Taller Principios
SRP – OCP – LCP

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

Nombre: Alex Trejo

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.

- SRP: La clase “Vehicle” realizaba más acciones de las que le correspondían, en otras palabras, la cohesión era baja. Por lo que se debe mudar los métodos que no corresponden a otra clase.

```
// this is not a car's responsibility.
/*public void reFuel() {
    remainingFuel = maxFuel;
}*/
```

- OCP: En la clase “EventHandler” de forma inicial era de tipo cerrada/cerrada por lo que cuando queramos agregar otro modo de manejo se deberán cambiar las clases “DrivingMode “ y “EventHandler”. Por lo que se debe dividir en otras clases para permitir que puedan acceder a los métodos y atributos y que se puedan modificar.

```
void changeDrivingMode(final DrivingMode drivingMode) {
    switch (drivingMode) {
        case SPORT -> {
            vehicle.setPower(500);
            vehicle.setSuspensionHeight(10);
        }
        case COMFORT -> {
            vehicle.setPower(400);
            vehicle.setSuspensionHeight(20);
        }
        case ECONOMY -> {
            vehicle.setPower(200);
            vehicle.setSuspensionHeight(30);
        }
    }
}

enum DrivingMode {
    SPORT, COMFORT, ECONOMY
}
```

- LSP: La clase de “Ostrich” no puede heredar todos los métodos de los clase “Bird”, que en este caso es “fly”, en otras palabras la clase hija no puede comportarse como una superclase.

```
public class Ostrich extends Bird {

    @Override
    void eat() {
        System.out.println("Ostrich eating");
    }

    @Override
    void fly() {
        System.out.println("Ostriches can't fly");
    }
}
```

Entregable

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