Parcial de OO2 - Curso 2022 - 16/jul/2022

Ejercicio 1 - Patrones

Sea una estación meteorológica hogareña que permite conocer información de varios aspectos del clima. Esta estación está implementada con la clase HomeWeatherStation que interactúa con varios sensores para conocer fenómenos físicos. La misma implementa los siguientes métodos:

```
//retorna la temperatura en grados Fahrenheit
public double getTemperaturaFahrenheit()

//retorna la presión atmosférica en hPa
public double getPresion()

//retorna la radiación solar
public double getRadiacionSolar()

//retorna una lista con todas las temperaturas sensadas hasta el momento, en
grados Fahrenheit
public List<Double> getTemperaturasFahrenheit()
```

Esta clase se encuentra implementada por terceros y **no se puede modificar.**

Nos piden construir una aplicación que además de lo anteriormente descripto pueda obtener:

- La temperatura en grados Celsius (°C = (°F-32) ÷ 1.8).
- El promedio de las temperaturas históricas en grados Fahrenheit.

Además, la aplicación debe permitir al usuario configurar qué datos mostrar y en qué orden. Esto significa que podría querer ver la información de muchas maneras, por ejemplo:

- *Ejemplo 1: "Presión atmosférica: 1008"*
- *Ejemplo 2: "Presión atmosférica: 1008 Radiación solar: 500"*
- *Ejemplo 3: "Radiación solar: 500 Temperatura C: 28 Promedio de temperaturas C: 25"*

Para ello, usted debe proveer en algún punto de su solución, la implementación del mensaje public String displayData() que devuelva los datos elegidos en el orden configurado (dado que la app aun no cuenta con interface de usuario).

Haga uso de la clase HomeWeatherStation sin modificarla.

Tareas:
1- Modele una solución para el problema planteado. Si utiliza algún patrón, indique cuál
2- Implemente en Java
3- Implemente un test para validar la configuración del ejemplo 2, asumiendo que en el momento de la ejecución del mismo, los sensores arrojan los valores del ejemplo.

Ejercicio 2 - Refactoring

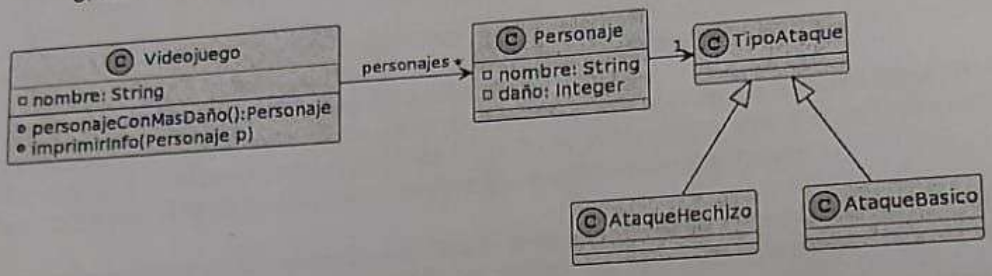
Para el siguiente código, realice las siguientes tareas:

(i) indique qué mal olor presenta

(ii) indique el refactoring que lo corrige

(iii) aplique el refactoring mostrando únicamente el código que cambió, detallando cada paso intermedio.

Si vuelve a encontrar un mal olor, retorne al paso (i).




```
public class Videojuego{
    // ...
    //
public Personaje personajeConMasDaño() {
      Personaje temp = null;
      double max= 0;
      for (Personaje p : personajes) {
            double daño = p.getTipoAtaque().calcularDaño(p.getDaño());
            if (daño > max){
                  temp = p;
                  max = daño;
           }
       }
       return temp;
 }

public void imprimirInfo(Personaje p){
     System.out.println(p.getNombre() + "tiene como daño " + p.getDaño());
     if (p.getTipoAtaque().getClass() == AtaqueHechizo.class) {
          System.out.println("Ataque tipo hechizo");
          System.out.println("Este ataque dobla tu fuerza");
    } else{
         System.out.println("Ataque tipo Ataque Básico");
         System.out.println("Este ataque mantiene tu fuerza");
         }
}
```