Ingeniería de Software
Año 2025
Práctico 3: Patrón Observer

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

https://github.com/ingenieria-de-software-unrc/practicos-ingenieria-25

La carpeta `3-observer` contiene el código complementario a esta práctica. Se proveen scripts gradle para construir (*build*) automáticamente el proyecto (`./gradlew build`), ejecutar el main (`./gradlew run`), y ejecutar los tests (`./gradlew test`).
Se recomienda utilizar algún IDE (por ejemplo, IntelliJ IDEA) para facilitar la codificación, el refactoring, la ejecución de tests y el debugging.

Importante: Tenga en cuenta los siguientes requisitos al realizar los ejercicios. Piense en un diseño para su aplicación y realice un diagrama de clases del diseño propuesto antes de comenzar a codificar. Mantenga el diagrama de clases actualizado con el diseño de su aplicación durante todo el ejercicio. Asegúrese que su diseño adhiere (en la medida de lo posible) a los principios de diseño vistos en la teoría. Utilice la metodología de TDD.

Ejercicio 1. Utilice el código provisto del Weather Station para experimentar y entender el patrón observer.

a) Cree varios tests con los displays provistos (de tipo `Observer`), y tests en donde cambien los displays asociados a un objeto de tipo `WeatherData` en tiempo de ejecución.
b) Agregue un nuevo tipo de display a `WeatherData` que compute y muestre la temperatura actual en grados centígrados. Cree distintos tests que incluyan este nuevo display.
c) Agregue un nuevo display a `WeatherData` 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 tests que ejerciten 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 `WeatherData` que implemente la versión pull del patrón Observer. Es decir, modifique 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`.

Ejercicio 3. Refactorice el código del ejercicio 3 de la práctica anterior (su implementación del Conway's Game of Life) para soportar distintos tipos de displays. Utilice el patrón observer para implementar los displays.

a) Soporte primero displays que muestren el juego en diferentes colores. Implemente un display que muestre las células vivas en negro y las muertas en blanco, y otro display que muestre las células vivas en blanco y las muertas en negro. Intercambiar entre un display y otro debe ser tan simple como crear un objeto diferente en un único lugar de su código. Además, debe poder intercambiar los displays en tiempo de ejecución. Cree displays para mostrar de al menos dos formas distintas el juego de la vida con colores (ej. puede crear un display que intercambiar los colores por otros diferentes).
b) Permita que su implementación vaya logueando estadísticas de juego, inicialmente por pantalla. Puede usar otros tipos de observers para esto. Implemente un observer que vaya reportando la cantidad de células de cada tipo en el tablero actual, y se vaya actualizando a medida que el tablero cambie.
c) Modifique su diseño para que luego de una cierta cantidad de iteraciones del juego (dada por el usuario) el observer del inciso anterior reporte el promedio de células de distintos tipos que registró hasta el momento (ej. 50 células vivas, 75 muertas, en promedio.
d) Agregue un nuevo observador de estadísticas que reporte qué regla aplicó en cada paso, y que cada cierta cantidad de iteraciones reporte el porcentaje de reglas que se aplicaron (ej. 10% regla de nacimiento, 50% regla de supervivencia, 40% regla de muerte).
e) Permita que los observadores de estadísticas guarden información en archivos además de por pantalla. Combine el patrón strategy con el observer para lograr esto.