**Practica 2: Arquitectura software**

**Enlace github:** https://github.com/JorgeVanco/practica2\_paradigmas\_programacion.git

**Principios SOLID en el código:**

* SRP:

En un principio el método StartPersecution de la clase PoliceCar también alertaba a la comisaría si no había sido alertado por ella el propio coche. Sin embargo, esto no cumple Single Responsibility Principle porque el método hace más cosas de las que dice su nombre.

Texto

Descripción generada automáticamente

Esto has sido modificado a dos métodos, uno que es AlertPoliceStation, que manda la alerta a la estación de policías y StartPersecution, que simplemente hace que el coche entre en la persecución. Este último método es llamado por la comisaría.

Texto

Descripción generada automáticamente

* OCP:

Para cumplir este principio se ha tenido que tener especial cuidado en el método TriggerRadar de la clase SpeedRadar, pues con el nuevo Scooter, se tiene que tener en cuenta el caso en el que es un vehículo sin motor.

Texto

Descripción generada automáticamente

* LSP:

Se ha creado una clase VehicleWithPlate que hereda de Vehicle para diferenciar entre los vehículos con y sin matrícula cumpliendo el Liskov Substitution Principle.

* ISP:

Este se cumple porque solo hay una interfaz con un método, todas las clases que la implementan necesitan el método.

* DIP:

No hay ninguna clase que dependa de los requisitos de otra para su funcionalidad.

**7. Policía tiene un único medidor**

Tal y como está el código, se incumpliría el Open Closed Principle, puesto que el policía usa directamente el radar para obtener las medidas. Además, habría que cambiar el nombre del método a UseDevice o algo similar para no incumplir Single Responsibility Principle, porque podría medir el nivel de alcohol además de la velocidad.

**UML:**

