Trabajo Práctico 5 – Adapter y Decorator

1. Un sistema trabaja con diferentes tipos de motores (Común, Económico) que comparten características comunes así como su funcionamiento, que está dado por operaciones como **arrancar**, **acelerar** y **apagar**.

Se desea incorporar al sistema una clase ya existente de tipo motor eléctrico con un funcionamiento diferente al de los demás. Estos motores pueden realizar operaciones como **conectar** y **activar**, **moverMasRapido**, **detener** y **desconectar**.

Se debe adaptar la nueva clase de forma que no se vea afectada la lógica de la aplicación cliente que interactúa con las clases que representan los motores.

a) Aplique el patrón **Adapter,** utilizando un adaptador de objetos, para diseñar el modelo de clases que de solución al problema planteado.

b) Implemente la solución en Java, utilizando mensajes informativos por pantalla de la operación que se está realizando.

2. Dado el ejemplo del Canvas, Figuras aplicando composite (https://github.com/enriquemolinari/oop2-patterns1), quitemos la dependencia de cada Figura sobre Graphics2D utilizando el patron adapter. Primero cree una interfaz "Panel" que ofrezca los servicios de pintar los tres tipos de figura, circulo, linea y texto. Luego implemente un adapter para adaptar Panel a Graphics2D de modo que el sistema funcione. De esta forma tenemos el mismo resultado pero habiendo quitado la dependencia de las figuras sobre Graphics2D.

3. Supongamos la siguiente clase Reporte:

```
class Report {

 private String reporte;

 public Report(String reporte) {
          this.reporte = reporte;
 }
 void export(File file) {
   if (file == null) {
     throw new IllegalArgumentException(
       "File es NULL; no puedo exportar..."
     );
   }
   if (file.exists()) {
     throw new IllegalArgumentException(
       "El archivo ya existe..."
     );
   }
   // Exportar el reporte a un archivo.
 }
```

```
}
```

a. Implemente la exportación.

b. Utilice el pattern Decorador para reescribir la funcionalidad de Reporte, de modo tal que le permita escribir Reportes que exporten sin verificar si el archivo existe (o sea, lo sobreescriba) y Reportes que no permitan sobrescribir el archivo.

4. Un restaurante de comidas rápidas ofrece 3 tipos de combos (Combo Básico, Combo Familiar, Combo Especial). De cada combo podemos conocer su descripción que nos detalla el contenido del combo, y por otro lado podemos conocer su precio.

El restaurante también ofrece la posibilidad de aumentar el pedido mediante diferentes porciones adicionales (Tomate, Papas, Carne, Queso). Cada porción que se agrega al combo tiene un costo adicional.

Se desea crear un sistema de pedidos que permita al usuario seleccionar el combo deseado, así como armar su propio pedido con las porciones adicionales que desee. El sistema deberá informar sobre el pedido del usuario detallando su descripción y el valor total del mismo.

a) Aplique el patrón **Decorator** para diseñar el modelo de clases que de solución al problema planteado.

b) Implemente la solución en Java, especificando en el programa principal el armado de 2 combos distintos con al menos dos adicionales cada uno.

c) Implemente la creación de los combos utilizando el patron Builder.

5. Aplique decorador al ejercicio 4 del TP 2. Es decir, el email que se envía cada vez que un participante se inscribe en un concurso ahora realizelo implementado un decorador del concurso.