu0101330600@ull.edu.es)
Adnan (alu01@ull.edu.es) Paula (alu01@ull.edu.es)

November 30, 2023

Contents

List of Figures

1 Introducción

El código proporcionado presenta una implementación de un sistema de observación meteorológica en Java. Este sistema utiliza el patrón de diseño Observador para recibir actualizaciones de las condiciones meteorológicas de diversas ciudades a través de la API de WeatherStack.

2 Estructura del Código

2.1 Clase `WeatherStackAPI`

La clase `WeatherStackAPI` se encarga de interactuar con la API de WeatherStack para obtener las condiciones meteorológicas de una ciudad específica. Utiliza la biblioteca Unirest para realizar solicitudes HTTP.

- **Método `getWeatherConditionsFrom(String zone)`**: Este método recibe el nombre de una ciudad como parámetro, realiza una solicitud a la API de WeatherStack y devuelve las condiciones meteorológicas en formato JSON.

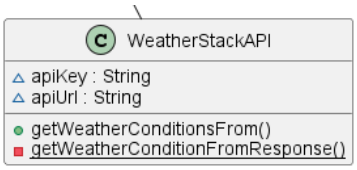

Figure 1: Diagrama de Clase weatherstack

2.2 Interfaz `Observador`

La interfaz `Observador` define el método `actualizar`, que las clases interesadas en recibir actualizaciones meteorológicas deben implementar.

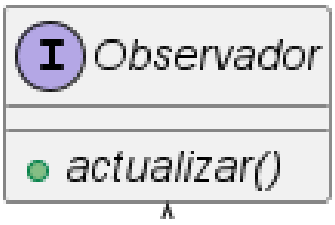

Figure 2: Diagrama de Clase IObservador

2.3 Clase `PantallaMeteorologica`

La clase `PantallaMeteorologica` implementa la interfaz `Observador` y representa la interfaz gráfica que mostrará las condiciones meteorológicas. Utiliza la biblioteca Swing para la interfaz de usuario.

- **Método `actualizar(String condiciones)`**: Este método recibe las condiciones meteorológicas como una cadena JSON y actualiza la interfaz gráfica con la información relevante.

Figure 3: Diagrama de Clase PantallaMeteo

2.4 Clase `SeleccionCiudad`

La clase `SeleccionCiudad` se encarga de permitir al usuario seleccionar las ciudades para las cuales desea recibir actualizaciones meteorológicas. Utiliza la biblioteca Swing para la interfaz de usuario.

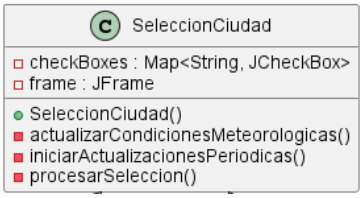

Figure 4: Diagrama de Clase seleccionCiudad

2.5 Clase `SujetoWeatherstack`

La clase `SujetoWeatherstack` actúa como el sujeto que mantiene una lista de observadores interesados en recibir actualizaciones meteorológicas.

- **Método `agregarObservador(Observador observador)`**: Este método agrega un observador a la lista de observadores.
- **Método `notificarObservadores(String condiciones)`**: Este método notifica a todos los observadores registrados con las condiciones meteorológicas actualizadas.

Figure 5: Diagrama de Clase weatherstacksujeto.png

2.6 Método `main`

El método `main` se encuentra en la clase `MainWeatherstack` y sirve como punto de entrada para la aplicación.

- Configura la interfaz de usuario inicializando instancias de `PantallaMeteorologica` y `SeleccionCiudad`.
- Registra la instancia de `PantallaMeteorologica` como observador en la instancia de `SujetoWeatherstack`.
- Permite al usuario seleccionar ciudades y muestra las condiciones meteorológicas en la interfaz gráfica.

3 Pasos del Flujo del Programa

El programa sigue estos pasos:

1. Se inicia la aplicación desde el método `main`.
2. Se configuran las instancias de la interfaz de usuario y se registran los observadores.
3. El usuario selecciona las ciudades de interés.

4. Se realiza una solicitud a la API de WeatherStack para obtener las condiciones meteorológicas de las ciudades seleccionadas.

5. Las condiciones meteorológicas se notifican a los observadores (instancia de `PantallaMeteorologica`) que actualizan la interfaz gráfica.

6. El proceso se repite cuando el usuario selecciona nuevas ciudades.

4 Interfaz Gráfica para el Usuario

La interfaz gráfica incluye una pantalla para mostrar las condiciones meteorológicas y un formulario para que el usuario seleccione las ciudades de interés.

Figure 6: Interfaz de Pantalla Meteorológica

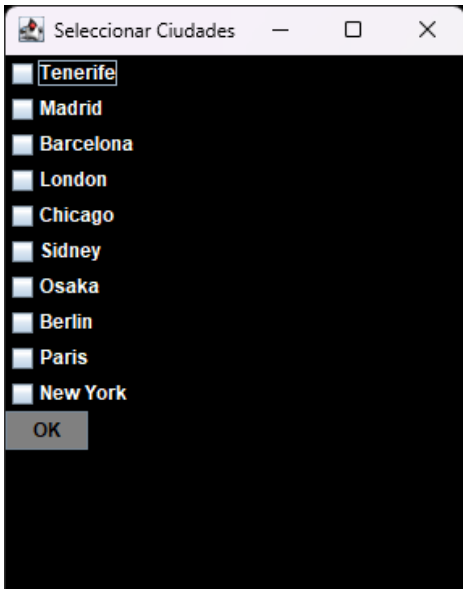

Figure 7: Interfaz de Selección de Ciudad

5 Diagrama UML

El diagrama de clases UML muestra las relaciones entre las clases del sistema.

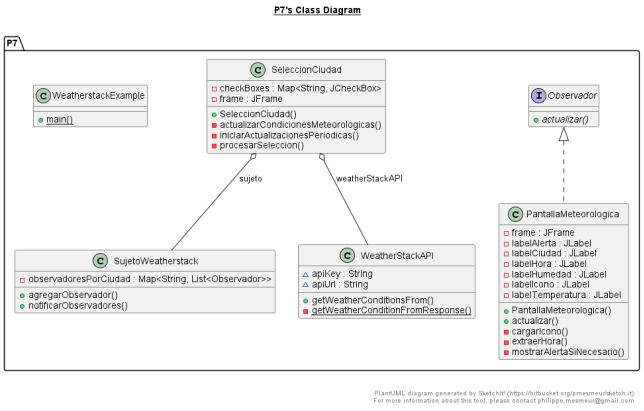

Figure 8: Diagrama de Clases UML

6 Estimación de Plazo

Se estima que el desarrollo del proyecto tomará aproximadamente el siguiente tiempo:

TAREA	TIEMPO ESPERADO	TIEMPO FINAL
Implementación del Sistema	4 Horas	8 Horas
Pruebas y Depuración	4 Horas	4 Horas
Documentación	2 Horas	3 Horas
Implementación en GitHub	1 Hora	2 Horas

Table 1: Estimación de Plazo

7 Conclusiones

El sistema de observación meteorológica implementado proporciona una interfaz amigable para que los usuarios obtengan información en tiempo real sobre las condiciones meteorológicas de las ciudades de su elección. El uso del patrón Observador facilita la actualización automática de la interfaz de usuario cuando cambian las condiciones meteorológicas.

8 Bibliografía

References

[1] Documentación de la API de WeatherStack. https://weatherstack.com/documentation

[2] Documentación de Unirest. http://kong.github.io/unirest-java/