FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — AlgoStar

[7507/9502] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre de 2022

INTEGRANTES:

Lucas Perez Esnaola - #107990
< *lpereze@fi.uba.ar* >

Tomás Pierdominici - #106439
< *tpierdominici@fi.uba.ar* >

Valentin Brizuela - #108071
< *vabrizuela@fi.uba.ar* >

Rubin, Iván - #100577
< *irubin@fi.uba.ar* >

Índice

1. Introducción

El presente informe reúne la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un juego 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

- Ambos Jugadores tienen razas distintas (Zerg o Protoss).
- El numero de bases es par.
- El número de acciones entre turnos es ilimitado
- Las unidades se pueden crear y mover en cualquier rango.
- Se puede construir un criadero dentro del rango del Pilón.

3. Diagramas de clase

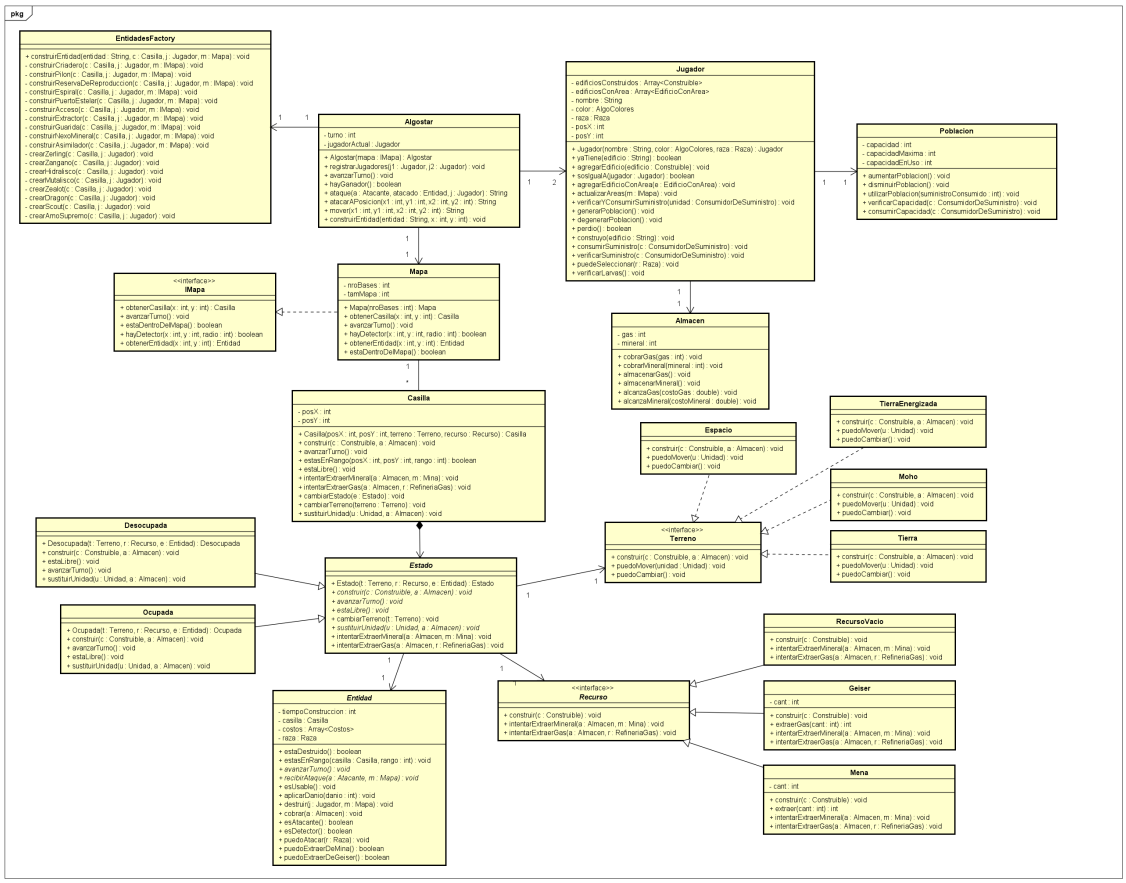

Figura 1: Diagrama de clases general.

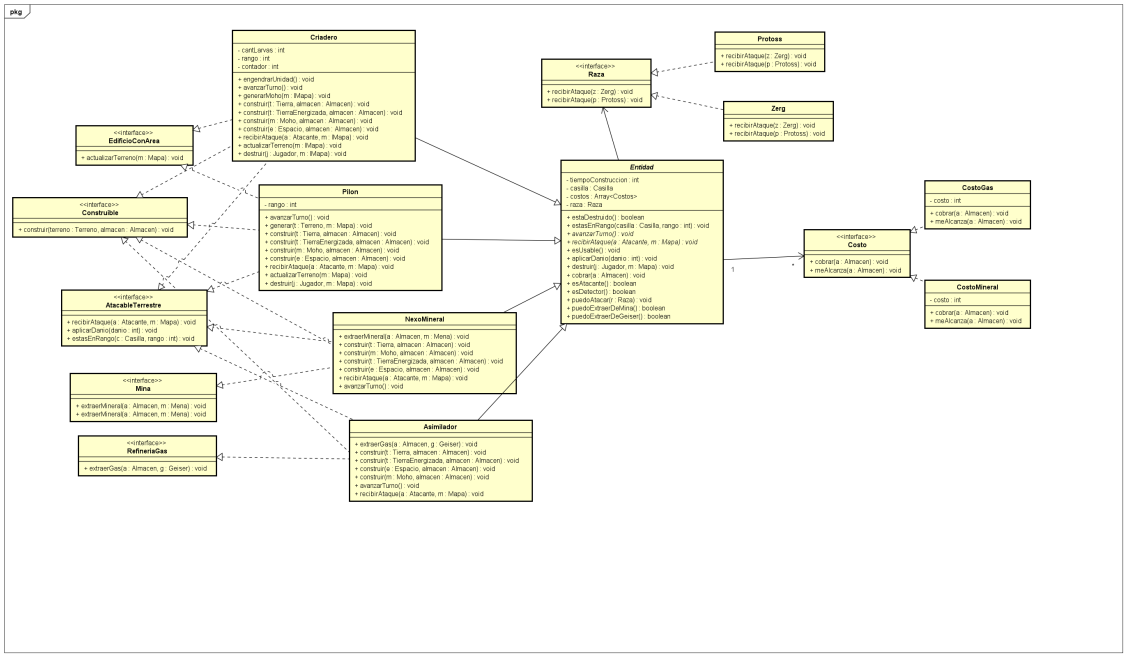

Figura 2: Diagrama de clases de los Edificios.

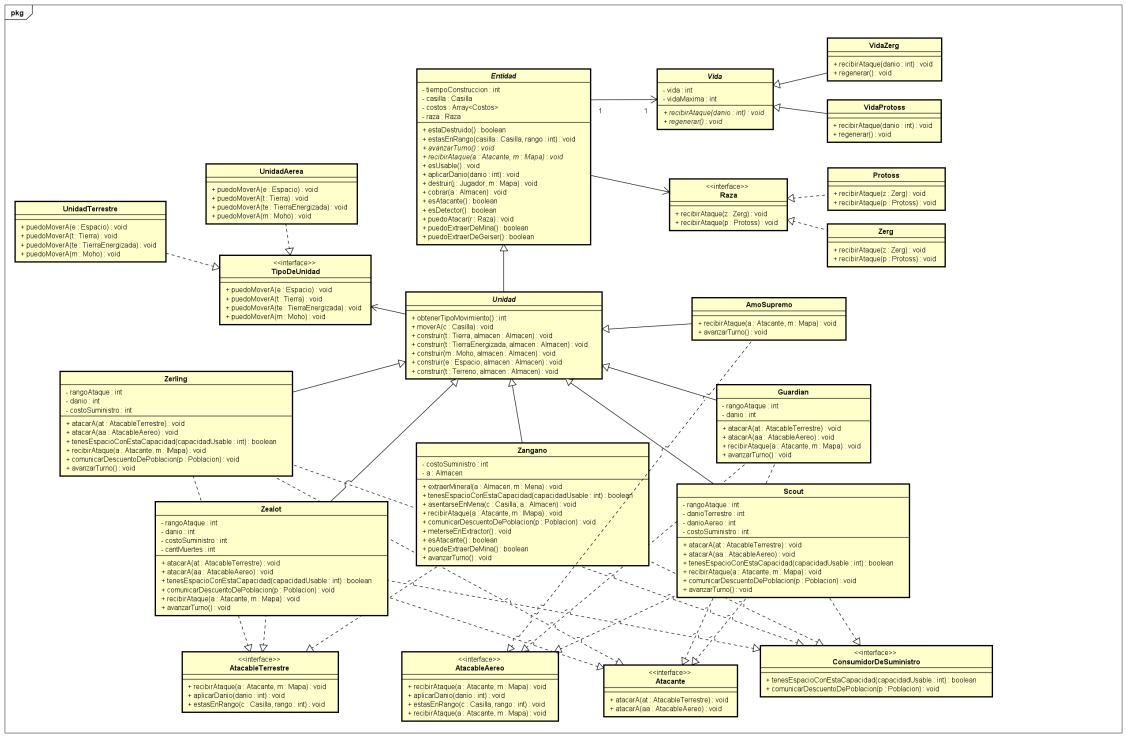

Figura 3: Diagrama de clases de las Unidades.

4. Diagramas de secuencia

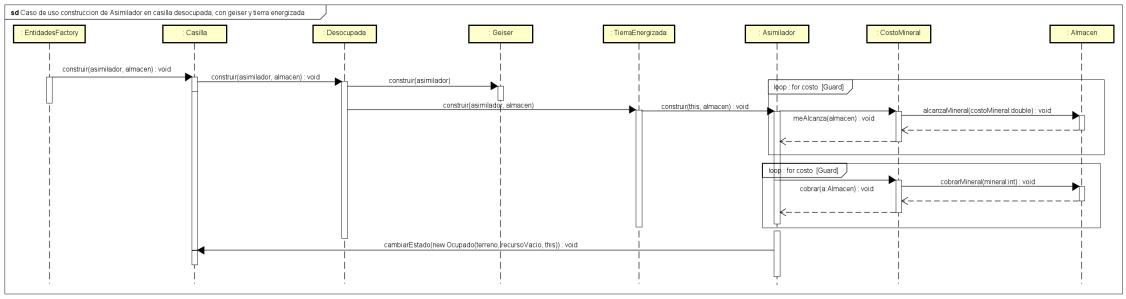

Figura 4: Caso de uso construcción de Asimilador en casilla desocupada, con Geiser y tierra energizada.

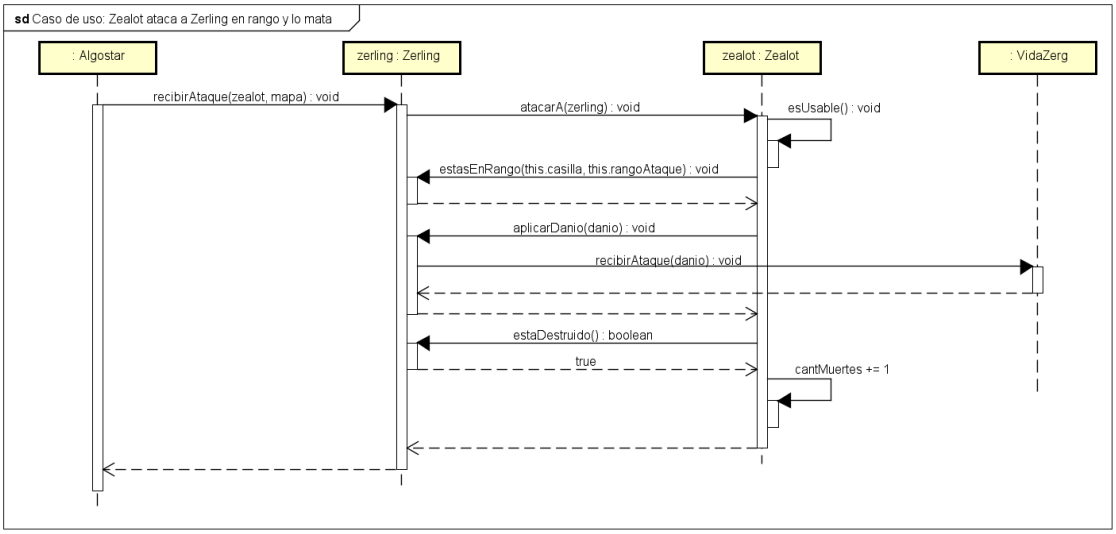

Figura 5: Caso de uso Zealot ataca a Zerling en rango y lo mata.

5. Diagrama de paquetes

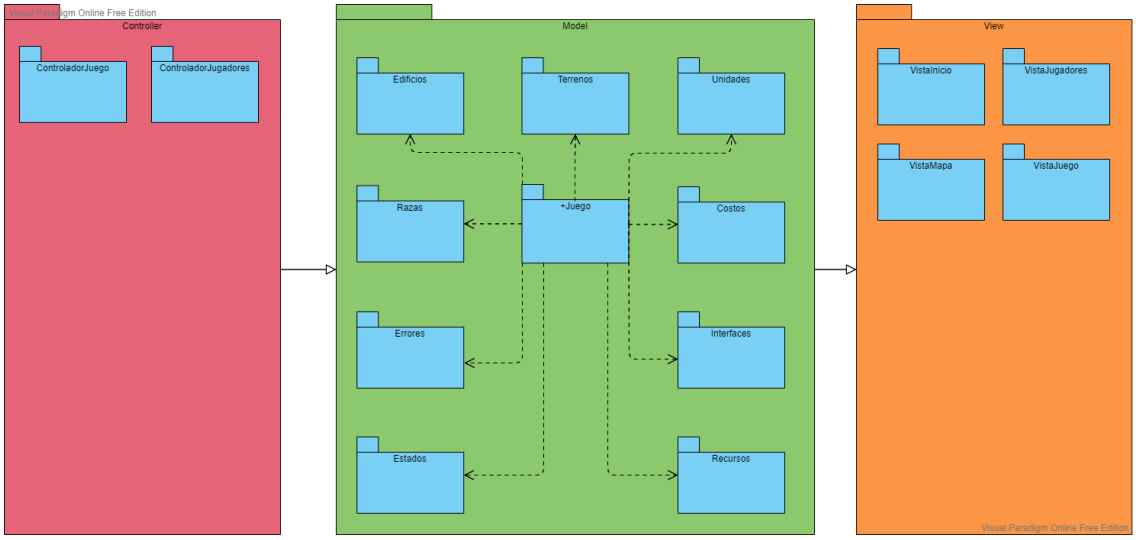

Figura 6: Diagrama de paquetes.

6. Diagramas de estado

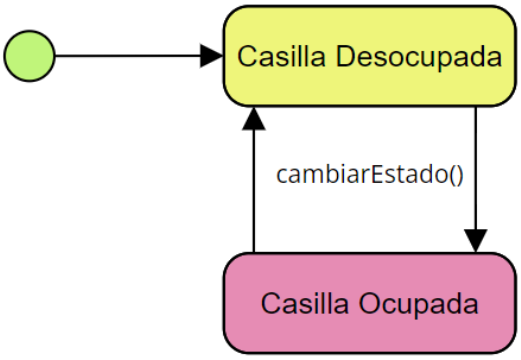

Figura 7: Diagrama de Estado de la casilla.

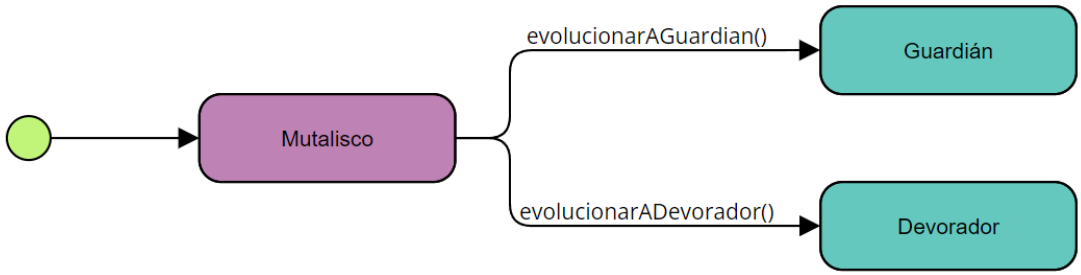

Figura 8: Diagrama de Estado del Mutalisco.

Figura 9: Diagrama de Estado de los edificios.

7. Detalles de implementación

7.1. Double Dispatch

Se utilizó Double Dispatch para los ataques entre unidades. Con las interfaces **Atacante**, **AtacableAereo** y **AtacableTerrestre** se pudo definir que unidades podian atacar a otras según si eran Aereas o Terrestres. El double dispatch se aplica en los objetos que implementan la interfaz **Atacante** ya que son obligados a definir un método *atacarA()* que recibe un **AtacableAereo** y otro del mismo nombre que recibe un **AtacableTerrestre**.

7.2. Patrón State

Se aplicó el patrón State para el estado de las casillas del mapa. Estas poseen un atributo Estado que cambia cuando la casilla es ocupada o desocupada. Según el estado en el que esté la casilla se le delega a Estado distintos métodos como *construir()* y *sustituirUnidad()*.

7.3. Patrón Null

Este patrón para evitar chequeso de *null* a lo largo del código se implementó con la clase *RecursoVacio*.

7.4. Patrón Factory

Se utilizó el patrón Factory para la creación de las unidades. Para esto se creó la clase *EntidadFactory* donde se utiliza la función *switch( )* para decidir que entidad se debe crear de acuerdo al nombre de edificio que le llega.

7.5. Delegación por sobre herencia

Como se vió a lo largo del curso, en la mayoría de los casos conviene utilizar delegación por sobre herencia ya que se evita que las clases hijas estén encasilladas por los métodos que heredan

de la clase padre. Con delegación las clases se vuelven mas flexibles lo cual permite tener clases más fácilmente expansibles. Un ejemplo de esta implementación se puede ver en la clase *Jugador* donde hay un atributo *Raza* que contiene la raza a la cual pertenece cada jugador. Con herencia se podría haber creado dos clases que hereden de jugador llamadas "JugadorZergz "JugadorProtoss"pero de esta manera estas clases hubieran heredado metodos que no iban a utilizar.

8. Excepciones

8.1. Jugador

- JugadoresInsuficientesError
- JugadorInvalidoError
- NombreDeJugadorInvalidoError
- NroBasesImparesError
- PoblacionInsuficienteError
- RecursosInsuficientesError

8.2. Ataques

- AtaqueInvalidoError
- AtaquePorAireInvalidoError
- AtaquePorTierraInvalidoError

8.3. Casillas

- CasillaOcupadaError
- FueraDeRangoError
- MovimientoAEspacioError
- NoHayGasEnLaCasillaError
- NoHayMenaEnLaCasillaError
- SeleccionInvalidaError

8.4. Construcción de Unidades

- ConstruccionNoPermitidaError
- ConstruccionNoPermitidaRecursoError
- ConstruccionNoPermitidaTerrenoError
- CreacionDeUnidadInvalida
- EnConstruccionError
- ExtractorLlenoError
- GeneracionInvalidaError
- LarvasInsuficientesError

- UnidadNoPuedeAtacarError
- ZealotInvisibleError