FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — GPS Challenge

[7507/9502] Algoritmos y Programación III
Curso 2
Primer cuatrimestre de 2022

Alumno	Padrón	Correo
IGLESIAS, Tomas	105995	tiglesias@fi.uba.ar
PUCCAR, Nicolas	107635	npuccar@fi.uba.ar
LARDIEZ, Mateo	107992	mlardiez@fi.uba.ar
IANNIELLO, Rocio	107414	rianniello@fi.uba.ar
CABRERA, Mauricio	101334	mcabrera@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de manera grupal aplicando todos los conceptos vistos en el curso, utilizando un lenguaje de tipado estático (Java) con un diseño del modelo orientado a objetos y trabajando con las técnicas de TDD e Integración Continua.

2. Supuestos

Para este modelo se tomaron las siguientes decisiones arbitrarias:

- Siempre habrá un y solo un jugador por partida
- La meta siempre aparecerá en una de las esquinas pertenecientes a la última columna del mapa (tomando como última columna la que se encuentra más a la derecha de todas)
- Puede haber dos interferencias entre dos esquinas
- Puede haber como máximo 10 interferencias en todo el mapa
- Si un vehiculo intenta desplazarse a través de un obstáculo, se le acumulará los movimientos correspondientes independientemente de si logre atravesarlo o no. *Por ejemplo, si un auto intenta atravesar un piquete, no se desplazará y se le sumará 1 a la cantidad de movimientos totales hasta el momento*

3. Modelo de dominio

Gameplay GPS es un juego de estrategia por turnos. El escenario es una ciudad y el objetivo, guiar un vehículo a la meta en la menor cantidad de movimientos posibles.

El juego se jugará por turnos, y en cada turno el usuario decide hacia cual de las 4 esquinas posibles avanzará.

Vehículos El jugador podrá optar por tres diferentes tipos de vehículos (Auto, Moto, 4x4). Todos podrán desplazarse por el mapa pero reaccionarán de manera distinta ante los obstáculos

Obstáculos Al atravesar una esquina el jugador se podrá encontrar con alguno de los siguientes obstáculos:

- Pozos: Le suma 3 movimientos de penalización a autos y motos. Para una 4x4 penaliza en 2 movimientos luego de atravesar 3 pozos.
- Piquete: Autos y 4x4 deben pegar la vuelta, no pueden pasar. Las motos pueden pasar con una penalización de 2 movimientos.
- Control Policial: Para todos los vehículos la penalización es de 3 movimientos, sin embargo la probabilidad de que el vehículo quede demorado por el control y sea penalizado es de 0,3 para las 4x4, 0,5 para los autos y 0,8 para las motos ya que nunca llevan el casco puesto.

Sorpresas También se podrán encontrar diferentes tipos de sorpresas, las cuales figuran en el mapa como un regalo y no se sabrá qué es hasta que el vehículo la accione:

- Sorpresa Favorable: Resta el 20 % de los movimientos hechos.
- Sorpresa Desfavorable: Suma el 25 % de los movimientos hechos.
- Sorpresa Cambio de Vehículo: Cambia el vehículo del jugador. Si es una moto, la convierte en auto. Si es un auto lo convierte en 4x4. Si es una 4x4 la convierte en moto.

Escenario El jugador no podrá ver más que dos manzanas a la redonda de la posición de su vehículo y la bandera a cuadros que marca la meta. El resto del mapa permanecerá en sombras.

El tamaño del escenario no será fijo, y tendrá un punto de partida y una meta.

Significado de los objetos en el escenario:

Interferencias

Control policial: cuadrado azul con estrella
Piquete: cuadrado rojo con lineas
Pozo: cuadrado marrón con círculo
Meta: cuadrado blanco

Sorpresas

Sorpresa cambio de vehículo: cuadrado amarillo con el dibujo de un auto
Sorpresa desfavorable: cuadrado naranja con una cara enojada
Sorpresa favorable: cuadrado verde con una cara feliz con estrellas

Vehículos

Moto: cuadrado violeta con letra M
Auto: cuadrado rosa con letra A
4x4: cuadrado negro con un 4

Figura 1: Ejemplo de escenario como lo visualiza el jugador.

Puntajes altos Se almacenará un ranking donde figuren los mejores puntajes asociados a un nickname que indique el usuario.

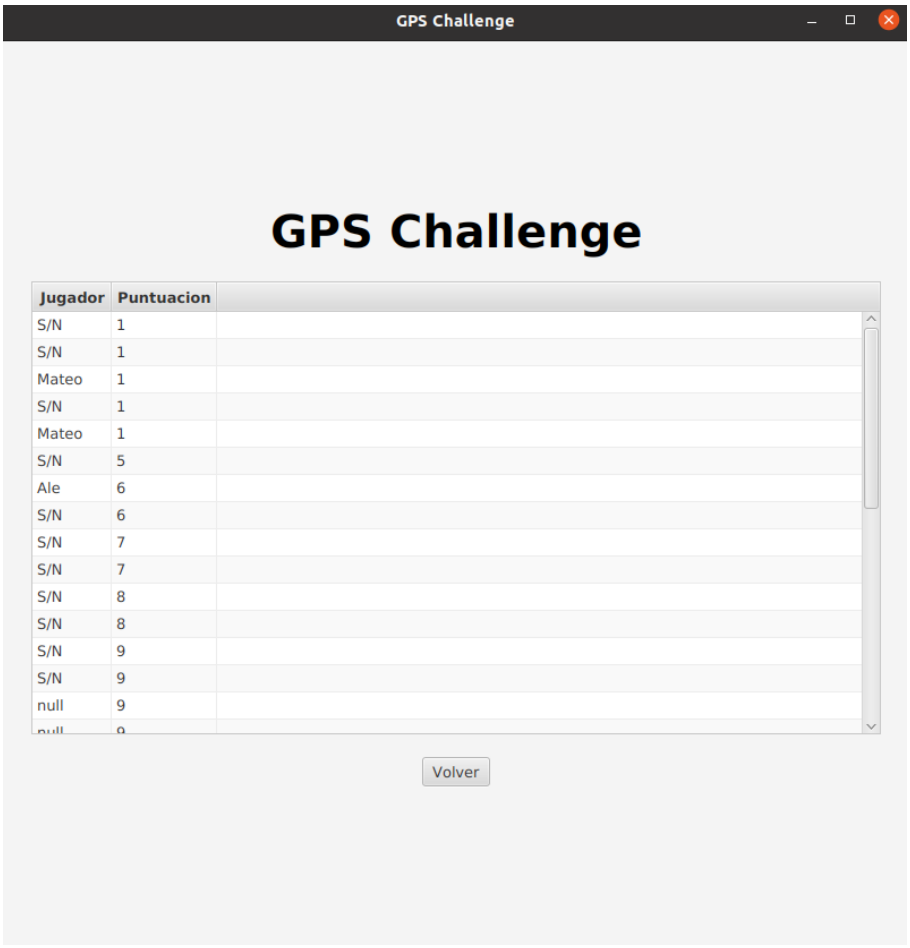

Figura 2: Visualización del ranking.

4. Diagramas de clase

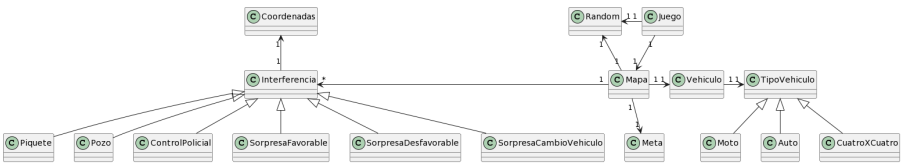

Figura 3: Diagrama general.

Figura 4: Diagrama en detalle de Juego y Mapa.

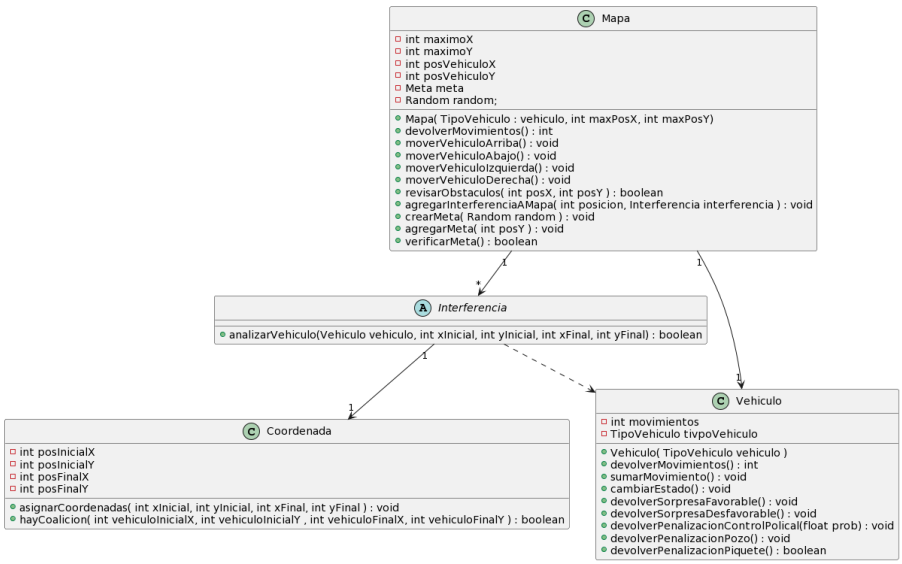

Figura 5: Relación del Mapa con Vehículo y Coordenada.

Figura 6: Diagrama detallado de Vehiculo.

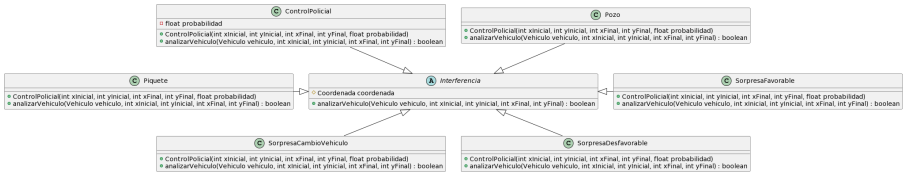

Figura 7: Diagrama detallado de Interferencia.

5. Detalles de implementación

5.1. Coordenada

La entidad Coordenada fue creada con el propósito de solucionar la abstracción del movimiento y posicion de otras entidades dentro del mapa, tomando como referencia las esquinas (intersección entre dos calles) del mismo. La lógica utilizada fue que una clase Coordenada contiene dos esquinas identificadas por sus posiciones en x e y. Una es tomada como inicial y otra como final, indicando la posición actual y a la que se desea desplazar respectivamente *Cabe añadir que a lo largo de este modelo se tomó como referencia cartesiana a la esquina superior izquierda como el punto (1,1). Hacia la derecha se incrementa el eje de las abscisas y hacia abajo el de las ordenadas.*

```
private int posXInicial;
private int posYInicial;
private int posXFinal;
private int posYFinal;
```

5.2. Random

El objetivo de ésta entidad es resolver problemas relacionados a la aleatoriedad en el contexto del modelo. Hace uso de la clase Random proporcionada por la librería

```
java.util
```

Random en el modelo entiende mensajes como

```
generarXInicial generarYInicinal generarXFinal generarYFinal
```

los cuales pueden resultar útiles a la hora se generar, por ejemplo, un Mapa con tamaño aleatorio.

6. Diagramas de secuencia

A continuación se muestran dos diagramas de secuencia para ejemplificar casos de uso donde

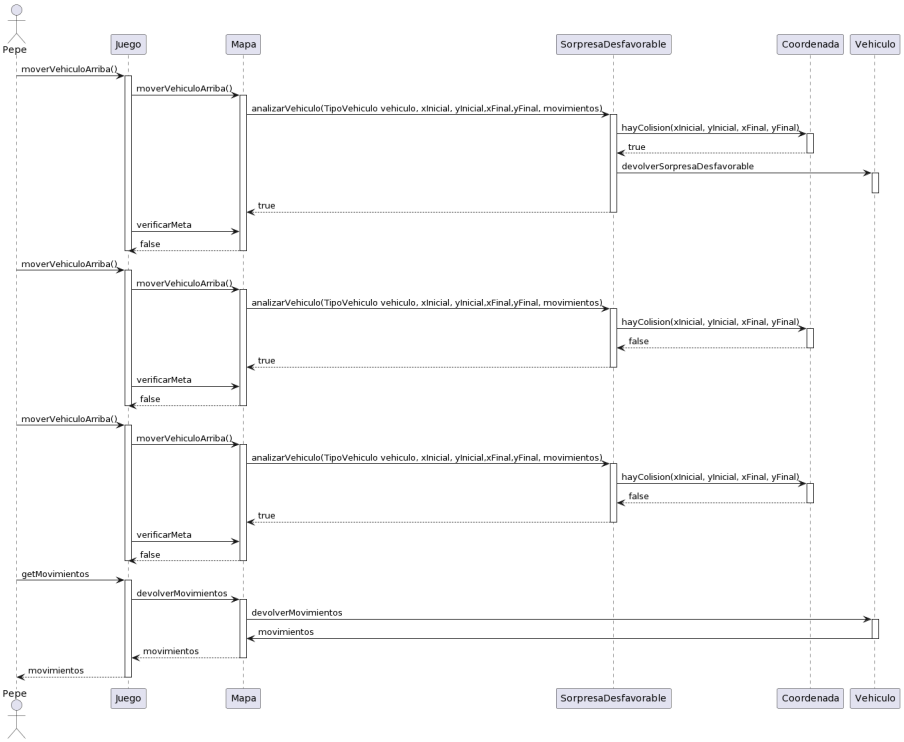

Figura 8: Diagrama de comportamiento al encontrarse con una sorpresa desfavorable.

Como se observa en la figura 4, cada vez que el jugador desea desplazar su vehículo, lo hace enviandole mensajes al juego. Éste delega el comportamiento en el mapa el cual le envía el mensaje analizarVehiculo a la SorpresaDesfavorable. Finalmente la secuencia de mensajes es resuelta dentro de Vehiculo.

SorpresaDesfavorable responde true para indicar que al mapa que al mover el vehiculo, se encontró con ella.

Notar que cada vez que se mueve el vehiculo se verifica si éste alcanzó la meta dado que si el mensaje devolviese true, se terminaría la partida.

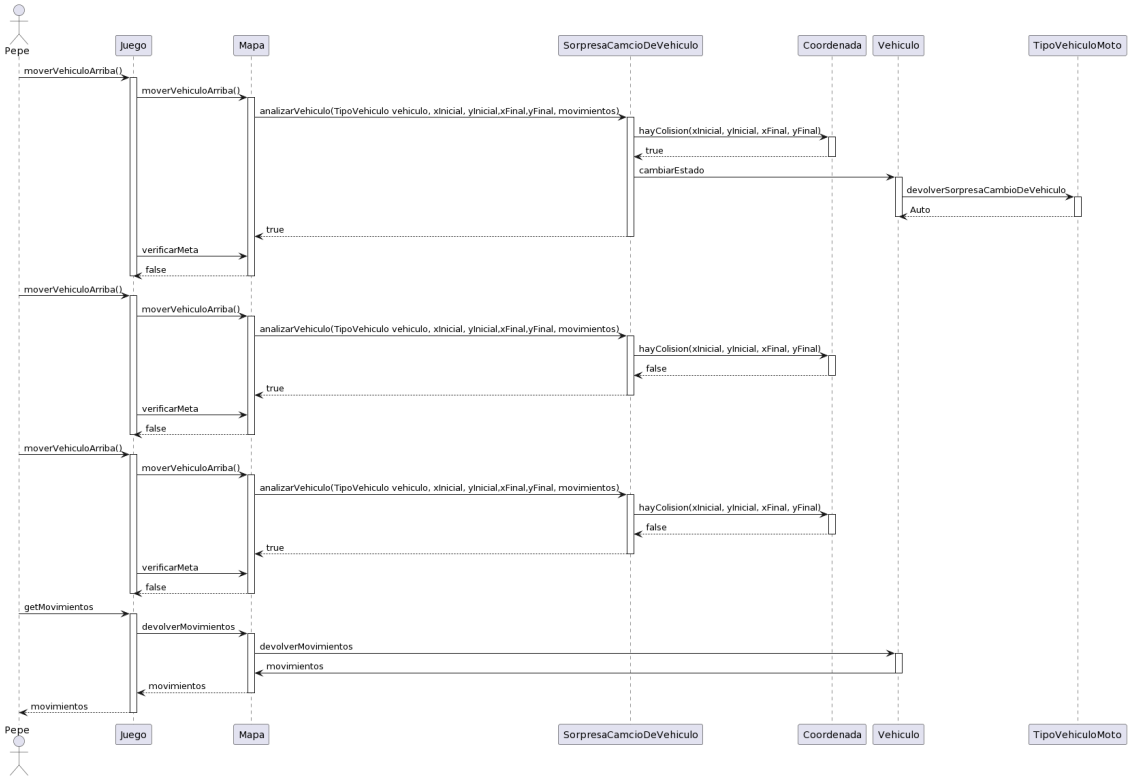

Figura 9: Diagrama de comportamiento al cambiar de vehículo.

El comportamiento es algo similar al del caso anterior. En este caso el tipo de Interferencia es SorpresaCambioDeVehiculo (ver diagrama de clases) y será la encargada de enviar el mensaje cambiarEstado a Vehiculo quien resolverá a través de TipoVehiculoMoto su cambio de estado.

Para este caso de uso como ejemplo una moto que cambia a auto. A partir de ese momento, todos los movimientos y verificaciones del vehículo son hechas desde las perspectiva de Auto y no de Moto.

7. Diagramas paquetes

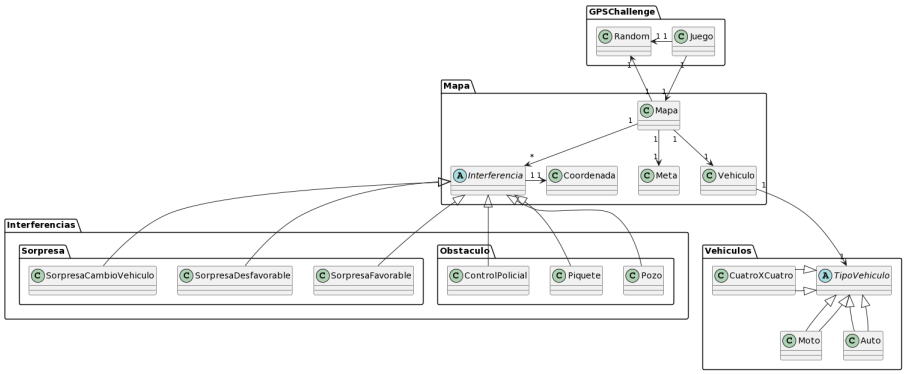

Figura 10: Diagrama de paquetes del proyecto

8. Mejoras a futuro

Dado que el modelado es un proceso iterativo y necesitamos plantearlo y refarctorizarlo varias veces para llegar a la mejor solución, el modelo actual presenta varias cuestiones que pueden implementarse de una manera mas óptima y entendible.

Algunos conceptos que pueden ser tenidos en cuenta para una próxima refactorización son:

Moviemiento del vehículo Como se observa en la clase Mapa, existen cuatro métodos para resolver el movimiento de un vehículo, uno para cada dirección, pero éstos tienen un comportamiento muy parecido siendo lo unico en variar su valor asociado a la posición actual. Podría resolverse éste inconveniente haciendo uso de polimorfismo y una nueva clase Dirección

Responsabilidades del mapa Lo primero que nos puede llamar la atención al ver el modelo es la cantidad de responsabilidades que tiene el mapa. Tales como el movimiento del vehiculo, la creación de interferencias, la creación y asignación de la meta, modificación del ranking, entre otras. Resultaría interesante e incluso necesario la delegación de estas responsabilidades debido a que cada entidad debería tener un y solo un problema concreto.

Random Como se mencionó anteriormente, el propósito de esta clase fue resolver problemas de aleatoriedad en el juego, pero al hacer uso directo de su homónimo proporcionado por el paquete java.util, podríamos plantearnos la manera de resolverlos sin la necesidad de crear nuestra propia clase Random.