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:
* OCP:

El principio es violado por que se estaban haciendo modificaciones en el proyecto, lo cual el principio dice que eso no es aceptable, lo que el principio dice es que nuestro proyecto debe de ser abierto a extensiones, pero cerrado a modificaciones. Cada vez que se añadía un nuevo ataque se estaba modificando el proyecto, añadiéndole líneas de códigos para poder realizar lo que hace el ataque en la clase ManejadorAtaque.java

* LSP:

Este principio es violado debido a que una clase derivada de la interfaz tiene una funcionalidad adicional que la clase “padre” no maneja, que es el hecho de que la moto acuática debe estar encendida para navegar, haciendo que el programa se caiga si una moto acuática no está encendida.

* ISP:

Se implementan métodos que las clases PokemonFuego y PokemonAgua no necesitan, esto se debe a que todos los métodos están agrupados en una sola interface, esa es la violación de este principio en esta parte del código.

* DIP:

En la clase programador se estaba creando el objeto ComputadoraWindows dentro del constructor vacío de programador, lo cual estaba violando el principio de DIP, ambos deberían de depender de la abstracción.

# 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.