Diseño de Software Orientado a Objetos
Práctico 4: Patrón Decorator

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

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

La carpeta `4-decorator-starbuzz` 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 Starbuzz Coffee para experimentar y entender el patrón Decorator. Utilice la metodología de TDD.

a) Cree varias instancias diferentes de objetos decorados y compute su descripción y su costo.
b) Incluya un: "double mocha soy latte with whip" que consiste en combinar un `HouseBlend` con soja (`Soy`), dos shots de moca (`Mocha`), y crema batida (`Whip`).
c) Agregue un nuevo condimento de su agrado, una nueva bebida, y cree instancias de objetos decorados que los usen. Compute su descripción y su costo.
d) ¿Tuvo que modificar la implementación de alguna de las clases existentes para resolver los puntos a, b y c anteriores?
e) Las bebidas disponibles ahora pueden tener 3 tamaños: pequeño, mediano y grande. El costo de los condimentos depende del tamaño de la bebida decorada. Por ejemplo, la soja puede costar $10, $15 y $20, para bebidas pequeñas, medianas y grandes, respectivamente. Modifique su diseño basado en el patrón decorator para soportar esta funcionalidad.

Ejercicio 2. Cree un nuevo decorador para `java.io.InputStream` que codifique el texto leído usando el cifrado César para una clave dada, y otro que la decodifique. Combínelo con el decorador que permite obtener la cantidad de líneas leídas, y chequee que los archivos encriptados tienen la misma cantidad de líneas que los archivos originales. Utilice la metodología de TDD.

Ejercicio 3. Para el juego de acción y aventuras desarrollado como solución al ejercicio 3 de la práctica anterior:

a) Añada la capacidad de engastar gemas en las armas que pueden tener los personajes. Diseñe distintos tipos de gemas que suman una cantidad fija al daño producido por el

arma; esta cantidad puede variar de acuerdo al tipo de gema. Las gemas engastadas ya no podrán quitarse nunca. Emplee el patrón decorator para implementar las gemas.

b) Agregue nuevas gemas y armas, y utilícelas para crear nuevos escenarios del juego.
c) ¿Tuvo que modificar la implementación de alguna de las clases existentes para resolver el punto c?
d) Permita que cada arma soporte un número máximo de gemas, que puede variar según el tipo de arma.

Utilice la metodología de TDD durante todo el ejercicio.