PRACTICA III

Decorator y Adapter

Modelado y Programación 2020-2

Profesora: Rosa Victoria Villa Padilla
Ayudante: Arturo Lemus Pablo
Laboratorista: José Ricardo Rosas Bocanegra

Licenciatura en Ciencias de la Computación Facultad de Ciencias, UNAM

Alumno:
Jorge Iván Pérez Pérez 314211349
Nava May Arely 314206396

Menciona los principios de diseño esenciales de los patrones Decorator y Adapter. Menciona una desventaja de cada patrón.

Decorator: El patrón decorator permite añadir responsabilidades a objetos concretos de forma dinámica. Los decoradores ofrecen una alternativa más flexible que la herencia para extender las funcionalidades.

A veces se desea adicionar responsabilidades a un objeto, pero no a toda la clase. Las responsabilidades se pueden adicionar por medio de los mecanismos de Herencia, pero este mecanismo no es flexible porque la responsabilidad es adicionada estáticamente. La solución flexible es la de rodear el objeto con otro objeto que es el que adiciona la nueva responsabilidad. Este nuevo objeto es el Decorator.

Este patrón se debe utilizar cuando:

Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia.

Se quiere agregar o quitar dinámicamente la funcionalidad de un objeto.

Desventajas:

- Un decorador y sus componentes no son idénticos, desde el punto de vista de la identidad de objetos, desde el punto de vista del programador o del cliente sí que se podrían considerar iguales.
- Muchos objetos pequeños. El sistema puede ser más difícil de aprender y de depurar.

Adapter: El Patrón adapter convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen las clases que de otra manera no podrían por tener clases incompatibles.

Sirve como intermediario entre dos clases, convirtiendo las interfases de una clase que pueda ser utilizada por otra.

A veces una clase es diseñada para que sea reusable, pero no lo es, porque la interfase no con concuerda con la interfase que una aplicación requiere es aquí donde entra Adapter.

Desventajas:

- Aumenta innecesariamente el tamaño del código, ya que la herencia de clase se usa menos y se duplica innecesariamente mucho código entre clases.
- A veces se requieren muchas adaptaciones a lo largo de una cadena adaptadora para alcanzar el tipo que se requiere.

Anotaciones

Para correr esta practica ubicarse en la carpeta src, compilar todos los archivos .java y después ejecutar el archivo Main.java