Práctica 02

Noe E. Amador González —- 419004815

7 de Agosto, 2020

1. Menciona los principios de diseño esenciales de los patrones Decorator y Adapter.

   - **Decorator**
     - Las clases deben estar abiertas a la extensión, pero cerradas a la modificación. O en otras palabras: implementar un diseño resistente al cambio y lo suficientemente flexible para soportar cambios en los requerimientos.
     - Adjuntar responsabilidades adicionales a un objeto de forma dinámica.
     - Proporcionar una alternativa flexible a la subclasificación para ampliar la funcionalidad.
   - **Adapter**
     - Convierte la interfaz de una clase en otra interfaz que se adapte a la que el usuario espera.
     - Permite a las clases (de dos proyectos distintos) trabajar juntas, a pesar de que sus interfaces sean incompatibles.

2. Menciona una desventaja de cada patrón.

   - **Decorator**
     - Eliminar decoradores sobre clases ya constriudas es difícil porque hay que buscar y seleccionar el decorador de manera recursiva (puede ser complicado de identificar), por lo que en el peor de los casos podria tomar costar mucho eliminarlo. Por otro lado, si el nucleo tiene demasiadas envolturas quiere decir que haber implementado *decorator* no era la mejor opción.
   - **Adapter**
     - Adapter se utiliza cuando hay 2 partes del codigo y una de estas partes no se puede modificar. En general no es bueno utilizarla a menos que sea el caso. Si utlizamos *Adapter* podríamos decir que estamos utilizando un parche en nuestro programa, lo cuál no se considera una buena práctica por su gran cantidad de desventajas.

README

- La implementación de la práctica se ha llevado a cabo con la versión 8 de Java.

- Para correr la simulacion hay que compilar todos los archivos *.java y despues utilizar la clase *WaySub* como la **clase principal** que ejecutará la simulación.

- Es importante destacar que el diagrama del patron *Decorator* y *Adapter* los he juntado en un solo diagrama. En este diagrama **no especifiqué metodos constructores** por cuestiones de estetica y tiempo.
  Tambien **omití la representacion de la clase *Menu***, la cual utilicé en la clase principal (WaySub) como apoyo para imprimir los menus necesarios en consola asi como el ticket y un método para construir un Baguette. Lo anterior con el fin de que la clase WaySub no se viera saturada de código repetido.

- Soy consciente que no tengo las mejores practicas de lenguaje (incluyendo documentacion); en mi defensa puedo decir que no tengo experiencia en Java (estudio Matematicas Aplicadas). Sin embargo tengo como meta mejorar estas practicas durante el curso y me comprometo a ir mejorando mis scripts en las siguientes tareas/practicas/proyectos nuevamente :D