FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2
GPS Challenge
Grupo 12

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

Alumno	Padron	Email
Bloise, Julieta	98592	jbloise@fi.uba.ar
Fernandez, Ramiro	105595	rfernandezm@fi.uba.ar
Fuentes, Azul	102184	afuentes@fi.uba.ar
Scazzola, Martin	106403	mscazzola@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 un juego, GPS Challenge, en el lenguje de programacion Java, utilizando los conceptos del paradigma de la orientación a objetos vistos en el curso.

2. Supuestos

- Como máximo habrá dos objetos por calle, sin contar el objeto sin penalización.
- Los movimientos del vehículo serán de esquina a esquina, no puede "posicionarse" en una calle.
- Los objetosCalle solo se pueden encontrar en las Calles y además de influir en el estado del vehículo, no lo dejarán transitar entre calles. En el caso de no dejar que el Vehiculo avance a la siguiente esquina se quedará en la esquina inicial.
- Si hay un Piquete en la Calle no importa si esta "posicionarse" o no, el auto no podra avanzar.
- Los obstáculos no desapareceren luego de ser encontrados por el vehículo.
- Los movimientos se cuentan como números enteros. En el caso de que un objetoCalle (sorpresa u obstáculo) influya en los mismos, se verá contemplado como un redondeo.
- La Meta se encuentra en la última columna y en una fila aleatoria.

3. Modelo de dominio

El trabajo consiste en un juego (GPSChallenge) en que un Vehiculo se encuentra con diferentes obstaculos y sorpresas en su camino a la meta. Para representar el juego se utiliza una clase Mapa, una clase Vehiculo y clases Calle y ObjetoCalle que consisten en lo que luego verá representado en pantalla, el usuario del programa.

4. Diagramas de clase

En los diagramas de clase se pueden ver las relaciones entre los diferentes actores del juego, de acuerdo con el paradigma de objetos. Las interacciones entre los mismos se ven explicadas dentro de la sección **Detalles de implementación**.

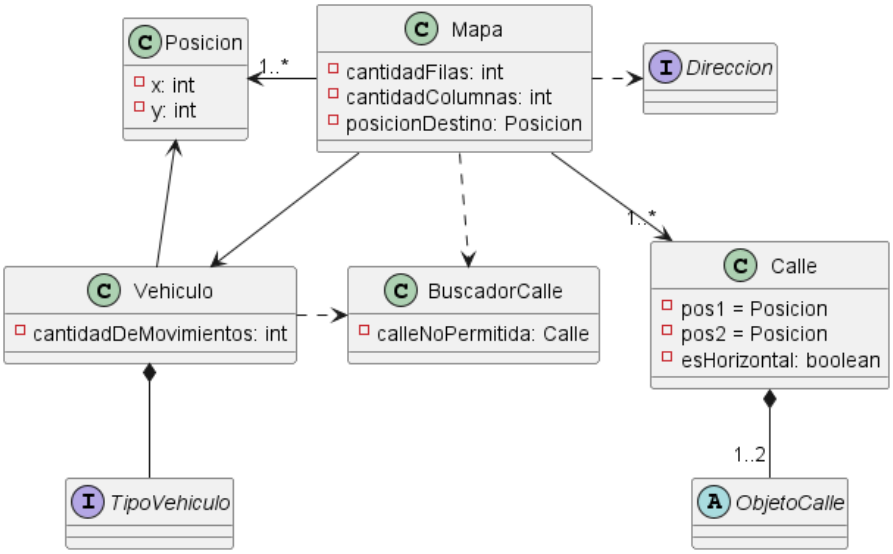

Figura 1: Diagrama general del Juego

Se ocultaron los métodos en la Figura 1 para dar claridad al diagrama al momento de leerlo

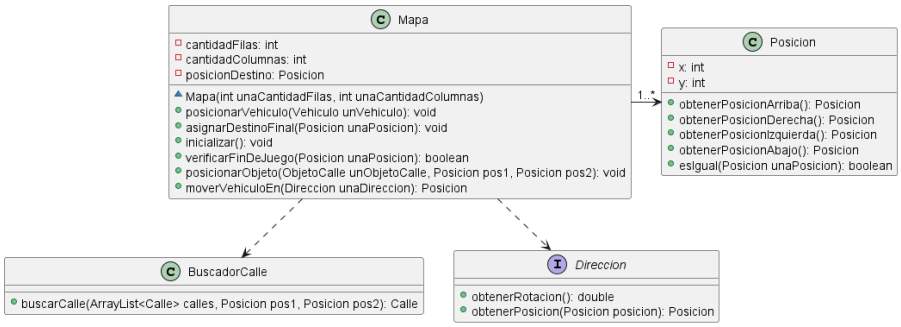

Figura 2: Diagrama del mapa del juego

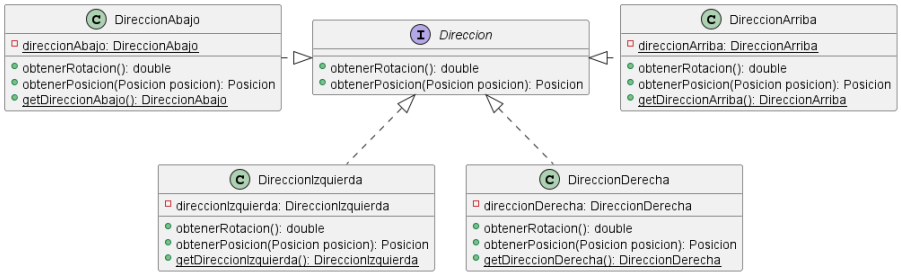

Figura 3: Diagrama de la dirección

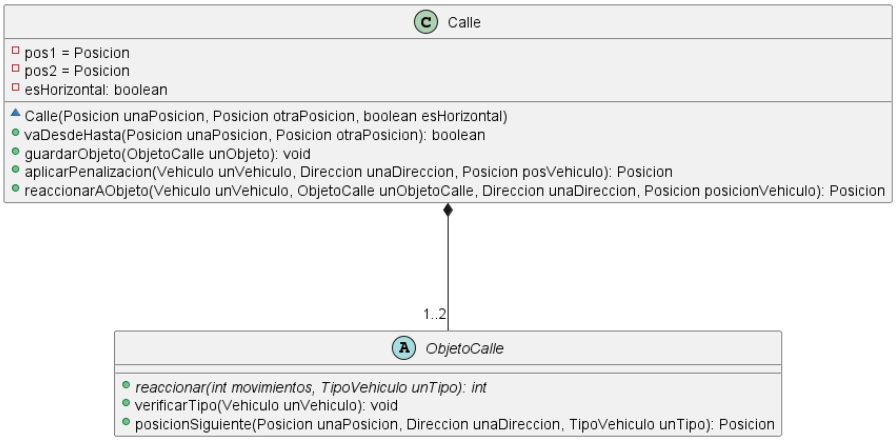

Figura 4: Diagrama de la calle

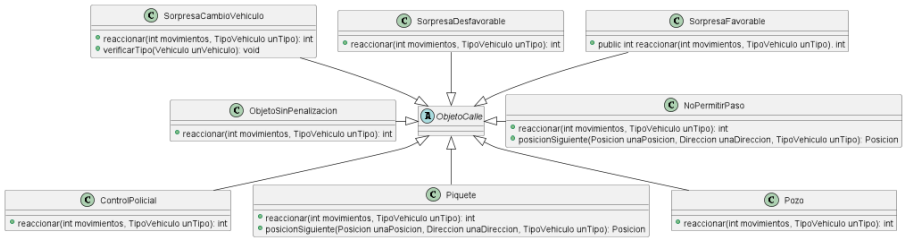

Figura 5: Diagrama de los objetos de la calle

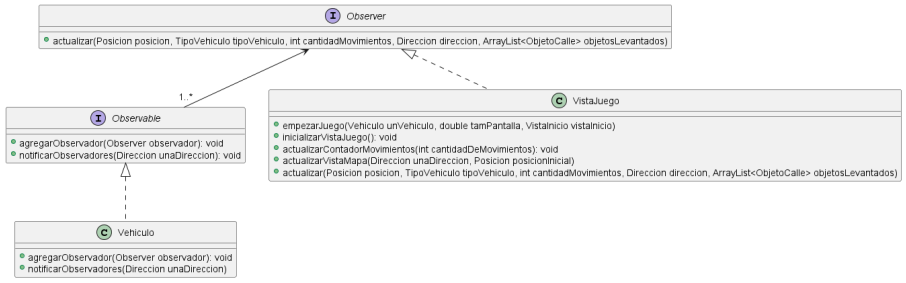

Figura 6: Diagrama de la implementacion del patrón observer
Se ocultaron algunos métodos en la Figura 6 de la clase Vehiculo y los atributos de la clase VistaJuego para simplificar el diagrama

5. Detalles de implementación

5.1. Patrones de Diseño

Debido a la implementación elegida por el grupo, el estado del Vehiculo se veía fuertemente afectado por su contexto, dado que se utilizaban diferentes tipos de vehículo. Para encarar de esta manera la realización del modelo fue necesaria la implementacion del **Patrón State**, ya que Vehiculo se comportaba diferente dependiendo de si su Tipo era Auto, Moto o CuatroPorCuatro, por lo que se tenia un enorme condicional que verificaba de que Tipo se trataba para poder reaccionar correctamente a los ObjetosCalle con los que se encontraba.

Gracias al **Patrón State** resolvimos este problema creando la interface TipoVehiculo y que luego fue aplicada por Moto, Auto y CuatroPorCuatro, redujimos el código duplicado y eliminamos los condicionales.

Podríamos haber utilizado el Patrón **Strategy**, pero en este caso nos pareció conveniente el State ya que permitía un uso explícito de las variantes del estado del jugador, dependiendo de su contexto, sin necesidad de settear su tipo desde afuera.

Además, debido a la implementación de las reacciones dentro de las calles, decidimos hacer un buen uso del **NUll Pattern** de tal manera que, en vez de gastar tiempo en detalles de implementación probablemente fuera del paradigma, pudiéramos hacer uso del patrón visto en clase que nos permite reaccionar a objetos sin hacer nada en concreto y, por tanto, un correcto reemplazo de un espacio vacío dentro de la calle. De esta manera, para la implementación que reccorre los objetos dentro de una calle no distingue entre un objeto real y un objeto que no genera ningún cambio en el estado del jugador o el juego.

También usamos el patrón **Singleton** para implementar las direcciones, ya que nos permitió asegurar una única instancia de la clase, además de acceso global a la misma.

5.2. Polimorfismo

El polimorfismo es utilizado en el trabajo práctico para poder mover los diferentes vehículos a través del mapa mediante los mismos mensajes, independientemente del vehículo del que se esté hablando.

5.3. Double dispatch

utilizamos double dispatch en las situaciones donde el vehículo debe reaccionar a los objetos de la calle, ya que no depende únicamente del receptor del mensaje, sino también de los argumentos del mismo para resolver el problema. Esto nos permite generar ambos delegación y polimorfismo en el mismo mensaje.

5.4. Herencia

El uso de la herencia en el trabajo práctico fue esencial para evitar la repetición de código en las implementaciones de los tipos de objetos de la calle, tanto por los métodos concretos que implementa la clase madre para ser utilizados por cada uno de ellos como por el método abstracto que permite la reacción implementada para cada objeto en concreto.

5.5. Interfaces

Las interfaces en el trabajo práctico permitieron asegurar la responsabilidad de comportamientos implementados por las clases Dirección y Tipo de vehículo para que las instancias de estas clases cumplieran con esa implementación.

6. Diagramas de secuencia

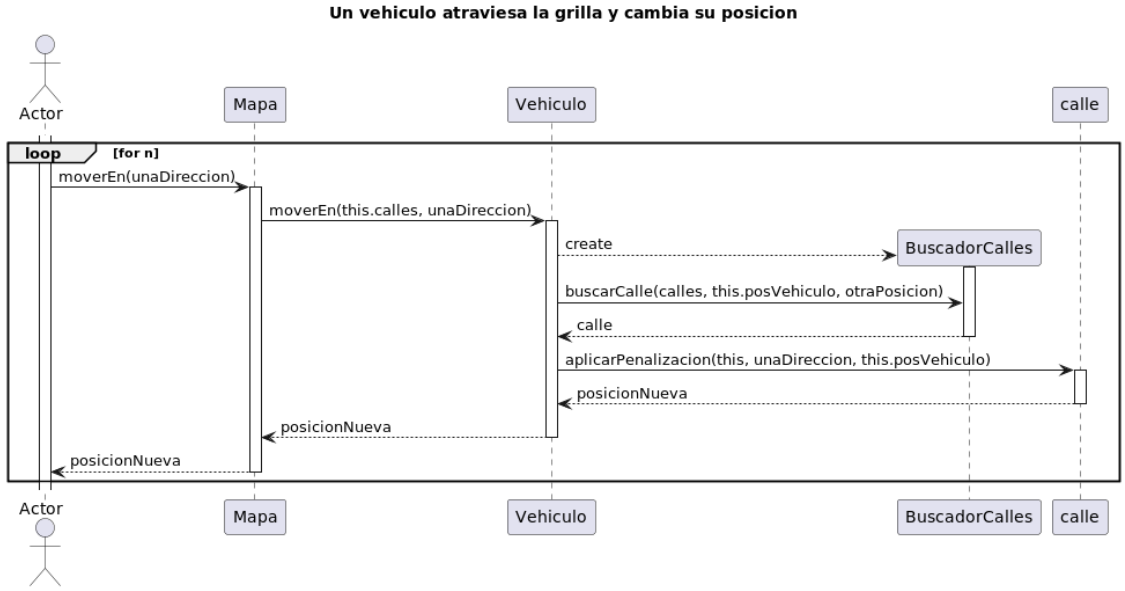

Figura 7: Un Vehiculo atraviesa la grilla y cambia su posicion

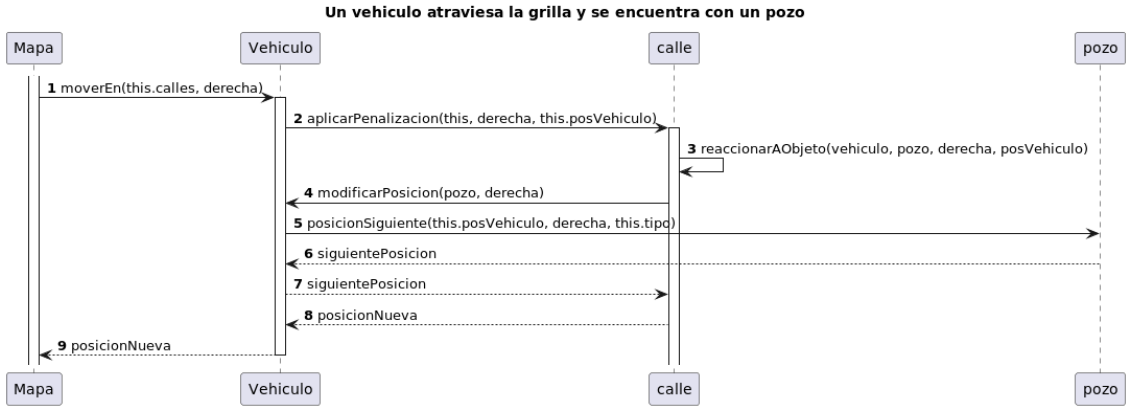

Figura 8: Un vehiculo atraviesa la grilla y se encuentra con un pozo

En este diagrama en particular, podemos observar como el ObjetoCalle es el que "mueve.ª"l Vehiculo, ya que el Vehiculo le pide a Pozo la posicionSiguiente para que calle se encargue de, a través de aplciarPenalizacion(), Vehiculo pueda modificar sus atributos privados.

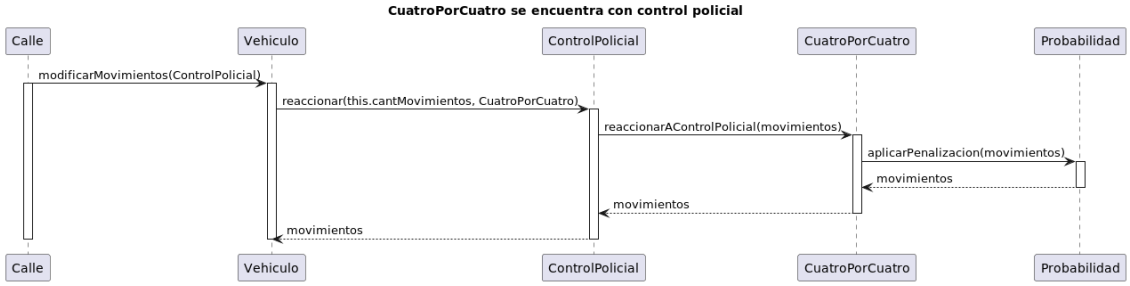

Figura 9: CuatroPorCuatro se encuentra con control policial

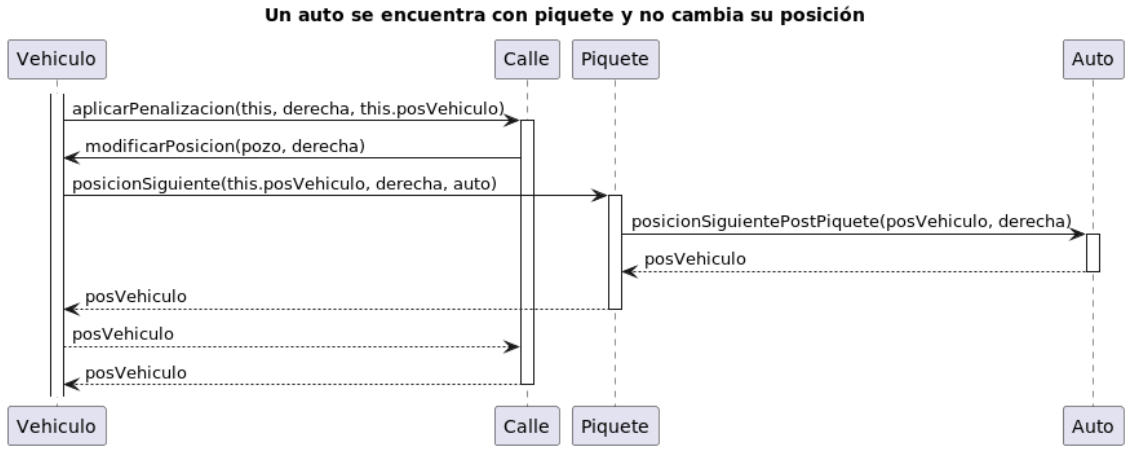

Figura 10: Un auto se encuentra con piquete y no cambia su posicion

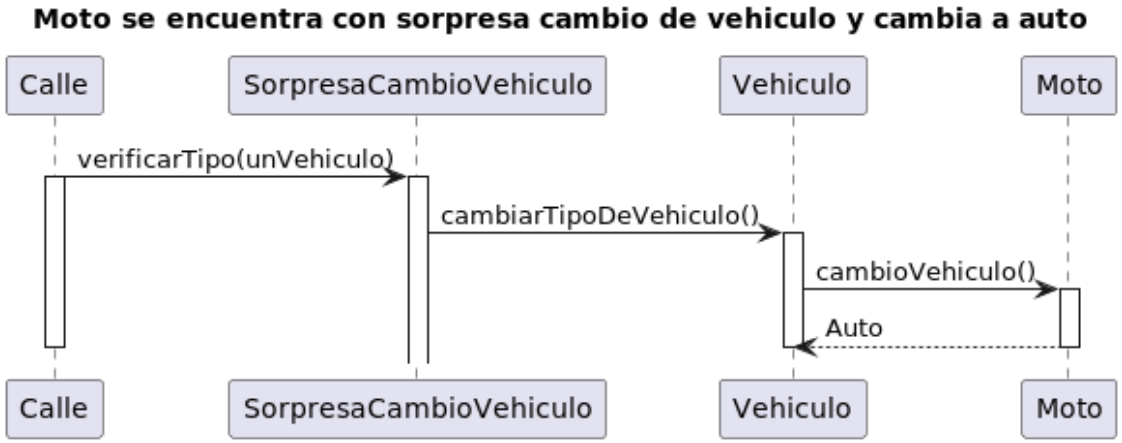

Figura 11: Moto se encuentra con sorpresa cambio de vehiculo y cambia a auto