* **Polimorfismo:** ¿Cuál es el mensaje polimórfico? ¿Qué objetos lo implementan? ¿Qué objeto se aprovecha de ello?

Utilizamos el **polimorfismo** en el mensaje choque(unAuto) y los objetos que lo implementas son los vehículos que pueden ser tanto el auto rojo como el camión. El objetivo por el cual aplicamos el polimorfismo en estos objetos es que el mensaje choque produjera un resultado diferente según el objeto con el que interactuaba en la colisión, sin tener que preocuparnos por lo que hace el auto cuando colisiona con un objeto.

* **Colecciones:** ¿Qué operaciones de colecciones se utilizan? ¿Usaron mensajes con y sin efecto? ¿Para qué?

Las colecciones fueron implementadas, una para crear los diferentes obstáculos ,eliminar los que ya no estaban en el tablero y otra para guardar los puntos críticos de la pista ya que nos evitamos utilizar un if con varios Ors.Las operaciones que utilizamos fueron contains(contador), get(), add(), any(), filter(), first(),remove() y forEach().

Si, utilizamos mensajes con efecto en la lista de obstáculos para agregar nuevos obstáculos e ir eliminando los que ya no estaban en el tablero. Y usamos los mensajes sin efecto para consultar si determinado obstáculo se encontraba en la lista o en el caso de la lista de puntos críticos de la pista para saber si el contador estaba dentro de la lista ara saber que se debe realizar, o en la lista de obstáculos si había uno que cumpliera una determinada condición.

* **Clases**: ¿Usan clases? ¿Por qué? ¿Dónde o cuándo se instancian los objetos?

Implementamos **clases** para definir los distintos obstáculos que aparecen aleatoriamente en el tablero y colisionan con el auto, ya que eran demasiados obstáculos implementados con solo un par distintos y luego todas “copias” de ellos. Las clases de objetos se instancian en el archivo de creadoresObstaculos.wlk cuando definimos un objeto el cual crea un obstáculo, comenzando el juego comienzan a instanciarse los obstáculos y finalizan junto con el juego.

* **Herencia:** ¿Entre quiénes y por qué? ¿Qué comportamiento es común y cuál distinto?

A la **herencia** la usamos al crear los objetos auto rojo , camión y la colisión a partir de lo heredado por vehículo, debido a que tenían varios métodos y también variables en común, con esto logramos evitar la repetición de código y el tener que acordarse que funciones y variables habría que definir para considerar al objeto un vehículo. Los comportamientos distintos son los métodos relacionados con el choque del vehículo como explotar(), destruirse(), ganar() (el resultado de la colisión del obstáculo con el vehículo), los métodos que crea cada vehículo y además la modificación de la variable estado del auto rojo ya que el camión no cuenta con esto. Y los componentes que compartían son las variables posición, imagen, los métodos position (), imagen () y moverseA(nuevaPosicion).

También utilizamos la **herencia** al definir, los obstáculos que son planta, aceite, barril, piedra, vida y meta que heredan métodos y variables de la clase obstáculo. Lo implementamos debido a que todos estos obstáculos tenían la misma estructura solo cambiaba un método que es el de choque (unAuto) y con esto nos ahorramos código y evitamos la repetición de lógica, a las variables imagen, posición y nombre todos las implementas y además la clase obstáculo cuanta con métodos como position() , position(nuevaPosicion),imagen()y estaFuera() que son comunes a todas.

* **Composición:** ¿Qué objetos interactúan? ¿Dónde se delega? ¿Por qué no herencia?

Utilizamos composicion para definir los estados del auto, los objetos que interactúan son autoRojo, autoPrimerChoque, autoChocado, autodestruido, autoRoto.

Se delega en los métodos que están dentro del objeto auto rojo como explotar(), destruirse(),reparar() y cambiar estado(), y decidimos no utilizar herencia ya que necesitábamos que el estado del auto se fuera agravando, pasando de un estado a otro pero sabiendo el valor del estado anterior para así saber a qué estado cambiar después.