Diseño de Software Orientado a Objetos
Práctico 3: Patrón Observer

Ejercicio 0. Descargue el código provisto en el repositorio git a continuación:

https://github.com/pponzio/oo-design-2024.git

La carpeta `3-observer-weather-station` contiene el código complementario a esta práctica. Se proveen scripts gradle para compilar automáticamente el código y ejecutar los tests (ej. ejecutar *gradle test* en línea de comandos). Se recomienda utilizar algún IDE (ej. IntelliJ Idea) para facilitar la codificación, el refactoring, la ejecución de tests y el debugging.

Ejercicio 1. Utilice el código provisto del Weather Station para experimentar y entender el patrón Observer. Utilice la metodología de TDD.

a) Cree varias instancias de los displays provistos (de tipo `Observer`), y escenarios en donde cambien los displays asociados a un objeto de tipo `WeatherData` en tiempo de ejecución.
b) Agregue un nuevo display que compute y muestre la temperatura actual en grados centígrados, a partir de los datos que le pasa un subject de tipo `WeatherData`. Cree distintos escenarios de ejecución que incluyan este nuevo display.
c) Agregue un nuevo display que compute y muestre la sensación térmica. El archivo `compute-heat-index.txt` contiene una función para computar la sensación térmica a partir de la temperatura y la humedad. Cree distintos escenarios de ejecución que incluyan este nuevo display.
d) ¿Tuvo que modificar la implementación de alguna de las clases existentes para resolver los puntos b y c anteriores?

Ejercicio 2. Cree una nueva versión de Weather Station modificando el diseño del ejercicio anterior para que cada display tome la parte que necesita del estado del objeto `WeatherData` al que está asociado (en lugar de recibir todos los datos como parte del `update`). Utilice la metodología de TDD.

Ejercicio 3. Agregue al juego de acción y aventuras desarrollado como solución al ejercicio 2 de la práctica anterior la capacidad de loguear los resultados de diferentes acciones que ocurren durante el juego. Utilice la metodología de TDD.

a) Desarrolle un logger, implementado mediante el patrón Observer, que reciba los resultados de las diferentes acciones que ocurren en el juego y los muestre por pantalla. Por ejemplo, que muestre información de cada golpe efectuado por cada

personaje en una pelea. Incluya entre la información a mostrar el tipo de personaje, el tipo de arma, el daño realizado, etc. Muestre también un mensaje cuando un personaje muere.

b) Implemente otro logger, pero que esta vez guarde los resultados en un archivo. Utilice el patrón Observer, de modo que el logger del ejercicio anterior y el del ejercicio actual puedan funcionar al mismo tiempo, y que los loggers puedan activarse y desactivarse dinámicamente.

c) Implemente un nuevo logger para llevar estadísticas de las peleas. Por ejemplo, incluya entre la información a almacenar la cantidad de victorias de cada combinación entre tipo de personaje y tipo de arma. Utilice el patrón Observer y permita que cualquier combinación posible de los loggers implementados pueda estar activada en una misma ejecución.

d) ¿Tuvo que modificar la implementación de alguna de las clases existentes para resolver los puntos b y c anteriores?