FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — AlgoStar

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

Alumno:	BARRIONUEVO, Lautaro
Número de padrón:	104007
Email:	lebarrionuevo@fi.uba.ar
Alumno:	FRANCIULLI, Lucas
Número de padrón:	107059
Email:	lfranciulli@fi.uba.ar
Alumno:	SUBERO, Andrés
Número de padrón:	109114
Email:	asubero@fi.uba.ar
Alumno:	GARCÍA, Nicolás
Número de padrón:	103448
Email:	nagarcia@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un juego similar a Starcraft utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Uno de los principales supuestos que tomamos para el trabajo tiene que ver con el final del juego. Ya que en la consigna se prestaba un poco a confusión, determinamos que el final del juego suceda no cuando se destruya la base inicial de un jugador, sino que cuando se destruyan todos los edificios de un jugador. Los otros dos supuestos que contemplamos al momento de realizar el trabajo es que se puedan dañar las construcciones antes de que pasen todos los turnos para quedar efectivas, que las unidades no posean un rango de movimiento y que la evolución de Mutalisco a Guardián o Devorador debe agregarse esta unidad y coincidir con la ubicación del Mutalisco en el mapa. El último supuesto es que los jugadores comenzaran con 225 minerales en lugar de 200.

3. Modelo de dominio

El programa comenzará con la creación de un Mapa, que contiene unas construcciones y unas zonas. El jugador, que puede pertenecer a dos tipos de razas distintas (Zerg y Protoss) tiene recursos y una lista de edificios que puede construir. Mediante la clase Jugador es que se distribuye todo el flujo del juego. Esta clase no solo está encargada de realizar las validaciones correspondientes de los inputs para inicializar una partida, sino que es desde aquí que se empiezan a manejar los ataques, correlatividades y agregado de construcciones e individuos.

Existe una clase madre Edificio de la cual heredan 10 tipos de edificios, 5 para la raza Zerg y otros 5 para la raza Protoss. Por otra parte hay 3 tipos de zonas: con moho, de energía y neutrales que heredan atributos y métodos de una clase madre Zona. El jugador podrá agregar una construcción al mapa siempre y cuando cumpla con las condiciones necesarias especificadas en la consigna, y al pasar de los turnos requeridos por cada construcción particular, queda construido el edificio. A partir de ahí cada edificio permite una extracción de un mineral particular y engendrar un tipo de bicho diferente. Hay algunos edificios que requieren de que otros estén construidos previamente para poder edificarse. Para comprobar esta validación es que utilizamos una interfaz llamada CorrelativaEdificio. El mismo caso aplica para los individuos, solo que ahí la interfaz se llama CorrelativaIndividuo logicamente.

Existen diferentes tipos de individuos, 7 pertenecientes a la clase Zerg y 3 a la clase Protoss. Sin importar la raza, estos individuos poseen una clase madre llamada Individuo. Mediante esta herencia es que se permite aplicar polimorfismo al momento del ataque, ya que existen dos tipos de individuos que son las unidades aéreas y las terrestres, algunos pueden atacar a construcciones y otros individuos y otros solo a estos últimos.

Hay ciertas clases como Volcan o Mena que pueden ser "trabajadas"por un zángano o por otros edificios. Para que esta clase trabajadora sepa cual es el Volcán o Mena sobre el que deben trabajar utilizamos una interfaz RecursoInyectable, que implementan estas últimas dos mas indican a la clase que reciben por parámetro en el método inyectarRecurso que es a ellas a quién deben extraerle el gas o el mineral.

Por último, en la interfaz gráfica, utilizamos el patrón de diseño Observer, con las interfaces Notificable y Notificador que nos permite indicar cuando un turno ya pasó y es el momento del otro jugador, para indicar que el juego terminó y también para indicar que queden inhabilitadas las unidades que no pertenecen al jugador del turno correspondiente.

4. Diagramas de clase

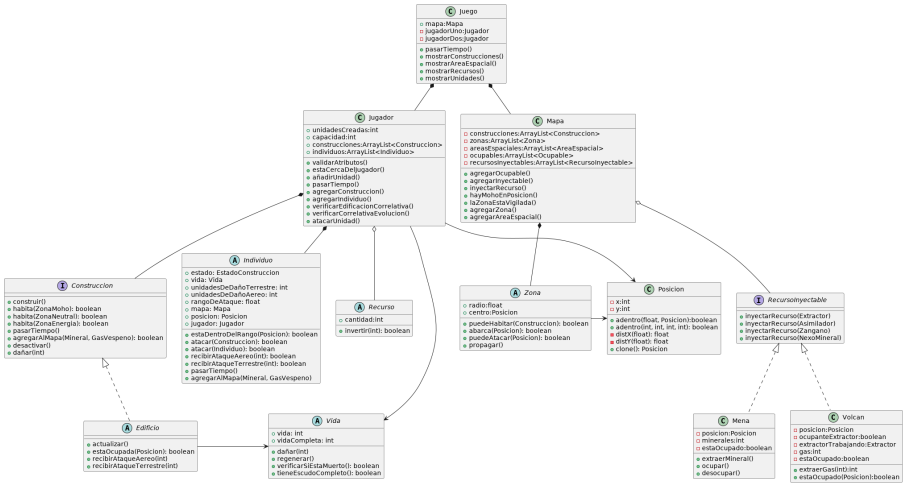

Figura 1: Diagrama de clases principal, que muestra la relación entre las clases más importantes del modelo.

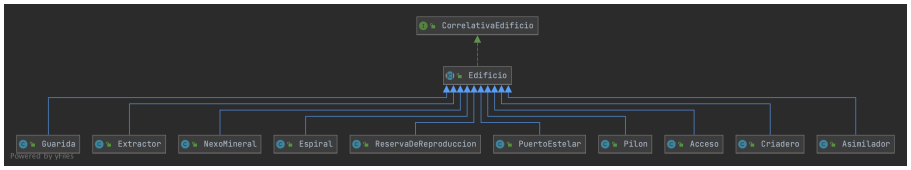

Figura 2: Diagrama de clases que muestra la relacion entre la clase madre Edificio y las que heredan de ella.

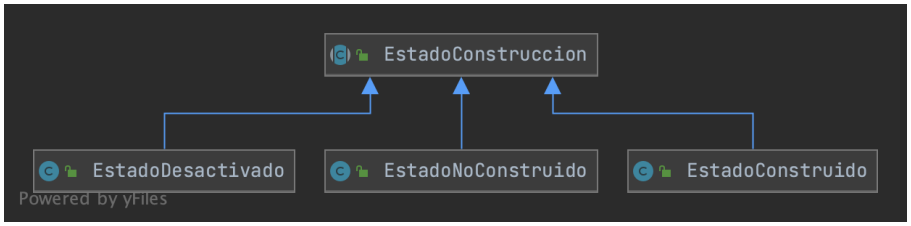

Figura 3: Diagrama de clases que muestra la relacion entre la clase madre EstadoConstruccion y las que heredan de ella.

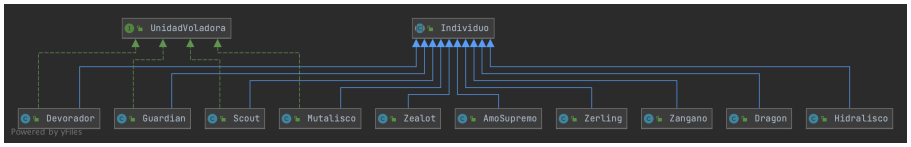

Figura 4: Diagrama de clases que muestra la relacion entre la clase madre Individuo y las que heredan de ella.

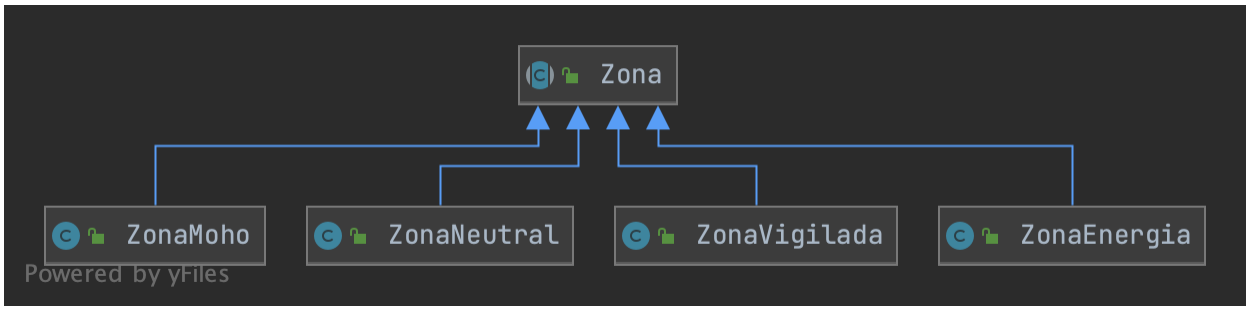

Figura 5: Diagrama de clases que muestra la relacion entre Zona y las clases que heredan de ella.

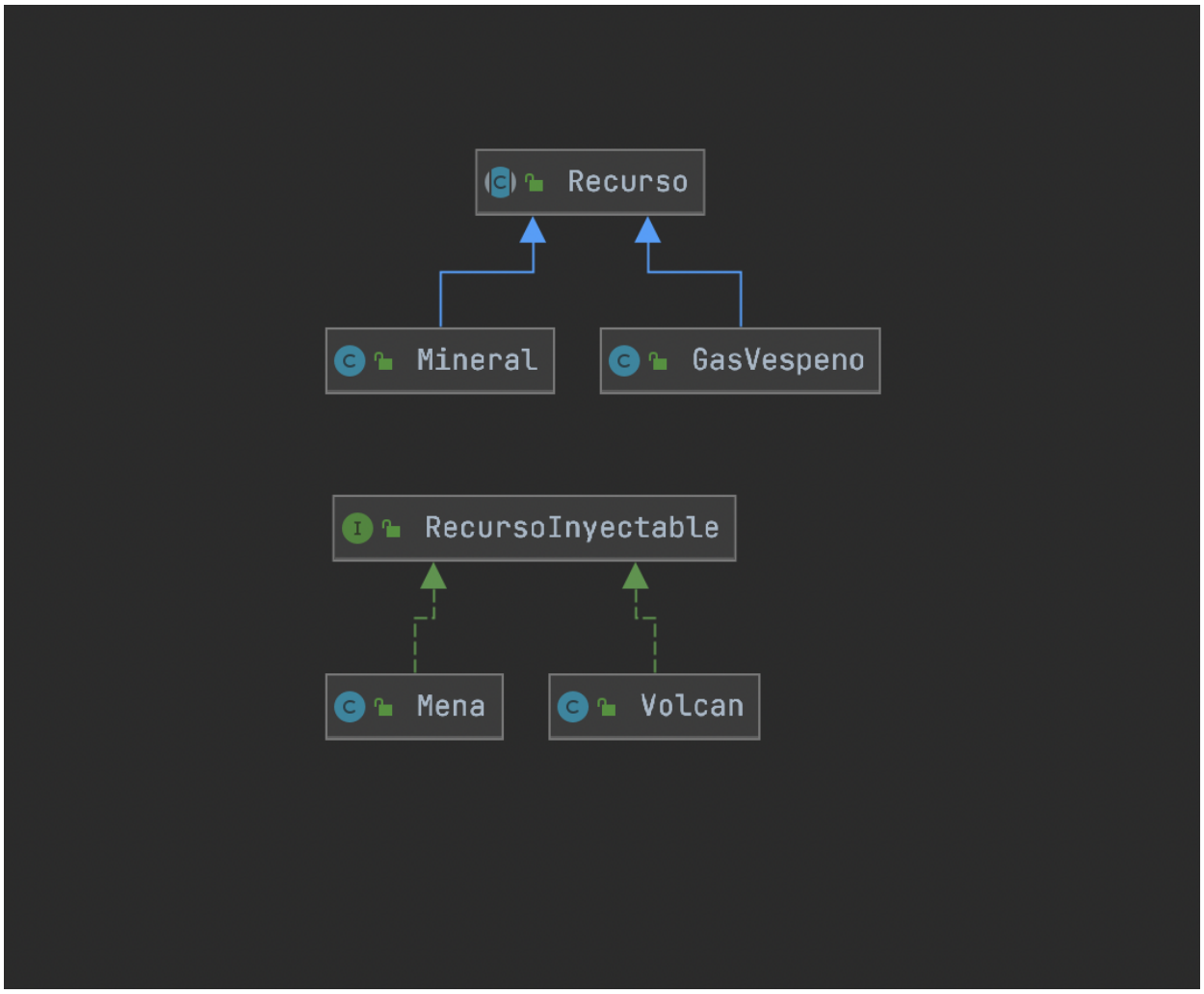

Figura 6: Diagrama de clases que muestra la relacion entre Recurso y las clases que heredan de ella.

5. Detalles de implementación

5.1. Edificio:

Es la clase madre de la cual heredan las 10 construcciones disponibles. Implementa además una interfaz Construcción. Posee distintos atributos y también métodos abstractos y entre otras cosas se encarga de: construir al edificio en sí, destruirlo, recibir daño y desactivarlo.

5.2. Individuo:

Clase abstracta madre de los 10 individuos existentes. Ataca, recibe ataques, pasa el tiempo y se mueve. También realiza algunas validaciones correspondientes a estas acciones.

5.3. Zona:

Clase abstracta madre de la cual heredan los 3 tipos de zonas: la zona con moho, la zona neutral y la zona con energía. La zona con moho es necesaria para realizar la construcción de los edificios zerg, mientras que las construcciones protoss deben estar energizadas.

5.4. Mapa:

Con una lista de construcciones y una de zonas, representa al "tablero"del juego. Se encarga de crear y destruir zonas mientras que delega la responsabilidad de crear y destruir edificios.

5.5. EstadoConstruccion:

Es la clase abstracta madre de los dos posibles estados de construcción: Construido y No-Construido. Estas dos pueden avisarle al edificio si puede ser construido basándose en los turnos actuales y los necesarios para construir y nos permite conocer si la construcción está activa.

5.6. Recurso:

Es la clase abstracta de la cual heredan los dos recursos disponibles que son los minerales y el gas vespeno. Se pueden invertir y colectar, mientras que los minerales exclusivamente pueden minar menas para adquirir más cantidad.

5.7. Mena:

Es la clase abstracta de la cual heredan los dos recursos disponibles que son los minerales y el gas vespeno. Se pueden invertir y colectar, mientras que los minerales exclusivamente pueden minar menas para adquirir más cantidad.

5.8. Volcan:

Es la clase abstracta de la cual heredan los dos recursos disponibles que son los minerales y el gas vespeno. Se pueden invertir y colectar, mientras que los minerales exclusivamente pueden minar menas para adquirir más cantidad.

5.9. VidaZerg:

Es la clase que modela y maneja la vida de las construcciones Zerg. Pueden recibir daño y también ser regeneradas.

5.10. VidaEscudoProtoss:

A diferencia de las construcciones Zerg, las Protoss también tienen un escudo. Por eso modelamos la clase VidaEscudoProtoss, que ademas de las responsabilidades que tiene la clase VidaZerg, pueden dañar y regenerar el escudo.

5.11. AreaEspacial:

Nos permite conocer si, al enviarle una posición, la misma se encuentra dentro del rango deseado para ser considerado un área espacial. Permite que se muevan los individuos de tipo aéreos.

5.12. Base:

Nos permite setear el concepto de la base, clave al inicio del juego. La clase nos permite agregar un volcán y otras construcciones con cristales para que el jugador comience su partida.

5.13. Juego:

Se utiliza desde la parte gráfica del juego. Nos permite pasar el tiempo al igual que devolver ciertos elementos para que sean visible.

5.14. Jugador:

Es la clase a partir de donde se distribuye todo el flujo del juego. Desde aqui se pueden manejar los minerales, los ataques, las construcciones y las validaciones. Se conecta con la interfaz gráfica para poder responder a las peticiones del usuario.

5.15. Posicion:

Realiza ciertos cálculos para informar si una posicion se encuentra dentro del rango deseado o no. También nos permite comparar entre dos posiciones para saber si se refieren al mismo lugar del mapa.

5.16. Ocupable:

Interfaz que nos facilita la información para saber si es posible ocupar una posición o no.

5.17. RecursoInyectable:

Interfaz que nos permite asignarle al Volcan o Mena el zángano o edificios que trabajaran sobre estas dos primeras clases.

6. Excepciones

Exception CriaderoNoDisponible: Se lanza cuando se quiere engendrar un zángano pero no quedan más larvas en el criadero.

Excepcion RecursosInsuficientes: Se arroja cuando se quiere crear una construcción pero el jugador no tiene los recursos suficientes para construirla.

Exception AccesoNoDisponible: Se lanza desde el Dragon o el Zealot cuando el jugador no cumple con las correlativas de construcción.

Exception AtributoInvalido: Se utiliza al inicio del juego, al momento de consultar el nombre, raza y color del jugador si es que hay algún problema con estos datos.

Exception EspiralNoDisponible: Se lanza desde el Devorador, Guardian o Mutalisoc cuando el jugador no cumple con las correlativas de construcción.

Exception PuertoEstelarNoDisponible: Se lanza desde el Hidralísco cuando el jugador no cumple con las correlativas de construcción.

Exception ReservaDeReproduccionNoDisponible: Se arroja desde el Zerling cuando el jugador no cumple con las correlativas de construcción.

Exception RequerimientosInsuficientes: Se arroja desde múltiples Edificios e Individuos cuando el jugador no tiene la cantidad de recursos necesarios para construir.

Exception VolcanOcupado: Se lanza cuando el volcán se encuentra ocupado

Exception GuaridaNoDisponible: La arroja el Hidralísco cuando el jugador no cumple con las correlativas de construcción.

Exception MaximaPoblacionAlcanzada: Se lanza en distintos edificios cuando no se pueden generar más individuos

Exception MenaOcupada: Como su nombre lo indica, se utiliza cuando la mena ya está ocupada

Exception NoExisteEdificioCorrelativo: Se arroja cuando se le pide comprobar una correlatividad con un edificio pero este no existe

Exception NoSeEncuentraAlIndividuo: Se lanza cuando se intenta atacar a un individuo pero el mismo no puede ser encontrado dentro de los que posee el otro jugador

7. Diagramas de secuencia

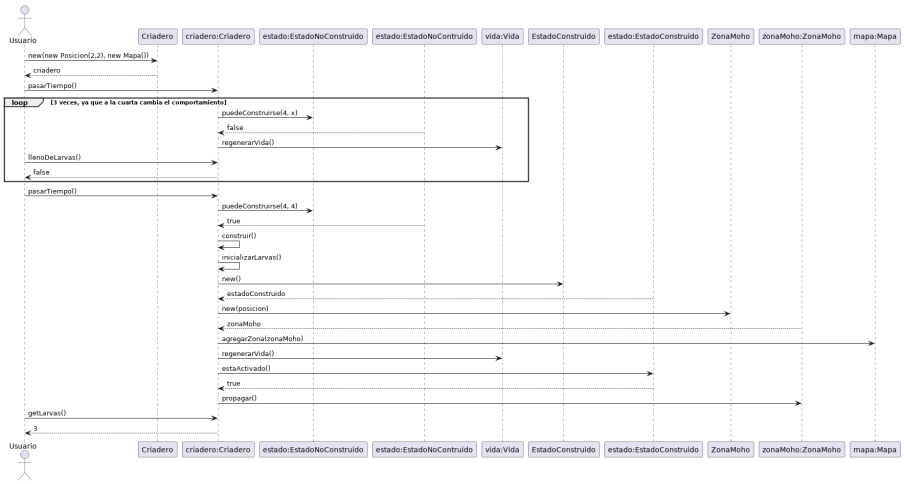

Figura 7: Diagrama que recrea una prueba del caso de uso 2.

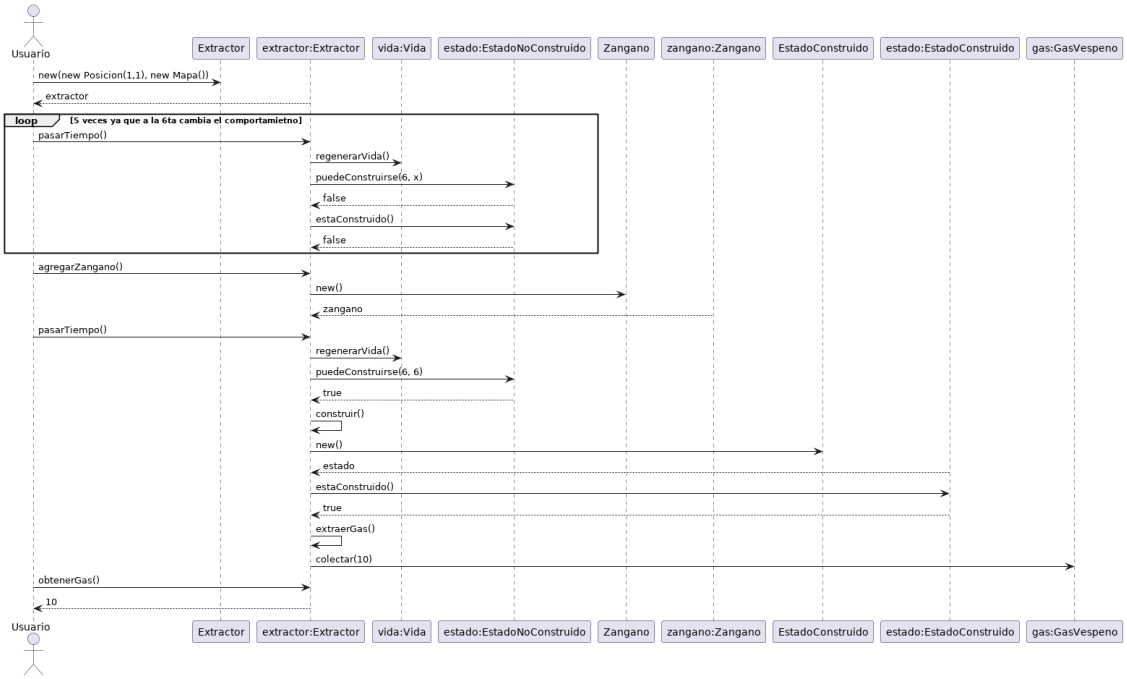

Figura 8: Diagrama que recrea como funciona un Extractor con un Zángano a la hora de extraer el gas.

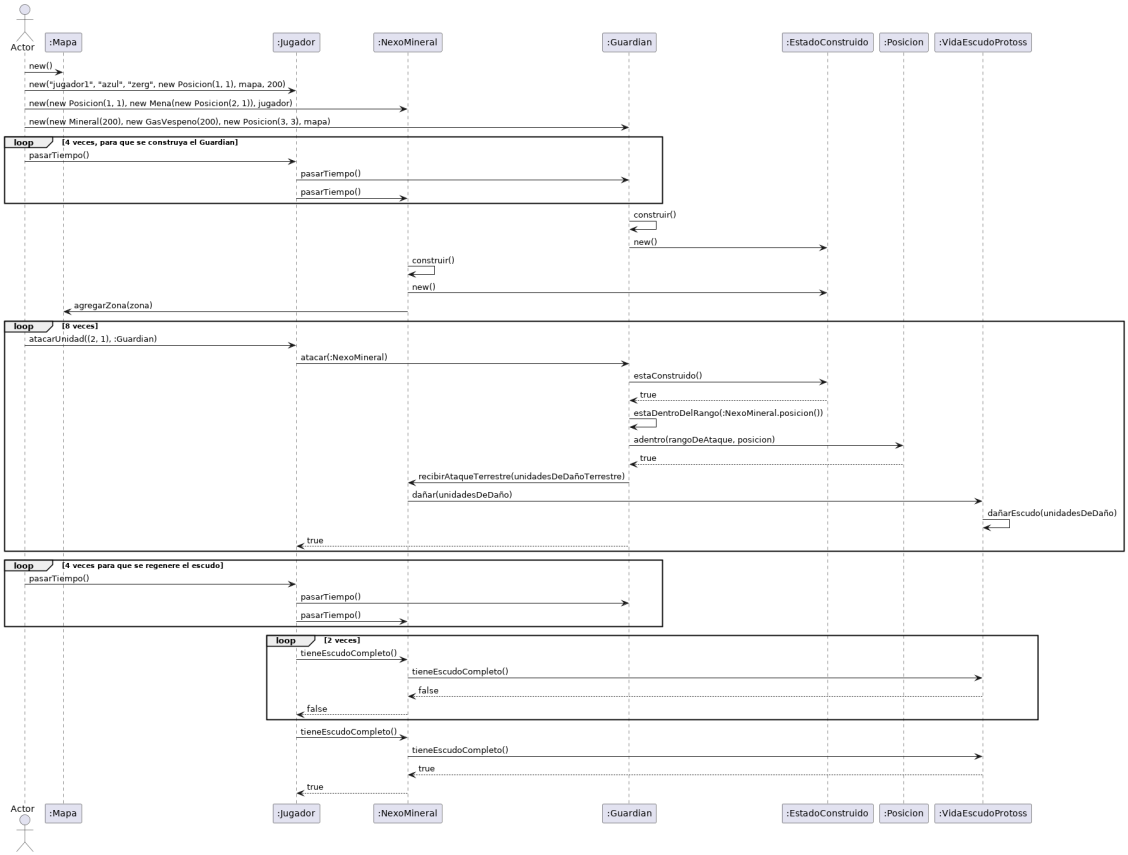

Figura 9: Diagrama correspondiente al test del caso de uso 18.

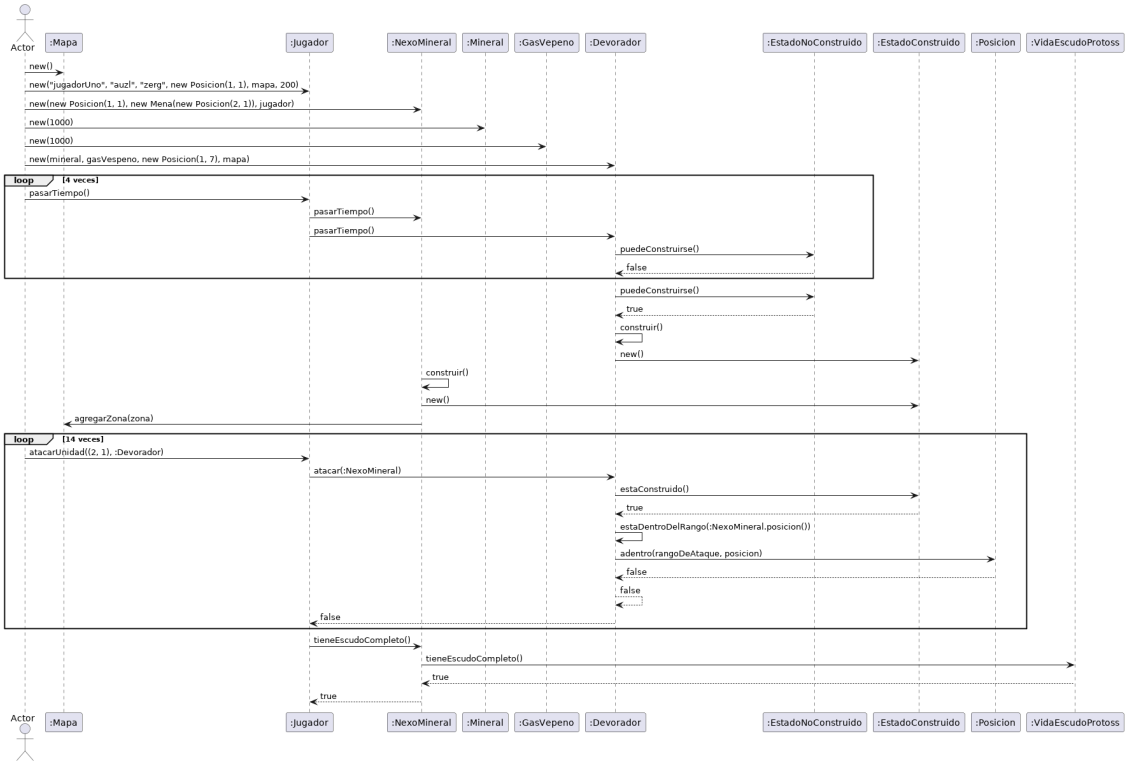

Figura 10: Diagrama correspondiente a un test del caso de uso 27.

8. Diagramas de paquetes

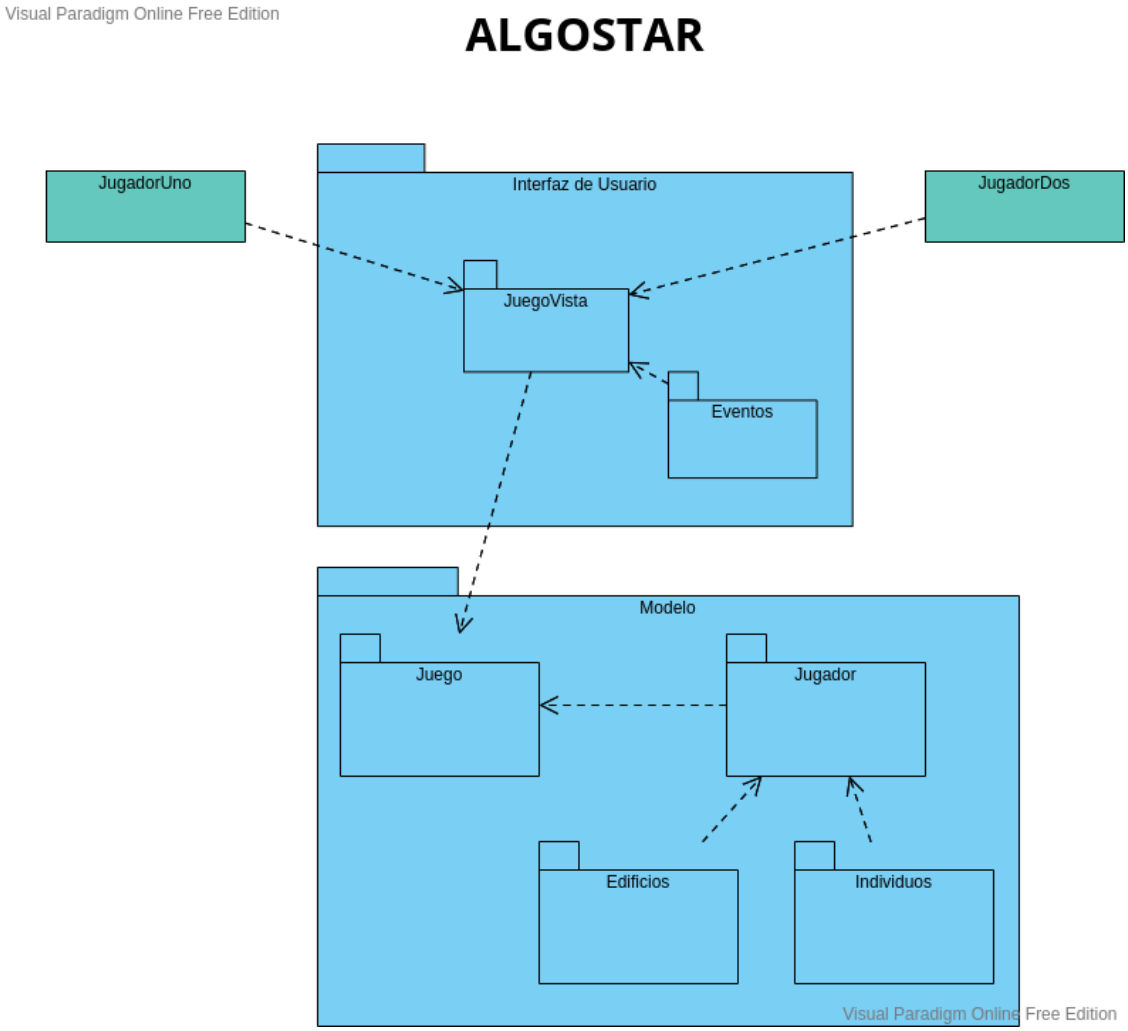

Figura 11: Diagrama de paquetes.