Taller Principios SOLID

# Objetivos:

* Identificar violaciones a los principios SOLID.
* Corregir código que viole principios SOLID.
* Utilizar Git como herramienta de apoyo al trabajo colaborativo.

# Instrucciones

* Trabajar en los grupos del proyecto.
* Usted trabajará con el proyecto en un repositorio de Github. El proyecto está compuesto por 10 paquetes (2 por cada principio, mal y bien).
* Realizar un Fork al repositorio del grupo, desde el repositorio llamado: <https://github.com/djurado/solid>
* Cada integrante debe haber solucionado por lo menos uno de principios y debe verse reflejado en el usuario de los commits al repositorio.
* Los paquetes que contienen **‘mal’** en su nombre, contiene clases e interfaces que viola un principio SOLID. Las tres primeras letras del nombre del paquete le indican el principio que se está violando.
* Analice el código y reestructure las clases, de modo que el principio SOLID ya no sea violado. Agregue su propuesta de solución en sub-paquetes **‘bien’** respectivos.

# Conteste

Explique brevemente como cada principio es violado en el código analizado.

* SRP: la habilidad de curar no debe ser realizada por el pokemon por lo tanto será asignada a una nueva clase llamada centroPokemon que tendrá la función de sanar a los pokemon.
* OCP: cree la interfaz ataquePokemon y cada tipo de pokemon implementará su forma de atacar, ya no utilizo los enum porque cada pokemon tiene una lista de objetos de tipo

Atack de esta manera si se quiere agregar un nuevo atack solo se setea al objeto

pokemon el atack que quiere añadir a su lista de arsenal.

Y ya no es necesario cambiar ninguna clase

* LSP: En la moto acuática cuando le mandamos la función navegar, si esta no estaba encendida nos lanzaba una excepción. Para arreglarlo a la hora de mandar a navegar la moto acuática debemos encenderla para que pueda navegar.
* ISP: Debido a que existe solo una interfaz que implementan todas las clases no abstractas, no quiere decir que cada una de las clases tenga que implementar todos métodos adquiridos. Como solución se decidió hacer 3 interfaces ya que una era diseñada específicamente para Pokemones tipo fuego, otra interfaz para Pokemones tipo agua y una tercera interfaz para cualquier tipo de Pokemon.
* DIP: Para que el programador tenga total independencia para elegir que computadora elegir, se creó una interfaz computadora, para que otras clases la puedan implementar, así que en el constructor se le envía como parámetro que clase de computadora es, para poderla instanciar.

# Entregable

* El enlace al repositorio de GitHub donde colocaron su solución a cada principio y este archivo Word con su análisis grupal.
* Subir el enlace a Sidweb.