**FIUBA - 7507**

**Algoritmos y programación 3**

*Trabajo práctico 2: GPS Challenge*

2do cuatrimestre, 2013

(trabajo grupal)

|  |  |  |
| --- | --- | --- |
| Mail | Padrón | Nombre |
| turco807@gmail.com | 92774 | Alic, Mirko |
| educ\_907@hotmail.com | 92662 | Calcabrini, Eduardo |
| ezequielrozen@gmail.com | 92754 | Rozenzwaig, Ezequiel |

Fecha de entrega final: 13/12/13

Tutor: Massuh, Pablo

Nota Final:

**Tabla de contenidos**

[Introducción](#id.gjdgxs)

[-Objetivo del trabajo](#id.30j0zll)

[Consigna general](#id.1fob9te)

[Descripción de la aplicación a desarrollar](#id.3znysh7)

[-Contexto](#id.2et92p0)

[Objetivo del juego](#id.tyjcwt)

[Puntos](#id.3dy6vkm)

[Dinamica del juego](#id.1t3h5sf)

[Disparos](#id.4d34og8)

[Tablero](#id.2s8eyo1)

[Naves](#id.3rdcrjn)

[Entregables](#id.26in1rg)

[Forma de entrega](#id.lnxbz9)

[Informe](#id.35nkun2)

-[Supuestos](#id.1ksv4uv)

[Modelo de dominio](#id.44sinio)

[Diagramas de clases](#id.2jxsxqh)

[Detalles de implementación](#id.z337ya)

[Excepciones](#id.3j2qqm3)

[Diagramas de secuencia](#id.1y810tw)

[Checklist de corrección](#id.2xcytpi)

[Código](#id.1ci93xb)

Introducción

* 1. Objetivo del trabajo

Aplicar los conceptos enseñados en la materia a la resolución de un problema, trabajando en forma grupal y utilizando un lenguaje de tipado estático (Java)

* 1. Consigna general

Desarrollar la aplicación completa, incluyendo el modelo de clases e interface gráfica. La aplicación deberá ser acompañada por prueba unitarias e integrales y documentación de diseño. En la siguiente sección se describe la aplicación a desarrollar.

Descripción de la aplicación a desarrollar

* 1. Contexto

La empresa Algo Ritmos SA dedicada al desarrollo de video juegos a decidido contratar a un grupo de programadores para implementar el juego GPS Challenge

* 1. Objetivo del juego

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.

* 1. Dinámica del juego

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

* 1. Vehículos

El jugador podrá optar por tres diferentes tipos de vehículos.

* moto
* auto
* 4x4
  1. Obstáculos

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

* Pozos: Le suma 3 movimientos de penalización a autos y motos, pero no afecta a las 4x4.
* Piquete: Autos y 4x4 deben pegar la vuelta, no pueden pasar. Las motos puede 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.
  1. Sorpresas

También se podrán encontrar diferentes tipos de sorpresas:

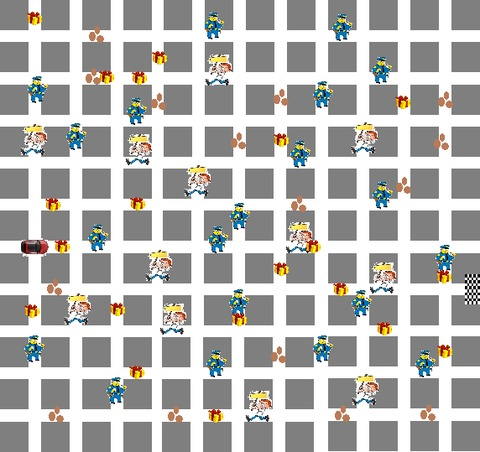
* Sorpresa Favorable: Resta el 20% de los movimientos hechos.
* Sorpresa Desfavorable: Suma el 25% de los movimientos hechos.
* Sorpresa Cambio de Vehículo: Cambia el vehículo del jugador. Si es una moto, la convierte en auto. Si es un auto lo convierte en 4x4. Si es una 4x4 la convierte en moto.

Las sorpresas figuraran en el mapa como un regalo y no se sabrá que es hasta que el vehículo la accione.

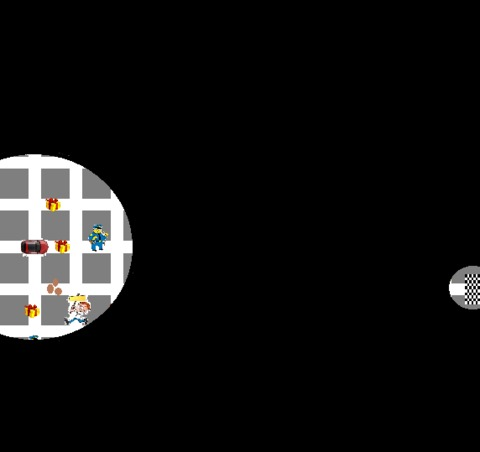
* 1. Escenario

Para hacerlo más interesante y jugable, 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.



Ejemplo de escenario



Ejemplo de escenario como lo visualiza el jugador

* 1. Puntajes altos

Se debe almacenar un ranking donde figuren los mejores puntajes asociados a un nickname que indique el usuario.

Entregables

* Código fuente de la aplicación completa, incluyendo también: código de la pruebas, archivos de recursos
* Script para compilación y ejecución (ant)
* Informe, acorde a lo especificado en este documento
  1. Forma de entrega

A coordinar con el docente asignado.

* 1. Fechas de entrega

Se deberá validar semanalmente con el docente asignado el avance del trabajo. El docente podrá solicitar ítems específicos a entregar en cada revisión semanal.

La entrega final deberá ser en la semana del 13 de diciembre, en la fecha del curso en que se está inscripto.

Informe

* 1. Supuestos

• Decidimos que el juego se base en tiempo y no en movimientos por lo tanto hubo modificaciones en los diferentes objetos que interactúan con el vehículo y en la visión general de como se desarrolla el mismo:

Los obstáculos que incrementaban cantidades de movimientos (Pozo, ControlPolicial, Piquete) se modificaron con el fin de que el auto quede inmóvil durante esa cantidad de movimientos penalizados. De esta forma se logra que transcurra tiempo.

Con las sorpresas que suman o restan movimientos en un porcentaje, se tomo como equivalente tener en cuenta el porcentaje de tiempo transcurrido al momento en que interactúa el vehículo con las mismas y sumarlo o restarlo al tiempo total.

Respecto a la neblina propuesta por el enunciado como un plus de dificultad, se decidió no implementarla ya que suponemos que la complejidad de que el movimiento del vehículo fuese dinámico y no por turnos, ya le agrega una dificultad interesante.

Debido a la eliminación del turnos, el objetivo del juego se cambio por el de intentar llegar a la meta en la menor cantidad de tiempo posible, no habiendo un límite para terminar el Nivel propuesto.

Modelo de dominio

• Mapa: Es la encargada de contener las posiciones habilitadas y almacenar diferentes elementos para el momento de jugar, como puede ser un vehículo o obstáculos/sorpresas teniendo en cuenta las posiciones habilitadas.

• EditorMapa: Tiene la responsabilidad de dejar al mapa en condiciones óptimas para jugar. Para tener una referencia visual es la encargada de establecer que posiciones son calles para que los obstáculos y vehículos puedan posicionarse.

• Vehículo: Esta clase se relaciona de forma directa con auto, moto y 4x4 a través de un estado y se verá afectada de forma diferente cuando actúe con los distintos obstáculos.

• Piloto: Es el que se encarga de conducir al Vehículo y a su vez un intermediario entre los obstáculos/sorpresas y el vehículo.

• Obstáculo y Sorpresa: Hay diferentes clases que depende de estas dos clases. Cada una interactuara con el vehículo de diferentes formas teniendo en cuenta penalizaciones, beneficios o cambios de vehículos.

* 1. Diagramas de clases

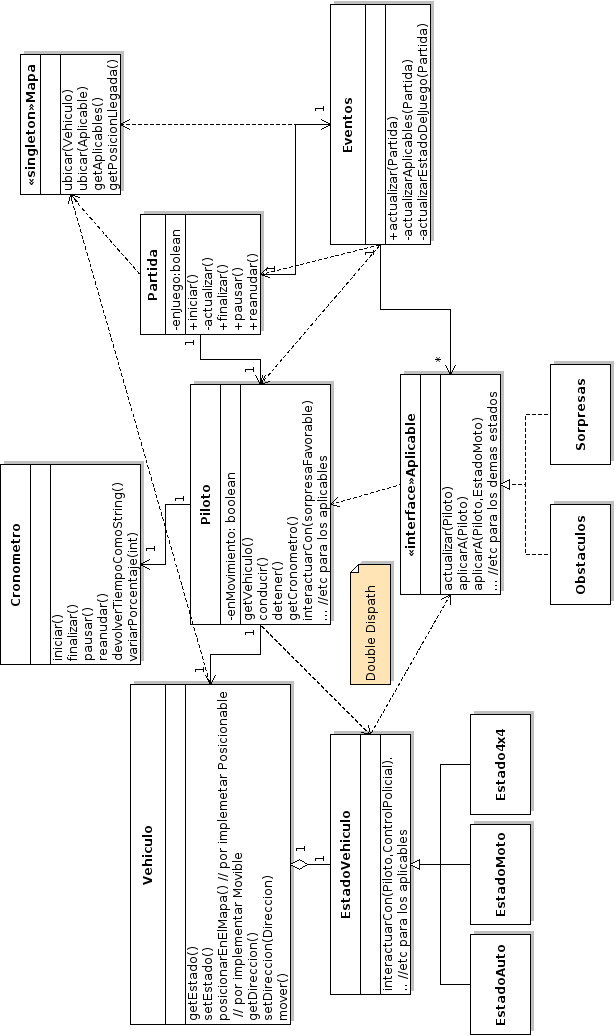
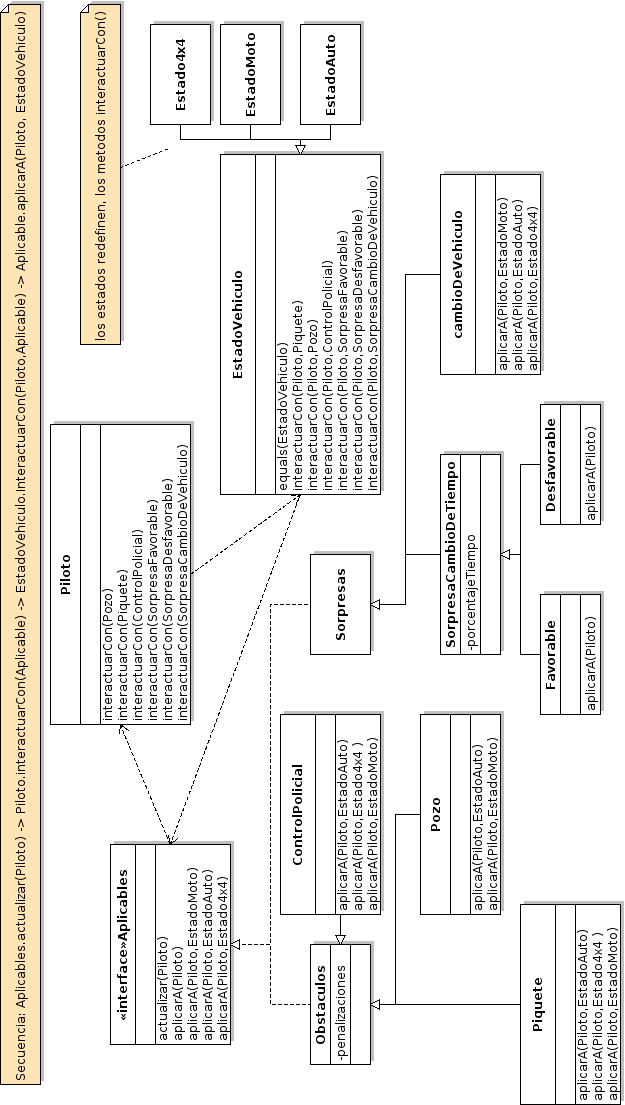
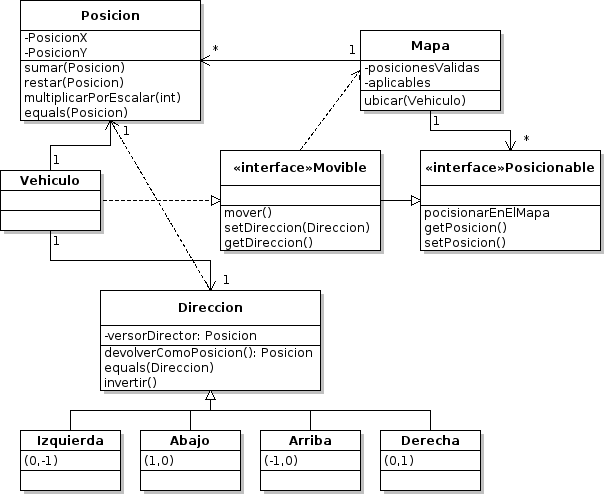
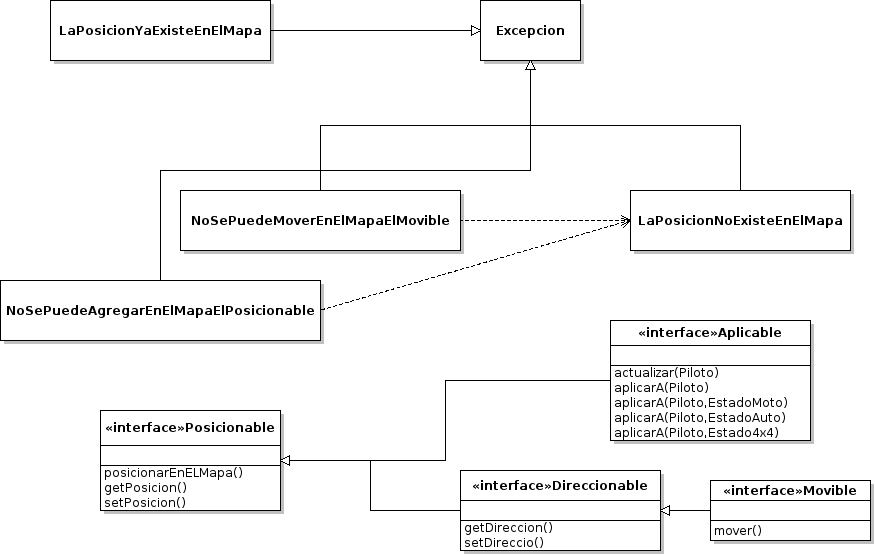
Diagrama de clase del juego:

Diagrama de clase de aplicables:



Diagrama de clase de un vehículo:

Diagrama de clase de interfaces y excepciones

* 1. Detalles de implementación

La clase Mapa, se ha implemento como un contenedor de los objetos relevantes para el juego (Vehículo, Sorpresas, Obstáculos, posición de inicio y posición de llegada) y de posiciones validas. Las mismas son la abstracción de las calles y pretenden ser el “piso” donde se ubican los demás objetos que contiene el Mapa. El mismo se propuso como una clase Singleton, ya que consideramos que atraviesa a todas las demás clases.

La clase Eventos es una clase que nos resulta interesante explicar, se encarga de actualizar los eventos de la partida en juego.

Por una parte tiene el comportamiento necesario para acoplar la interacción entre los aplicables que posea el mapa y el piloto que se encuentre conduciendo, es decir, les enviara los mensajes correspondientes a dichas clases para que puedan comunicarse, logrando así, un mayor nivel de encapsulamiento y una mejor agregación.

Por otra parte, estará al tanto del estado en el que se encuentra la partida, y en caso de que verifique su finalización se lo comunicara a la misma.

De esta forma la clase Eventos logra encapsular todo el comportamiento necesario para actualizar y actuar respecto a los eventos que van sucediendo.

La clase Estado, es la abstracción de los diferentes tipos de vehículos con los que se puede jugar Auto, Moto y 4x4, sus equivalentes son EstadoAuto, EstadoMoto y Estado4x4. Esta decisión de diseño entrega como ventaja, simplificar la clase Vehículo, quitándole la responsabilidad de saber interactuar con el Piloto y los Aplicables y delegando este comportamiento al estado en el que se presente. Además, dicha interacción solo depende del estado del mismo y no de otras características o potencialidades del Vehículo.

Como desventaja, produce cierto nivel de acoplamiento entre un Vehículo y su Estado, lo cual puede ser molesto a la hora de inicializar al vehículo o tener que pasar su estado en los casos que se requiera.

* 1. Excepciones

Las diferentes excepciones creadas son:

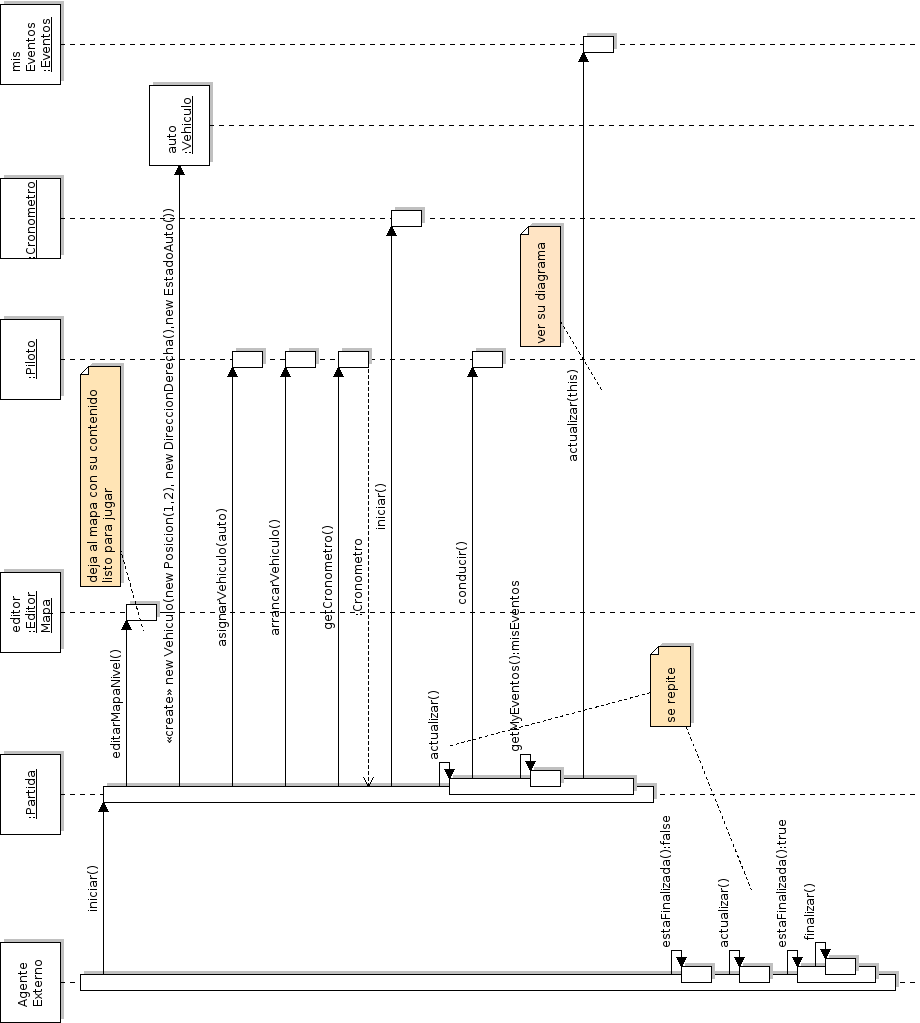
• LaPosicionYaExisteEnElMapa: Esta excepción es lanzada por la clase mapa, cuando se intenta agregar mas de una vez una posición a su lista de posiciones validas. La misma es capturada en el contexto del EditorMapa cuando este agrega las correspondientes posiciones al Mapa. En caso de producirse el error se opta por no hacer nada, y solo avisarle al programador por consola el error producido.

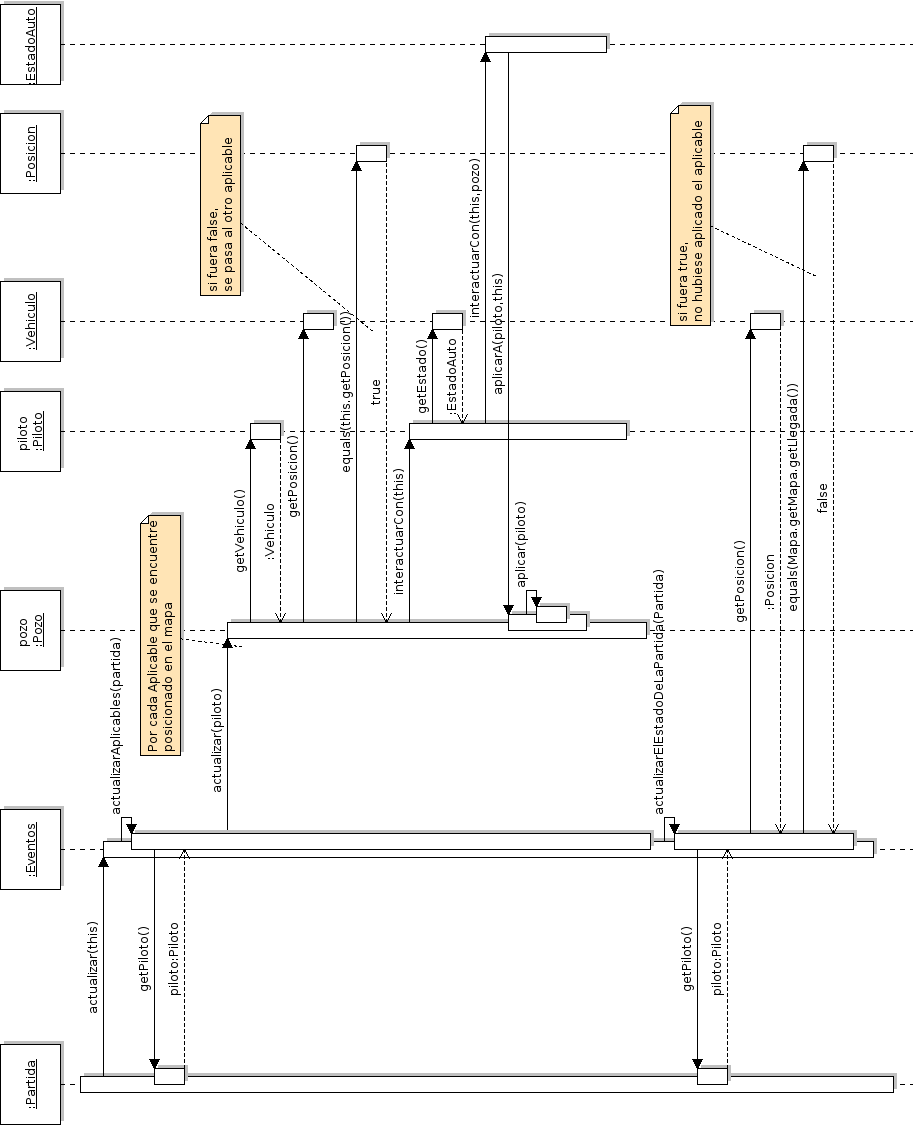
• LaPosicionNoExisteEnElMapa: Esta excepción es lanzada siempre que se intente ubicar algún objeto Posicionable (Vehículo, Sorpresas, Obstáculos) o la posición de Inicio y la posición de llegada, en una posición que no exista, es decir que no posea el Mapa como posición valida. La misma es capturada en el contexto del EditorMapa en los métodos para agregar un inicio y una llegada. Otro contexto donde se lanza es en el método ubicar de cada Posicionable. En el caso del EditoMapa, de producirse se opta por no hacer nada, y simplemente mandarle un mensaje al programador por consola detallando el error producido y para los Posicionables, al capturarse se lanza una nueva excepción (NoSePuedeAgregarEnElMapaElPosicionable) subiendo de nivel el error producido.

• NoSePuedeAgregarEnElMapaElPosicionable: Retomando lo anterior, esta excepción es lanzada como respuesta de haber intentado ubicar un Posicionable en una posición invalida para el Mapa. La misma es capturada en el contexto del EditorMapa cuando se intenta agregar las Sorpresas y Obstáculos al Mapa en el método editarMapaNivel, nuevamente al capturarse se opta simplemente por informar el programador del error. Otro posible contexto es en la clase Partida (paquete control) cuando se agrega al Mapa el vehículo, previamente elegido por el usuario. De producirse, nuevamente la decisión es la misma.

• NoSePuedeMoverEnElMapaElMovible: Esta excepción es lanzada en el método mover del vehículo y se produce al querer moverlo a una posición que no existe en el mapa (la abstracción es pensar que se quiere mover a una cuadra) por lo tanto no se le permite llevar a cabo la tarea. La misma es capturada en el contexto del Piloto en su método conducir y en caso de producirse se opta por no hacer nada, ya que en la dinámica del juego, es normal que el usuario ingrese una dirección en la cual el vehículo no pueda dirigirse, entonces simplemente con no hacer nada el auto quedara en su última posición valida esperando que se modifique su dirección hasta que que la misma permita seguir un recorrido en el Mapa.

* 1. Diagramas de secuencia

Diagrama de secuencia de un juego:

Diagrama de secuencia de la actualización de la clase Eventos:

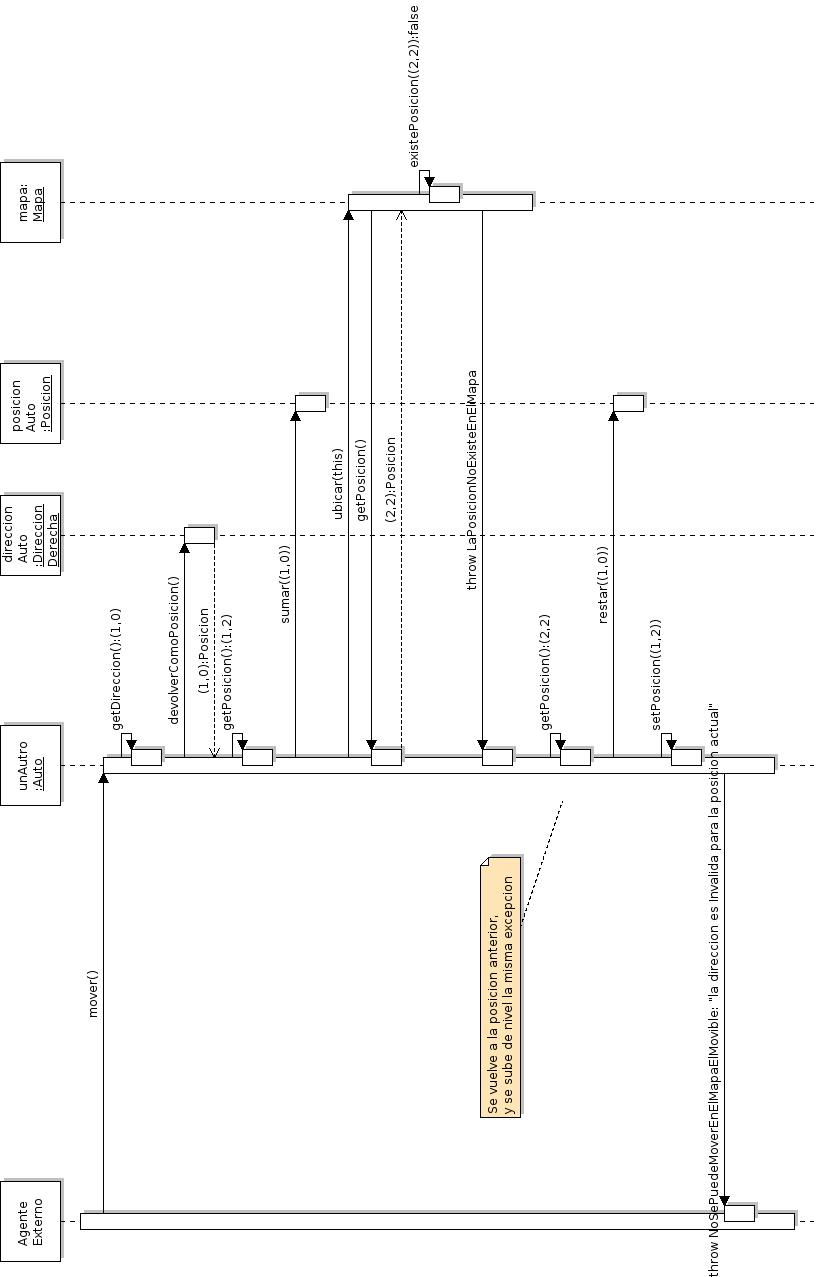
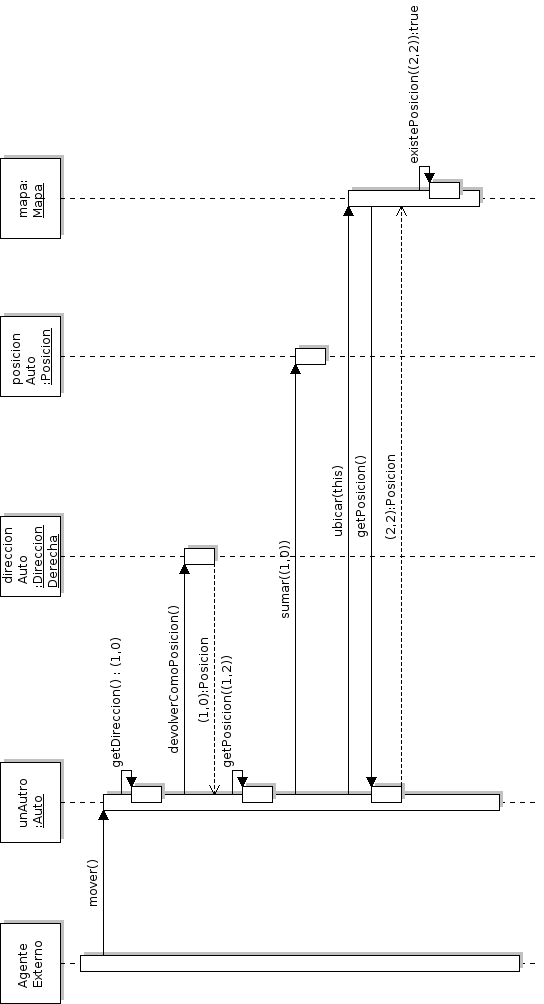
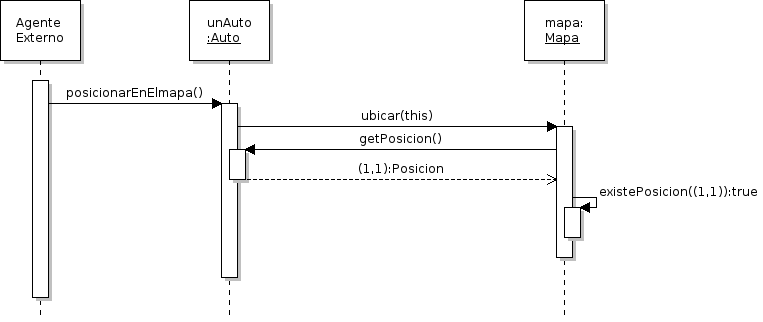
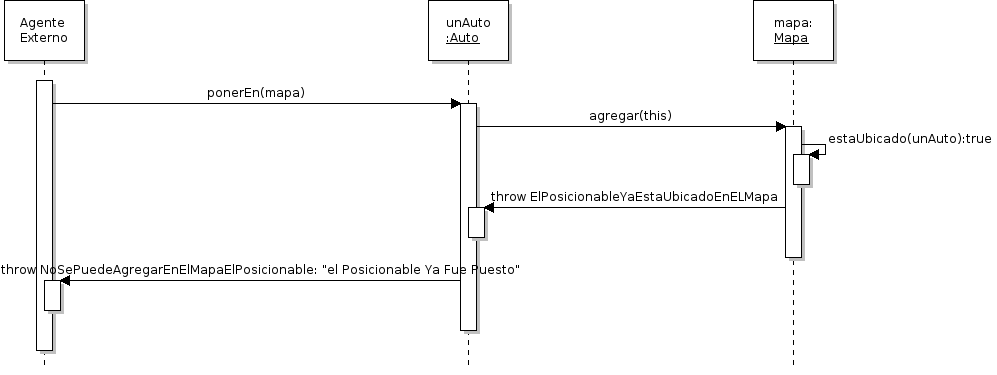
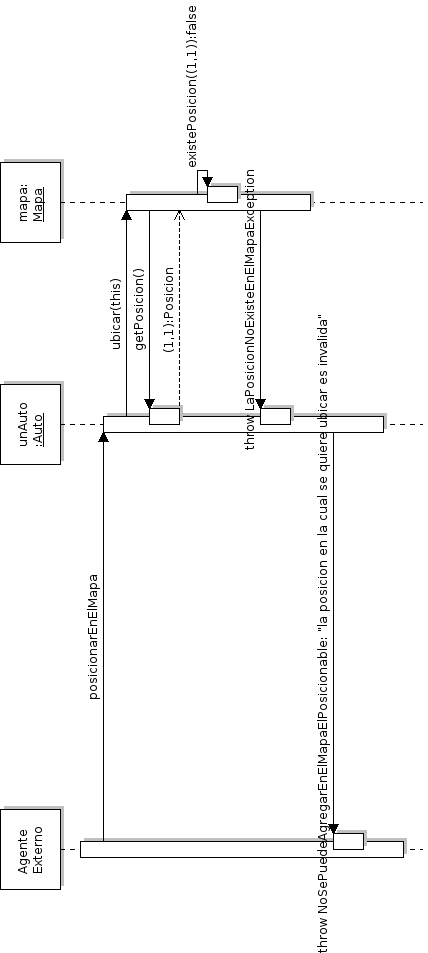
Diagrama de secuencia de un movible en una posición que no existe:

Diagrama de secuencia de un movible en un mapa:

Diagrama de secuencia de Posicionable colocado en un Mapa:   
  
  
  
Diagrama de secuencia ubicar un posicionable que ya existe en un mapa:



Diagrama de secuencia ubicar un posicionable en posición inexistente:

Checklist de corrección

Esta sección es para uso exclusivo de la cátedra, por favor no modificar.

**Carpeta**

**Generalidades**

* ¿Son correctos los supuestos y extensiones?
* ¿Es prolija la presentación? (hojas del mismo tamaño, numeradas y con tipografía uniforme)

**Modelo**

* ¿Está completo?¿Contempla la totalidad del problema?
* ¿Respeta encapsulamiento?
* ¿Hace un buen uso de excepciones?
* ¿Utiliza polimorfismo en las situaciones esperadas?

**Diagramas**

**Diagrama de clases**

* ¿Está completo?
* ¿Está bien utilizada la notación?

**Diagramas de secuencia**

* ¿Está completo?
* ¿Es consistente con el diagrama de clases?
* ¿Está bien utilizada la notación?

**Diagrama de estados**

* ¿Está completo?
* ¿Está bien utilizada la notación?
  1. Código

**Generalidades**

* ¿Respeta estándares de codificación?
* ¿Está correctamente documentado?