Taller Principios

SRP – OCP – LCP

**(Single Responsibility Principle** **/ Open Closed Principle / Liskov Principle)**

# Objetivos:

* Identificar incumplimiento a los principios SRP – OCP – LCP.
* Corregir código que incumpla los principios SRP – OCP – LCP.
* Utilizar Github como herramienta colaborativa para cargar los proyectos.
* Lengua de programación JAVA

# Instrucciones

* En taller se debe realizar individualmente.
* Usted trabajará con el proyecto en un repositorio de Github. El proyecto está compuesto por 6 paquetes (2 por cada principio, “Wrong” y “Good”).
* Descargue el proyecto desde el repositorio llamado: https://github.com/neortiz1
* Los paquetes que contienen **‘mal’** en su nombre, contiene clases e interfaces que incumplen un principio. Las tres primeras letras del nombre del paquete le indican el principio que se está incumpliendo.
* Analice el código y reestructure las clases (Refactorizar), de modo que el principio SOLID se esté cumpliendo. Agregue su propuesta de la solución en los sub-paquetes **‘Good’** respectivos.

# Conteste

Explique brevemente como cada principio se incumplió en el código analizado.

Nombre: Reishel Tipán

* SRP:

La clase “Vehicle” tiene atributos que indican que esta clase tiene la responsabilidad de avisar si se debe repostar (llenar de nuevo un depósito de combustible) el vehículo. Sin embargo, la función “reFuel” hace que cambie la forma de repostar combustible y por ello la clase no cumple el Single Responsibility Principle.

* OCP:

El Open Closed Principle se incumple ya que en el código de la clase “EventHandler” o se encuentran valores.

Además, también se incumple ya que si tenemos que añadir un nuevo modo de conducción, deberemos añadir el nuevo modo en la enumeración y además deberemos modificar el método changeDrivingMode(DrivingMode drivingMode) para tener en cuenta este nuevo modo.

* LSP:

Para cumplir el Liskov Principle deberíamos poder utilizar las clases Duck y Ostrich en lugar de la superclase Bird. Debido a que no se cumple este principio no se puede usar de forma indistinta la superclase o las subclases sin generar errores en la aplicación ya que la subclase Ostrich tiene restricciones a la superclase en el método fly().

# Entregable

* Enlace al repositorio de GitHub donde colocaron su solución a cada principio y este archivo Word.