P2 – Arquitectura de Software

**PRINCIPIOS SOLID:**

* Single responsability principle: sí que se cumple este principio porque cada clase hace la tarea para la que se ha creado. No hay ninguna clase que haga más de lo que debe. Este principio es útil para que las clases no hagan demasiadas cosas a la vez. Si hay varias cosas gestionándose a la vez, la probabilidad de error o fallo aumenta.
* Open/close principle: sí que se cumple porque se utilizan herencias para extender la funcionalidad del código (por ejemplo: Vehicle y VehicleWithPlate) sin modificar el código de la base, es decir, del padre. Se necesita para que el sistema sea escalable sin necesidad de tener que modificar el código que ya está reduciendo, una vez más, errores.
* Liskov Substitution Principle: Sí que se cumple porque al utilizar las herencias estas no modifican el comportamiento de la clase padre. Para poder lograr esto se ha tenido que hacer una clase intermedia entre Vehicle y Taxi/PoliceCar/Scooter para poder diferenciar aquellos vehículos que sí tienen matrícula y los que no. De esta manera se consigue que la abstracción de la clase Vehicle sea suficiente. Se aplica sobre todo para que las clases hijo mantengan la coherencia con la clase base.
* Interface Segregation Principle: sí que se cumple porque cada clase implementa los métodos que necesita y no más. Este principio favorece la comprensión y mantiene el código más limpio.
* Dependency Inversion Principle: sí que se cumple porque todas las clases dependen o de la interfaz o de una clase abstracta (en este caso, Vehicle). De esta manera desaparecen, en la medida de lo posible, las dependencias en el sistema y lo hacen más flexible y fácil de mantener.

7. Se podría decir que se incumplen dos de los principios SOLID:

* Liskov Substitution Principle: Si se añade un alcoholímetro habría que cambiar el código de la clase PoliceCar para permitir que se pudiesen usar cualquiera de los medidores. Para solucionar esto se podría crear una clase abstracta medidor que fuese el padre de ambos medidores (alcoholímetro y radar).
* Single responsability principle: la clase PoliceCar tendría más de una responsabilidad en el código, incumpliendo así este principio SOLID. Para poder solucionar esto, en lugar de tener un método para cada medidor que se añada, se tendría un método general “UseMeasuringDevice()” por ejemplo, y utiliza ese medidor en fusión del coche que lo contenga.