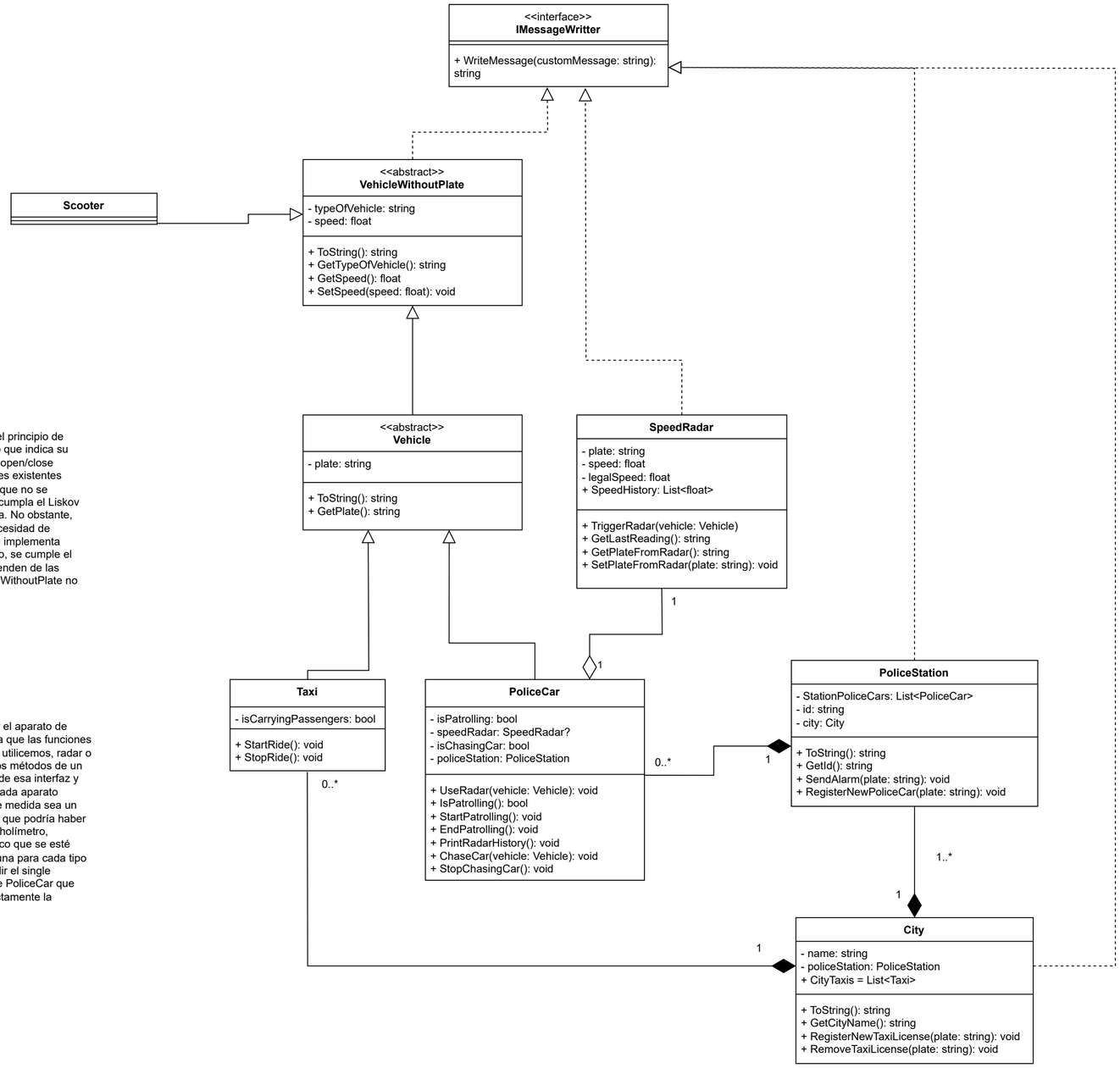

Creo que sí se están aplicando los principios SOLID. En primer lugar, el principio de single responsibility se estaría respetando ya que cada método hace lo que indica su nombre, sin ninguna otra funcionalidad adicional. En segundo lugar, el open/close principle se estaría respetando porque no ha habido que tocar las clases existentes para ampliar las funcionalidades del código. Solo habría un caso en el que no se cumple este principio que es modificando el código de forma que no incumpla el Liskov Sustitution Principle al introducir el patinete como vehículo sin matrícula. No obstante, esto solo hace que se puedan introducir más tipos de vehículos sin necesidad de modificar el código base. También se respeta el ISP porque cada clase implementa únicamente lo que necesita para su correcto funcionamiento. Por último, se cumple el Dependency Inversion Principle porque las clases de alto nivel no dependen de las clases de más bajo nivel. En nuestro caso, las clases Vehicle y VehicleWithoutPlate no dependen de las clases Taxi o PoliceCar, más bien es a la inversa.

Si 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, estaríamos incumpliendo el Dependency Inversion Principle ya que las funciones del aparato de medida dependerían del tipo de aparato de medida que utilicemos, radar o alcoholímetro. Se podría solucionar creando una interfaz que incluya los métodos de un aparato de medida en general y que cada uno de los aparatos herede de esa interfaz y sobreescriba los métodos según las acciones que debe llevar a cabo cada aparato específico. Además, si ahora damos la posibilidad de que el aparato de medida sea un alcoholímetro, se podría incumplir el Interface Segregation Principle ya que podría haber acciones en la misma interfaz propias de un radar y propias de un alcoholímetro, implementando métodos que puede que el aparato de medida específico que se esté usando en cada momento no posea. Se podrían crear dos interfaces, una para cada tipo de aparato de medida para solucionar esto. También se podría incumplir el single responsibility principle ya que el nombre del método que posea la clase PoliceCar que controle el aparato de medida correspondiente puede no ilustrar correctamente la funcionalidad específica de alguno de los dos aparatos de medida.

URL Repositorio: https://github.com/tgarviagallego/Practica2_Paradigmas.git