FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — GPS Challenge

[75.07/95.02] Algoritmos y Programación III
Primer cuatrimestre de 2022

Alumno	Padron	Email
CASTILLO, Carlos	108535	ccastillo@fi.uba.ar
DEALBERA, Pablo Andres	106585	pdealbera@fi.uba.ar
RECCHIA, Ramiro	102614	rrecchia@fi.uba.ar

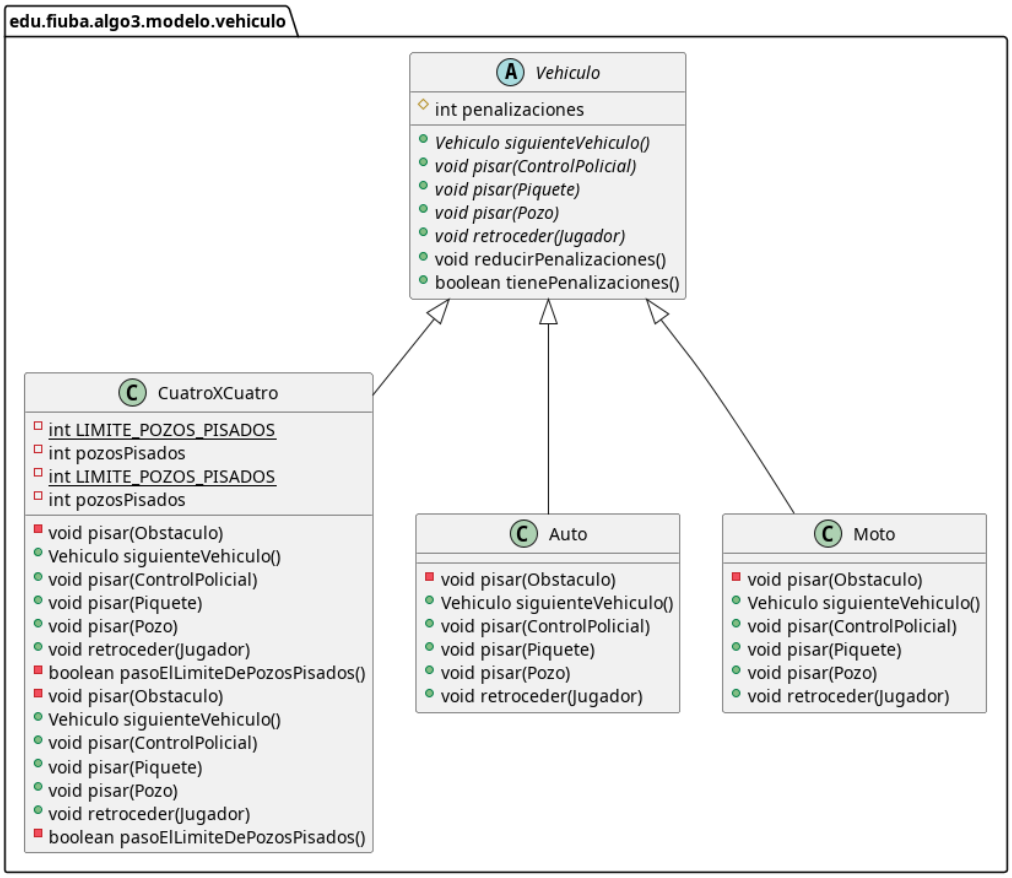
Corrector	Email
GOMEZ, Joaquin	gjoaquin@fi.uba.ar
VALDEZ, Santiago	vsantiago@fi.uba.ar

Índice

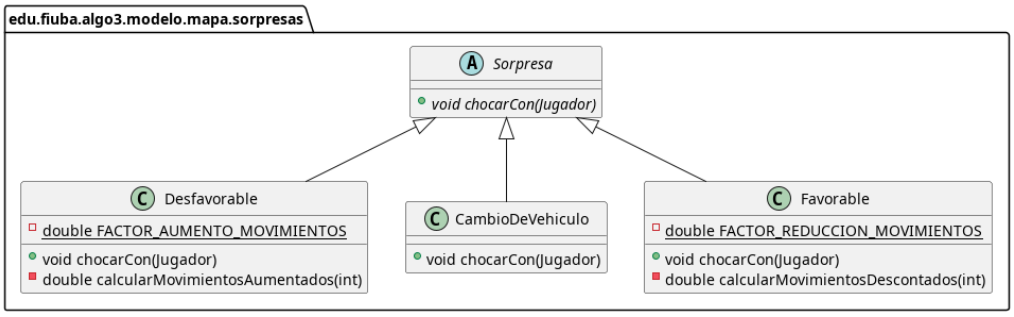
1. Supuestos

2. Diagramas de clases

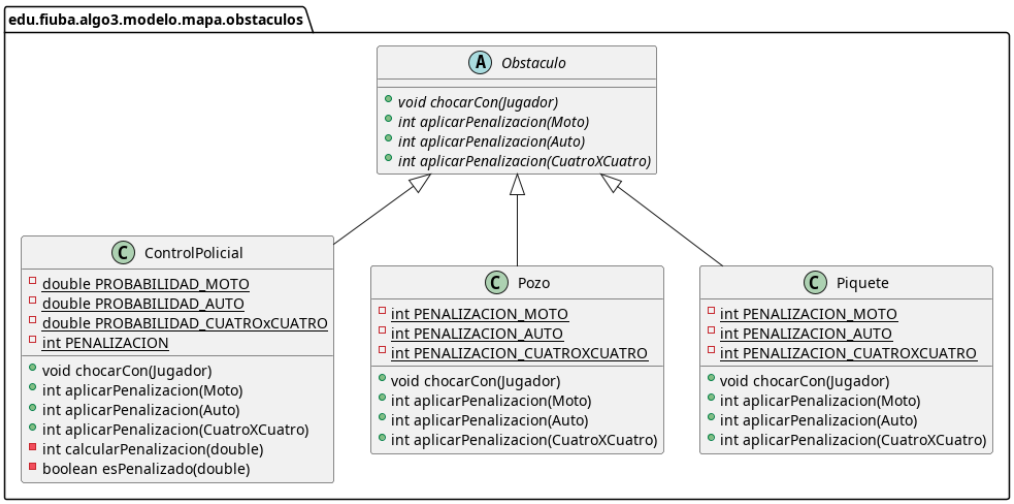
2.1. Vehiculo



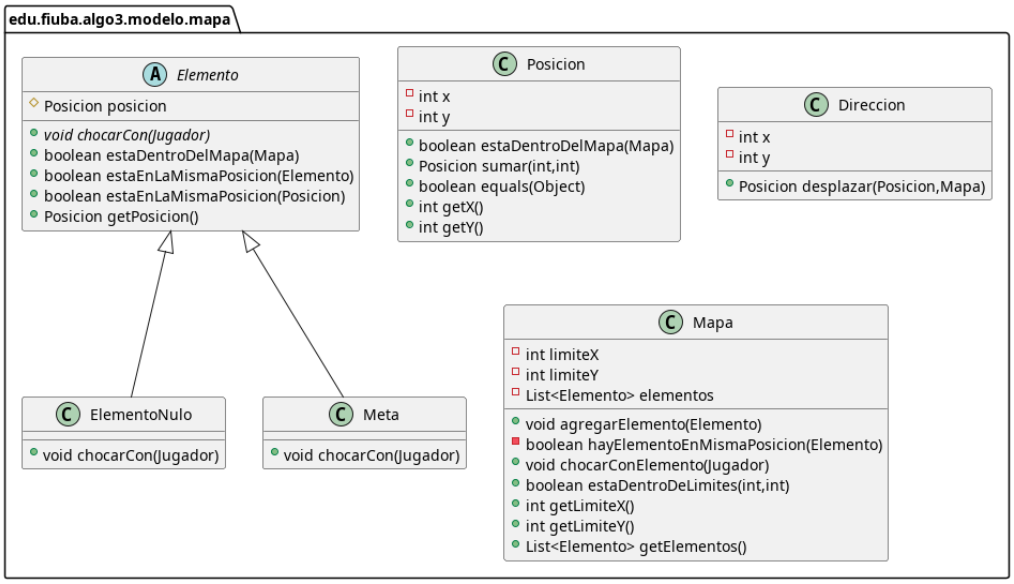
2.2. Sorpresas



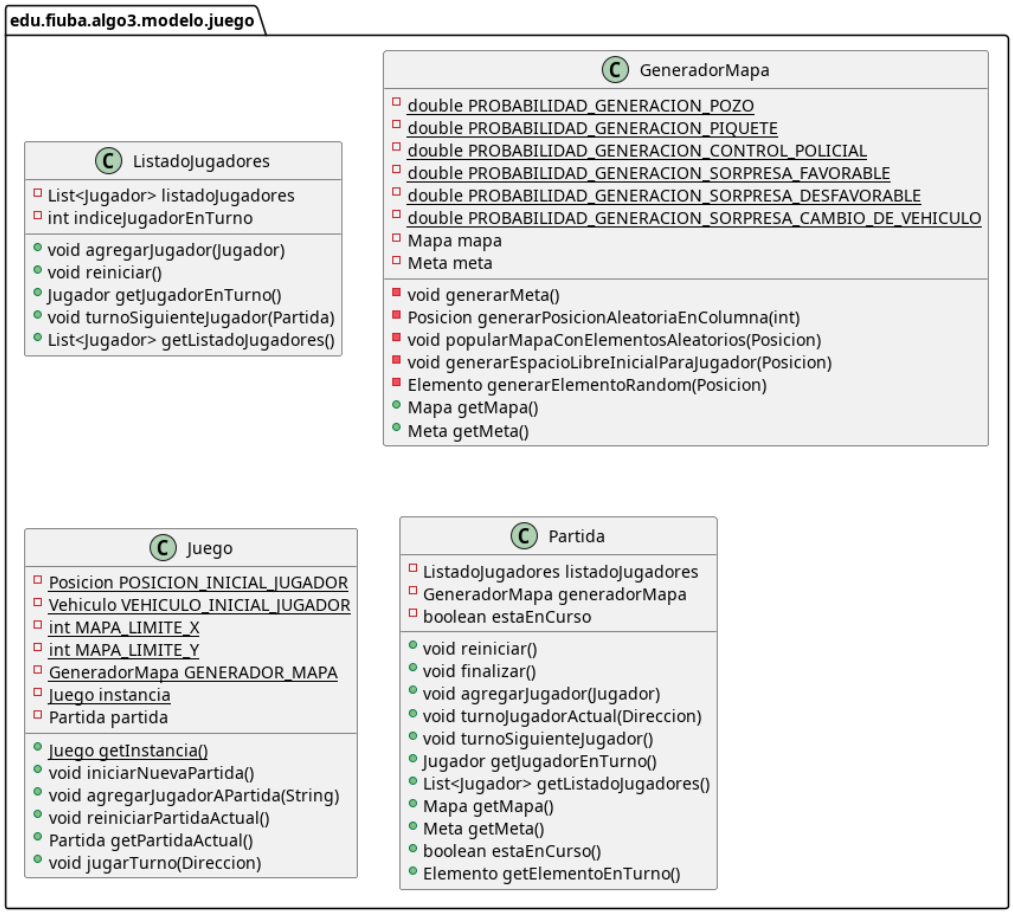
2.3. Obstaculos



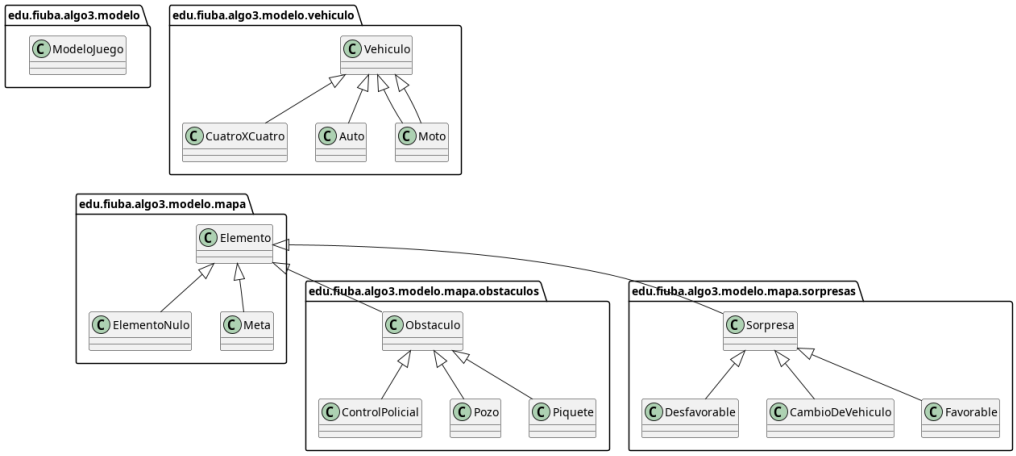
2.4. Mapa



2.5. Juego

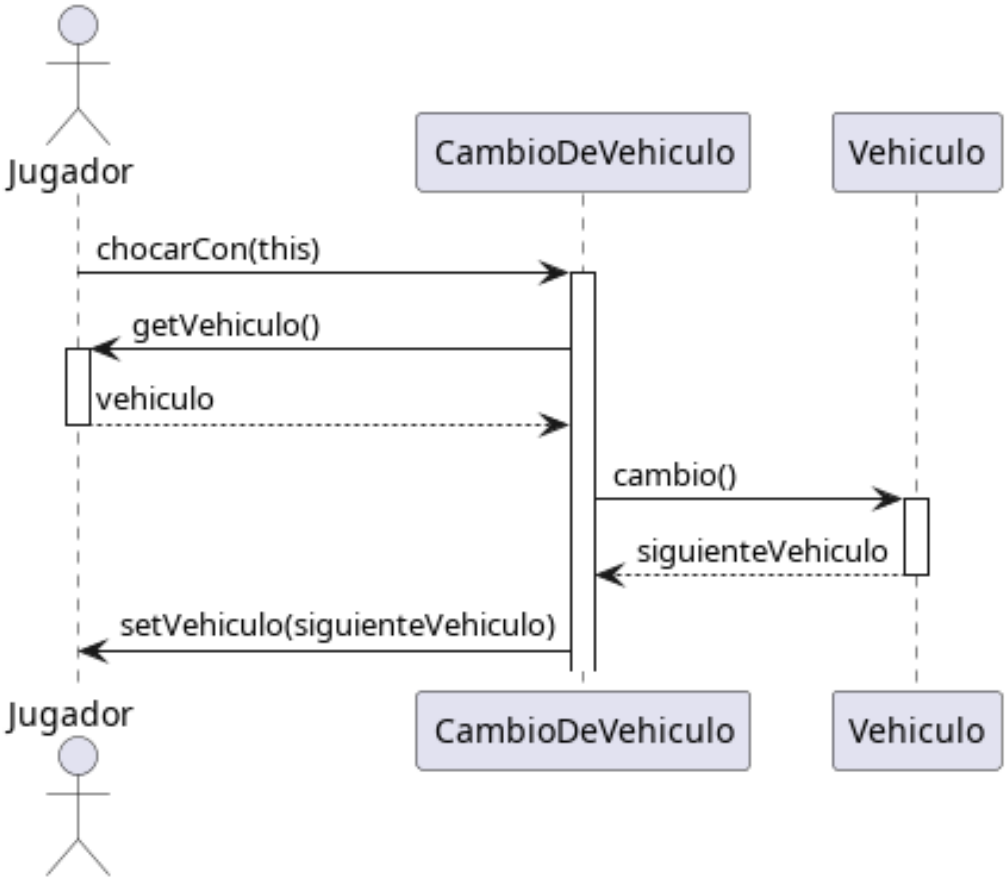


3. Diagrama de paquetes

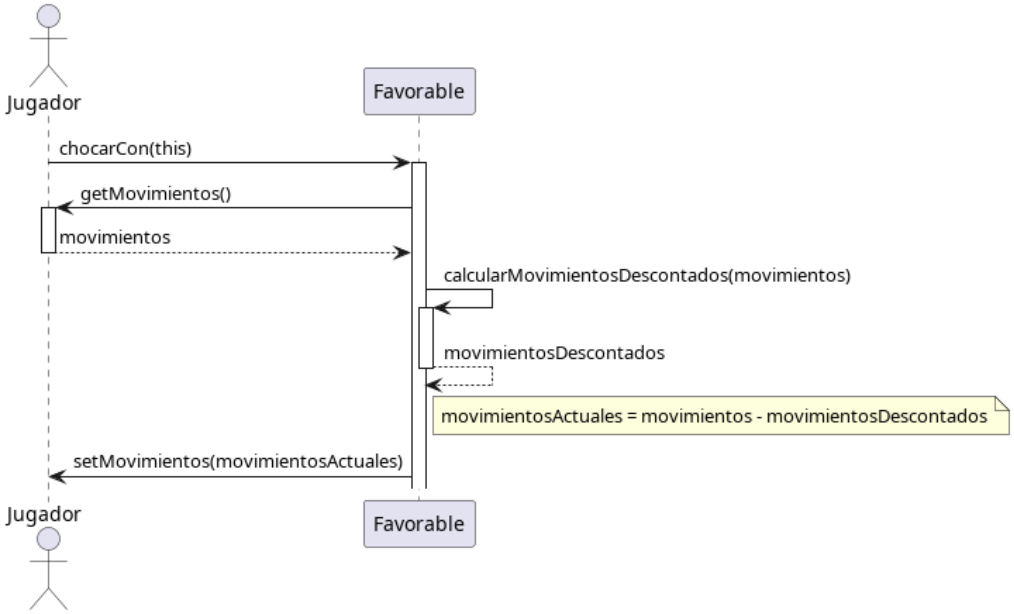


4. Diagramas de secuencia

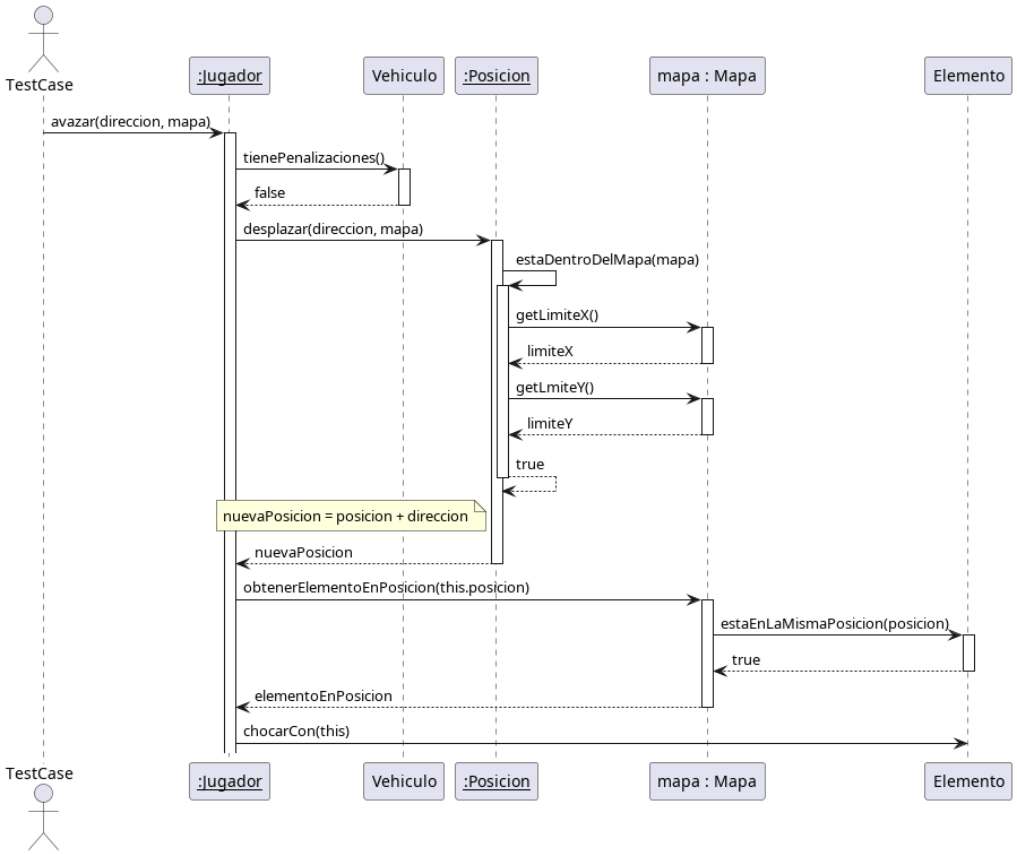
4.1. Interaccion Jugador - Sorpresa Cambio de Vehiculo



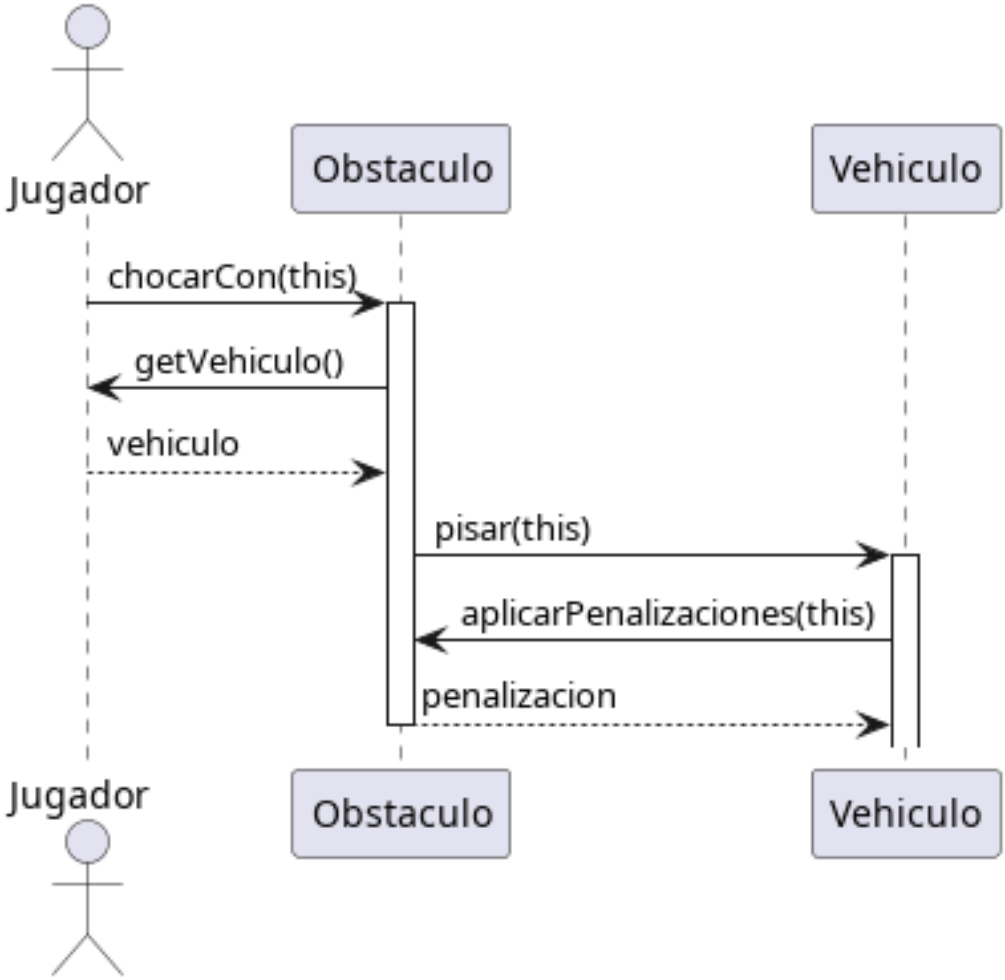
4.2. Interaccion Jugador - Sorpresa Favorable



4.3. Interaccion Jugador - Elemento



4.4. Jugador avanza y se encuentra con un Elemento



5. Diagramas de estado

6. Detalles de implementación

6.1. Vehiculo

En principio tenes una clase abstracta llamada *Vehiculo* y usamos herencia para abstraer comportamiento comun entre su tres clases hijas: Moto, Auto y CuatroXCuatro.

6.2. Elemento

Es una clase abstracta de la cual heredan dos clases:

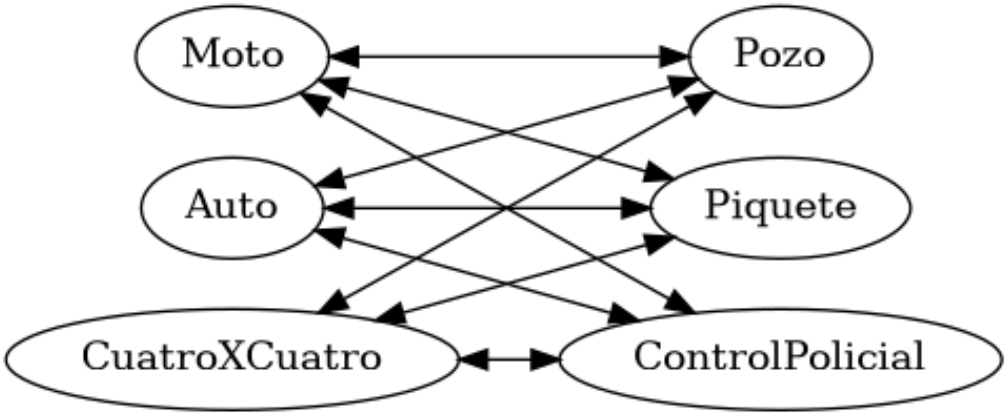
- Obstaculo
  - Pozo, Piquete y Control Policial.
- Sorpresa
  - Favorable y Cambio de Vehiculo

- Meta

Utilizamos esta clase para definir compotamientos que los distintos Elementos tienen en comun, como por ejemplo que pueden `chocaCon` un jugador, y algunas funciones de ayuda para saber si el elemento esta adentro del mapa, si esta en la misma posicion que otro elemento o una posicion arbitraria, etc.

6.3. Interaccion Vehiculo-Obstaculo

Para la interaccion Vehiculo-Obstaculo decidimos usar el patron *Double Dispatch* de forma ya que tenemos una interaccion de muchos a muchos entre los hijos de ambas clases abstractas:



Ademas de esto teniamos la necesidad de modelar implementaciones especificas como el caso de CuatroXCuatro-Pozo donde la CuatroXCuatro debe pisar tres pozos para recibir una penalizacion, cosa que no sucede en ninguna de las otras interacciones.

Para esto los Vehiculos tienen firmas segun cada implementacion de Obstaculo. Y cada implementacion de Obstaculo tiene firmas para cada Vehiculo.

6.4. Ranking y Persistencia

Para el ranking usamos un `HashMap<String, Long>` con el que almacenamos como clave el nombre del jugador y como valor la cantidad de movimientos minimo que hizo.

Esto se maneja en el `ControladorHistorialPartidas` que tiene dos metodos que hacen uso de la libreria Gson para crear un JSON del `HashMap`, escribirlo en un archivo `ranking.json` y otro metodo para obtener el historial en siguientes ejecuciones del programa *deserializando* el JSON parseado como un `HashMap` de nuevo.

```
{
  "Pablo": 24,
  "Ramiro": 20,
  "Carlos": 30
}
```

7. Excepciones

7.1. (No) Excepcion cuando el usuario intentar ir fuera del Mapa

Una observacion que tuvimos durante el desarrollo fue la posibilidad de agregar una excepcion cuando el usuario intenta ir fuera de los bordes del mapa. Nosotros como supuesto elegimos no tratar esto como un error y directamente el modelo maneja esta posibilidad y no permite al usuario avanzar por fuera de los limites del mapa.