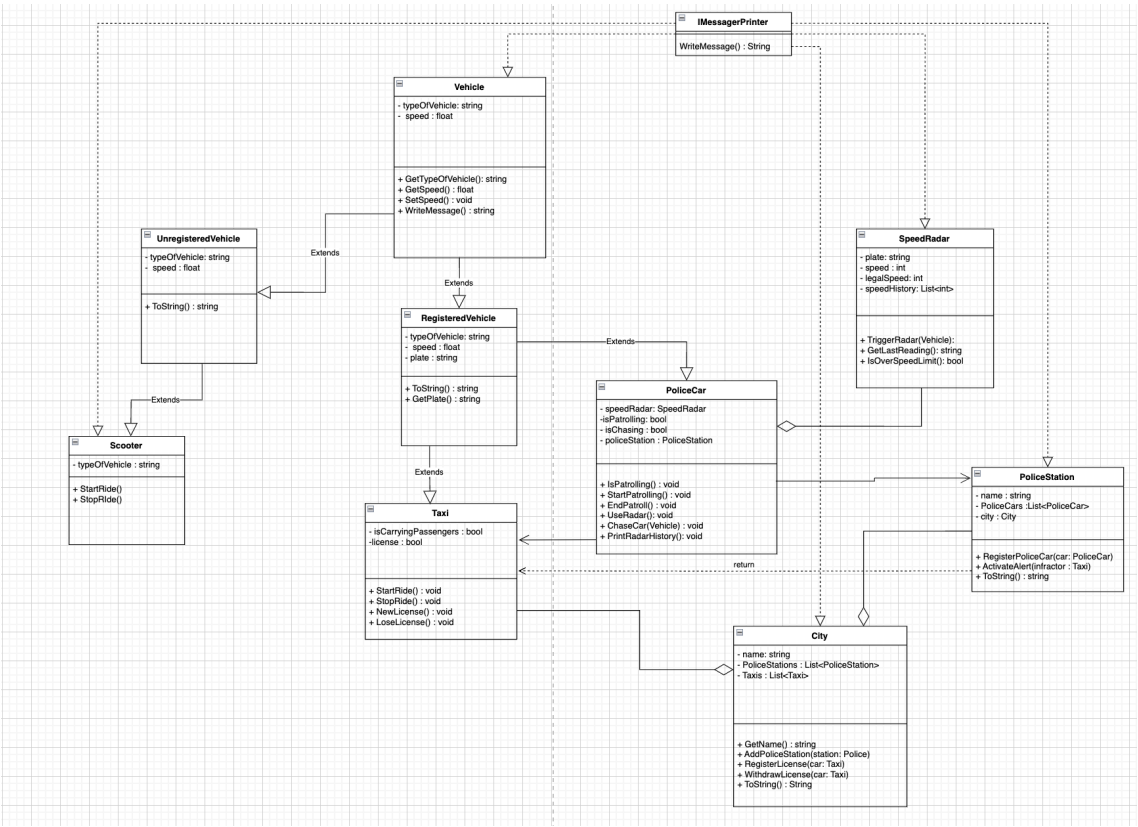
UML.



6 . Explicar brevemente si se están aplicando los principios SOLID dentro del código y el porqué de esa implementación, hacer las modificaciones necesarias en el código y en la arquitectura para que se cumplan los principios. Redactar en el mismo documento donde se incluya la arquitectura UML.

Single Responsibility Principle (SRP): Cada clase tiene una única responsabilidad. Es verdad que hay clases como las de coche de policía que se podría decir que tienen varias funcionalidades como disparar radares y perseguir a coches, pero al final son funcionalidades que están directamente relacionadas. Por otro lado las demás lo cumplen bastante bien.

Open/Closed Principle (OCP): Las clases están abiertas para la extensión pero cerradas para la modificación. Por ejemplo, al agregar nuevos tipos de vehículos (como Scooter), no es necesario cambiar las clases existentes, solo se extiende la funcionalidad.

Liskov Substitution Principle (LSP): Las clases derivadas deben poder sustituir a sus clases base. Taxi y PoliceCar son tipos de RegisteredVehicle, lo que significa que pueden ser tratados de manera intercambiable sin romper la funcionalidad del programa.

Interface Segregation Principle (ISP): Las interfaces deben ser específicas para los clientes que las utilizan. Con IMessageWritter la utilizamos para todas las clases ya que en todas queremos que cuando ejecute una acción ya sea disparar un radar o activar una alarma, queremos que esta instancia aparezca por pantalla seguido por dos puntos de la acción que procede a hacer.

Dependency Inversion Principle (DIP): Aquí no tenemos clases de alto nivel dependiendo de clases de bajo nivel, aunque pueda parecer que City depende de PoliceStation lo cierto esque para llevar a cabo el programa es vital que estas dos esten estrechamente relacionadas

7. Ahora queremos que el policía pueda tener diferentes aparatos de medida, que pueden ser un medidor de velocidad (Radar) o un medidos de alcohol (Alcoholímetro). Al coche de policía solo se le puede asignar un único medidor, y en el coche de policía únicamente va a haber un método para activar el aparato de medida. Con la arquitectura actual, ¿Qué principio SOLID incumpliríamos y cómo lo solucionarías? Redactar en el mismo documento donde se incluya la arquitectura UML, no será necesario implementarlo en el código.

Para no incumplir LSP tendríamos que crear una clase instrumento de medición y que la clase Radar y alcoholímetro hereden esta y que se elija cual se usa en el constructor por ejemplo. Ya que si creamos variables radares y alcoholímetros las dos en PoliceCar, habrá coches que no puedan medir el alcohol y otros que no puedan medir la velocidad cosa que damos por hecho que hace la clase PoliceCar