* **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 camión. El objetivo con el que aplicamos polimorfismo en estos objetos es producir 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.

También usamos **polimorfismo** en los estados de los vehículos a través de las funciones afectar (vehículo) y desafectar (vehículo), las cual nos permite abstraernos de la función de cada estado.

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

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

Si, utilizamos mensajes con efecto en la lista de obstáculos para agregar nuevos e ir eliminando los que ya no estan en el tablero, lo implementamos con add() y remove(),funciones con efecto(). Usamos los mensajes sin efecto para consultar si determinado obstáculo se encuentra en una lista ,o para saber si el contador estaba dentro de la lista de conflictos y así saber que se debe realizar , en la lista de obstáculos uno cumplía una determinada condición, a esto lo implementamos con contains(), first(),any().

* **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, porque son demasiados obstáculos implementados con solo unas pocas diferencias en sus métodos y constantes o variables. Las clases de objetos se instancian cuando comienza el juego y finalizan junto con él.

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

A la **herencia** la usamos al crear los objetos auto rojo y camión, a partir de lo heredado de vehículo, debido a que tienen varios métodos y variables en común. Así logramos evitar la repetición de código y el tener que acordarse que funciones y variables se tendría que utilizar para considerar al objeto un vehículo. Los comportamientos diferentes son métodos relacionados con el choque del vehículo como explotar (), destruirse (). Y los componentes son las variables posición, imagen, estado, los métodos position (), imagen () y moverseA(nuevaPosicion),gana(),perderControl().

También utilizamos la **herencia** al definir, los obstáculos como planta, aceite, barril, piedra, vida y meta, que heredan métodos y variables de la clase obstáculo. Se implementó debido a que todos los obstáculos tienen la misma estructura y solo un método diferente ,choque (unAuto) , así evitamos la repetición de lógica. Las variables imagen, posición y nombre son comunes a todas y la clase obstáculo cuanta con métodos como position() , estado(),cambiarEstado(nuevoEstado),imagen(nuevaImagen),imagen()y estaFuera() que son comunes a todas.

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

Utilizamos composición para definir los estados del auto, los objetos que interactúan son primerChoque, chocado, destruido, roto.

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