FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 - AlgoPoly

[7507/9502] Algoritmos y Programación III
Curso 2
Segundo cuatrimestre de 2017

Alumno	Numero de padrón	Mail
VASQUEZ, Agostina	99689	agostinavasquezisla@gmail.com
BOADA, Ignacio	95212	ignacio.boada@outlook.com
ROSSINI, Federico	97161	federicorossini09@gmail.com
CASTRO LOPEZ, Juan Cruz	96954	juan.castrolopez@outlook.com
BRACCELARGHE, Ailín	99366	ailinyelen@hotmail.com

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 una aplicación que implemente un juego relacionado con el clásico juego de mesa MonoPoly, utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso. Para el desarrollo se aplicará el Patrón de diseño Model-View-Controller (MVC).

2. Supuestos

Retroceso Dinámico: cuando el cálculo de los casilleros a retroceder da menor a cero, se supone que el jugador no retrocede ni avanza ningún casillero.

Cuando un jugador cae en una propiedad de otro jugador, debe pagar el alquiler correspondiente a las construcciones que haya en ese barrio. En el caso de los barrios que tienen sur y norte, no importa las construcciones que haya en la otra división a la hora de cobrar el alquiler.

Se supone que dos jugadores no pueden empatar. El juego finaliza únicamente cuando, al quedar solo 2 jugadores, un jugador cae en un casillero en el que está obligado a pagar y no tiene capital suficiente para hacerlo luego de vender sus propiedades.

Cuando un jugador decide vender alguno de sus barrios, cobra un 85 % de la suma de lo que vende, suponiendo que la suma de lo que vende es el precio del terreno.

Se supone que el jugador solo puede realizar acciones voluntarias como comprar, vender, construir, etc, previo a tirar los dados. Una vez que lanza los dados y mueve finaliza el turno, a menos que haya caído en un casillero en el que deba pagar y no tenga capital suficiente. En tal caso estará obligado a vender propiedades.

El intercambio de una propiedad solo se puede realizar si el jugador esta parado sobre la propiedad la cual es dueño y quiere intercambiar. El intercambio no se puede hacer en el mismo turno en que se compra la propiedad.

3. Modelo de dominio

Las entidades que tienen un lugar central en el desarrollo del juego 'AlgoPoly' son la clase Jugador y la Interfaz Encasillable. Con ellas se logra resolver el primer problema que plantea el juego que es aplicar la acción de un casillero sobre un jugador cuando este cae sobre él, haciendo que todos los casilleros implementen Encasillable, permitiendo que el método caerEn del jugador reciba como parámetro un Encasillable. Por lo tanto el jugador y los casilleros se relacionan a través de 2 métodos, uno es el método caerEn de Jugador que ejecuta el método actuarSobre que poseen todos los casilleros a través del cuál afectan al jugador dependiendo de la implementación de ese método en cada casillero.

El Jugador es responsable de conocer su capital, y poder incrementarlo y decrementarlo, debe saber el casillero en el que se encuentra, tener su boleta de Quini6 (explicado más adelante), su nombre, las propiedades que posee, su Estado (que depende de la Cárcel), y el Tablero que se utiliza para mover al jugador dependiendo de su estado. El Jugador se relaciona prácticamente con todas las entidades del modelo porque es el personaje principal del juego.

La clase Juego, implementada con el patron Singleton, es el punto de partida del juego, siendo su principar responsabilidad la administrar los turnos de los jugadores, eliminar un jugador cuando este pierde y saber cuando finaliza el juego. También crea el tablero que será utilizado a lo largo de la partida.

La clase Tablero tiene la responsabilidad de instanciar todos los casilleros, los cuales guarda en una lista para tener la posición de cada casillero de acuerdo al lugar que ocupa en la lista. Debido a que solo esta clase conoce las instancias de los casilleros le asigna al jugador el casillero en el que cae cuando este mueve, y es la responsable de hacerlos disponibles a las otras clases que los necesiten.

Se creó la clase Dinero con la finalidad de abstraer el tipo de dato utilizado para el manejo del dinero a lo largo del juego, de esa manera, en caso de tener que pasar de int a double, o de tener que implementar otro tipo de dato para el manejo del dinero, simplemente se deberá cambiar esa clase, que es la responsable de saber cómo se maneja el dinero (cómo aumentarlo, disminuirlo, multiplicarlo, etc).

Se creó una clase por cada casillero ya que cada casillero del juego realiza una acción diferente sobre el Jugador que cae en él, y la idea es abstraer la responsabilidad de cada casillero para evitar el acoplamiento.

Entre los casilleros hay dos grandes subdivisiones que son los casilleros que pueden ser comprados por jugadores (como Buenos Aires Sur, Santa Fe, etc) y los que no (como Cárcel, Quini6, etc). Para esta división se creó la clase abstracta Propiedad, la cual implementan las clases de los casilleros que pueden ser comprados por jugadores.

Dentro de los casilleros se implementaron distintas lógicas para resolver distintos problemas:

1. Quini6: Para este casillero se desarrolló una lógica para utilizar polimorfismo en lugar de condicionales, la cual se basa en que cada jugador tenga su boleta de Quini6, y cada boleta sabe qué premio debe cobrar el jugador que cae en Quini6. Para ello están las clases Premio, PrimerPremio, SegundoPremio y PremioSinValor. La clase premio es abstracta para que todos los premios hereden de esta y poder usar polimorfismo.

2. Cárcel y Policía: Al caer en el casillero de la policía, al jugador se le aplica el mismo proceso que cuando cae en la cárcel. Cuando el jugador cae en la cárcel, va preso, lo que hace que cambie su estado de Libre a PresoTurno0, el proximo turno será PresoTurno1, el siguiente PresoTurno2, cuando ya se le habilita pagar la fianza, y el último PresoTurno3. Para esto guarda su estado el Jugador. De esta manera no es necesario usar condicionales para saber en qué número de turno está el jugador en la cárcel.

3. Dentro de los casilleros apropiables hay dos grandes subdivisiones, los barrios y las compañías:

   - Respecto a los barrios, hay barrios de una sola zona y barrios que tienen zona norte y sur, se los llama barrio simple y barrio dividido, respectivamente. Cada casillero que tiene una sola zona, como Santa Fé, hereda el comportamiento de la clase BarrioSimple y utiliza los atributos propios del casillero. Los casilleros con dos zonas son atributos de una región. Se creó esta lógica para encapsular el comportamiento relativo a la construcción y demolición de casas y hoteles en la clase Región, ya que conoce tanto a la zona norte como a la sur, algo necesario para poder construir. Mientras que Barrio Dividido contiene el comportamiento que compete a cada zona en particular.
   - Para las compañías se utiliza una lógica donde los casilleros heredan el comportamiento de la clase abstracta Compania, donde se encapsula todo el comportamiento que debe tener una compañía, luego cada casillero tiene sus atributos según su precio y forma de cobrar. Y se crearon las clases ServiciosDeTransporte y ServiciosPublicos cuya responsabilidad es conocer a las compañías que les corresponden y efectuar el cobro cuando un jugador cae en ellas, ya que si se tienen las dos compañías, el cobro es diferente a si se tiene solo una.

4. Diagramas de clase

En los diagramas se muestran los atributos y métodos principales de las clases.

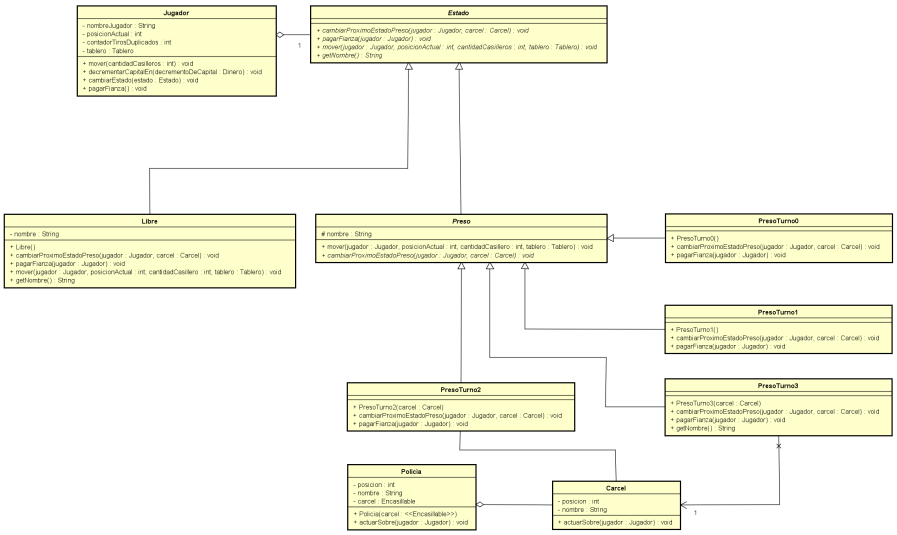

Figura 1: Diagrama de Clases de Jugador con sus estados.

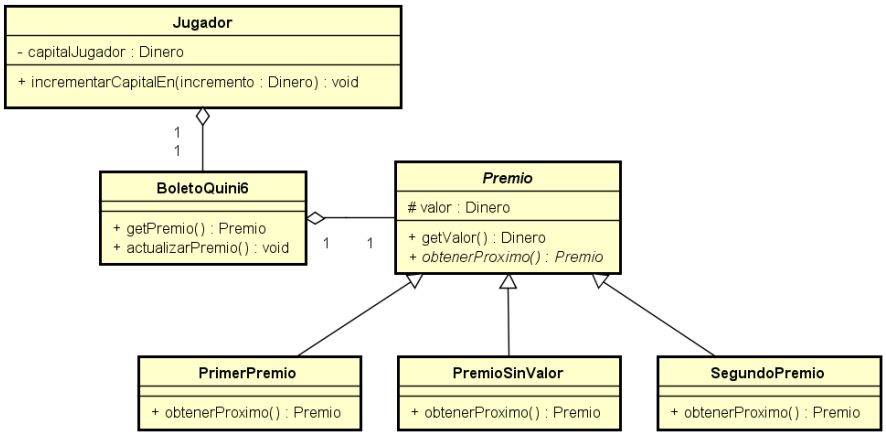

Figura 2: Diagrama de Clases de Jugador con su Boleto del Quini6.

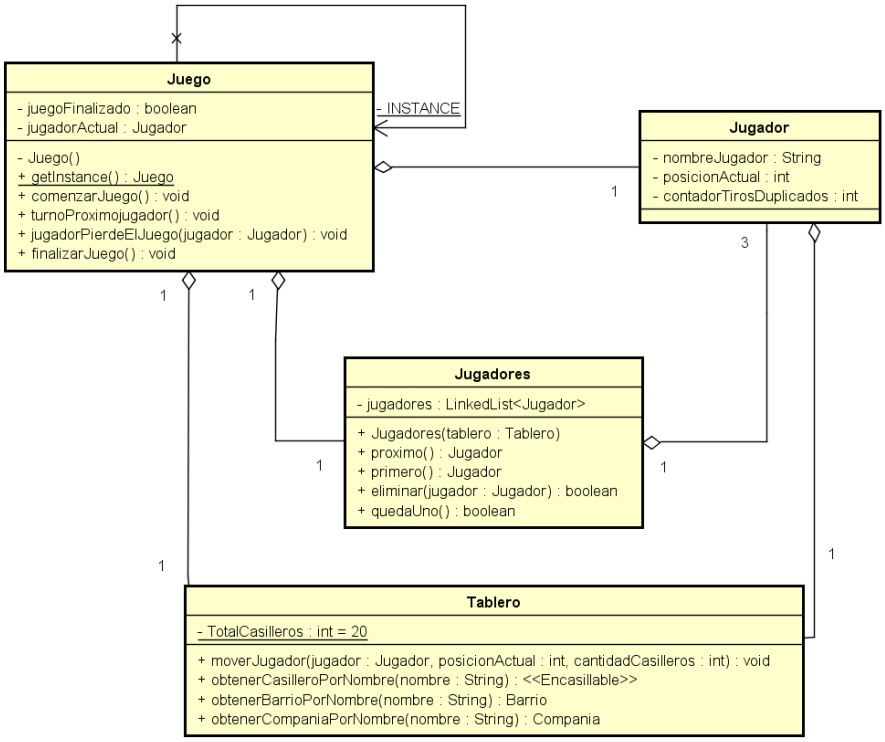

Figura 3: Diagrama de Clases de la logica del juego.

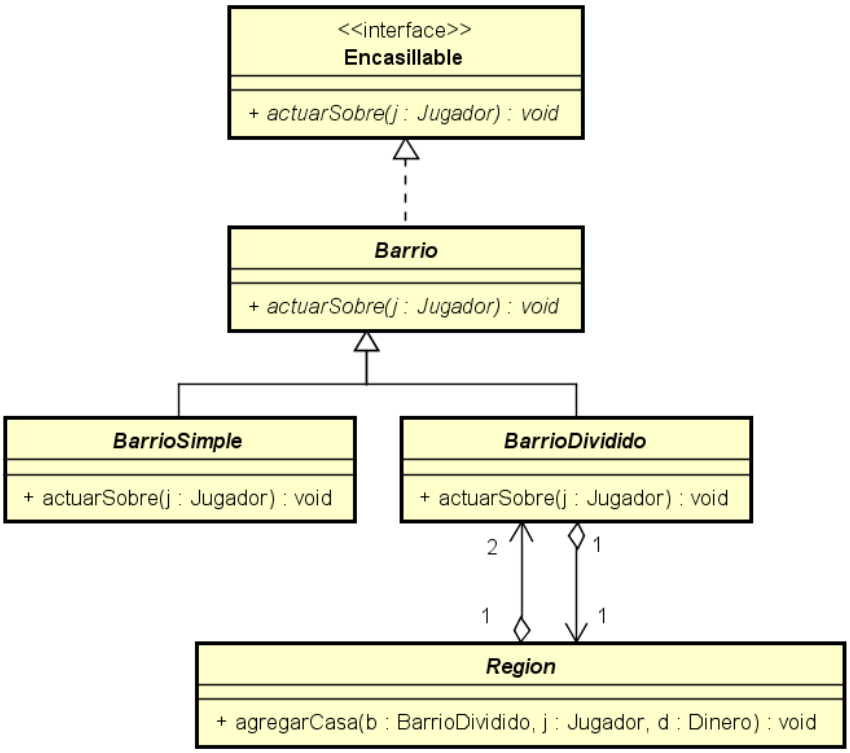

Figura 4: Diagrama de Clases del Barrio y sus subclases.

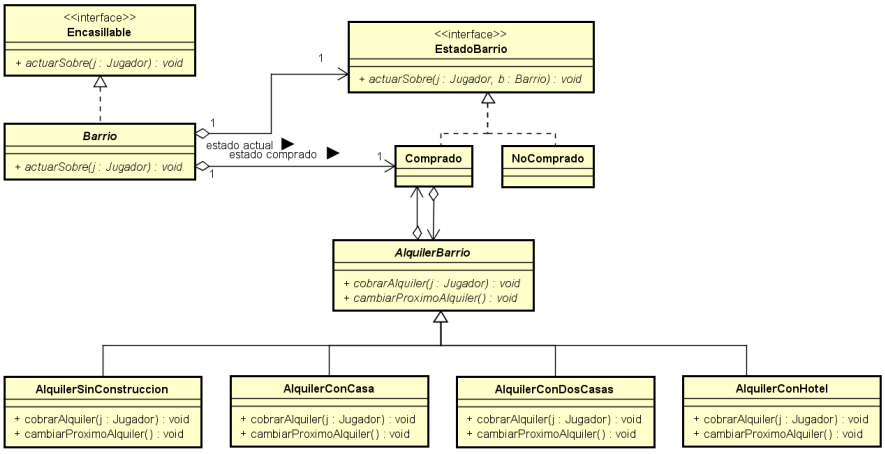

Figura 5: Diagrama de Clases de Los distintos tipos de alquiler a cobrar en un Barrio.

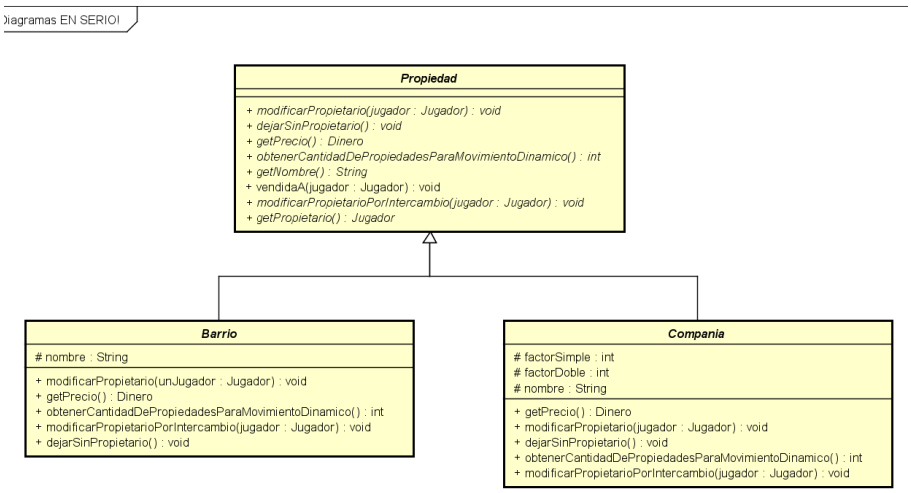

Figura 6: Diagrama de clase Propiedad y sus subclases.

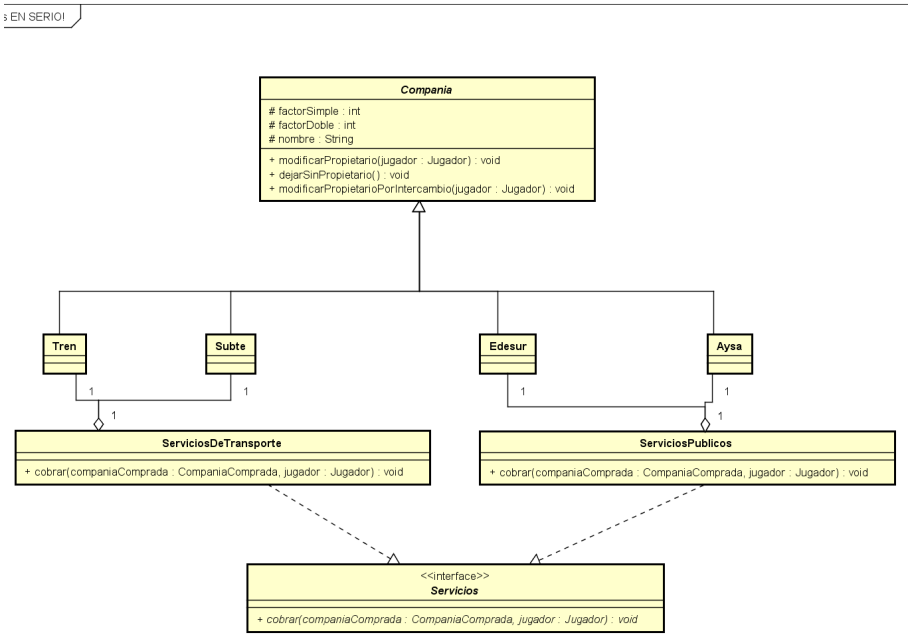

Figura 7: Diagrama de clase Compañia e interfaz.

5. Diagramas de secuencia

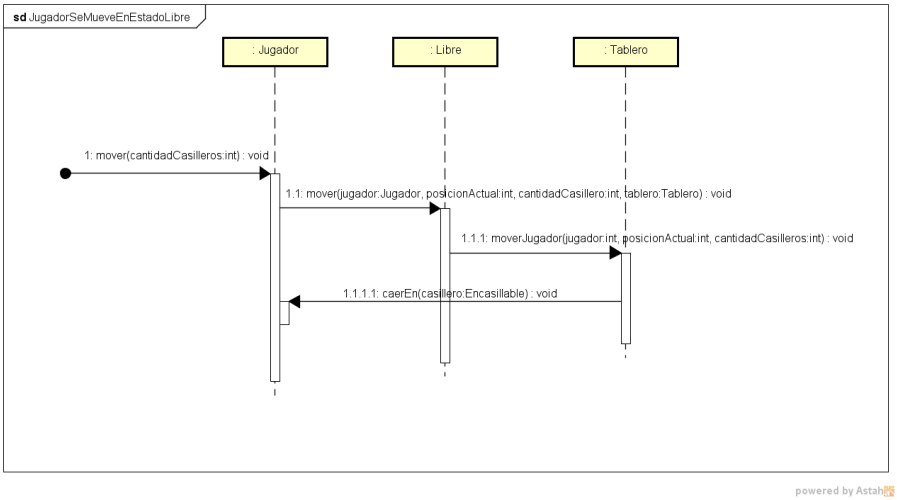

Figura 8: Secuencia del movimiento de un jugador libre.

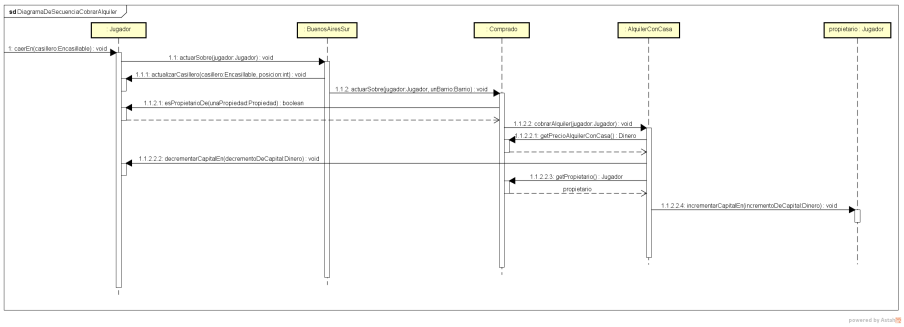

Figura 9: Secuencia del cobro de alquiler Barrio.

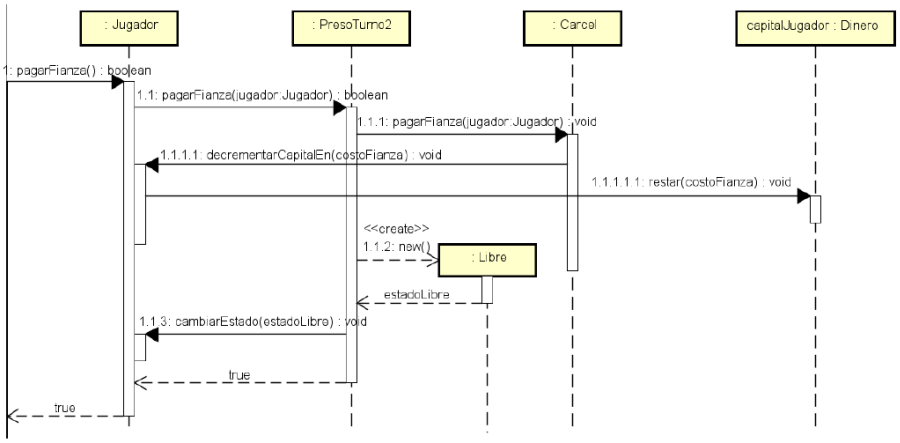

Figura 10: Secuencia Pagar Fianza de un jugador preso.

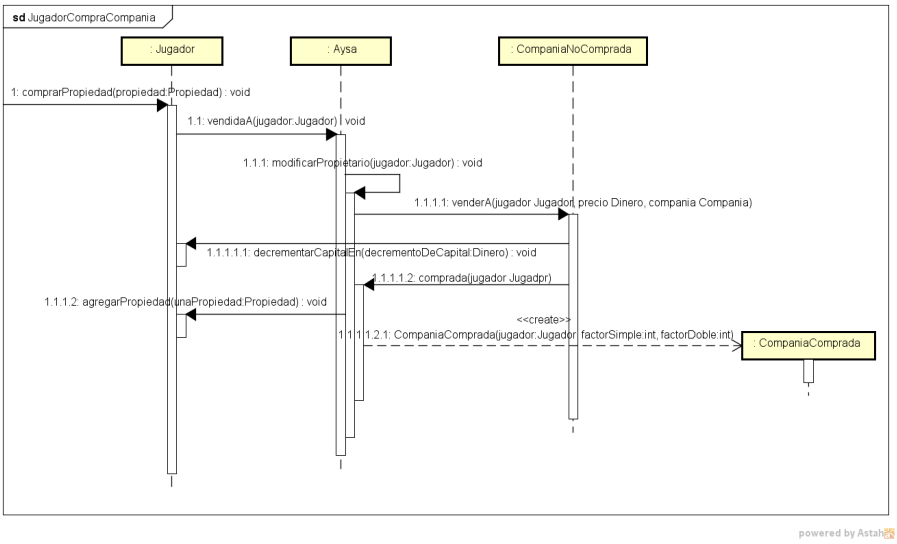

Figura 11: Secuencia Jugador compra compañía.

6. Diagramas de paquetes

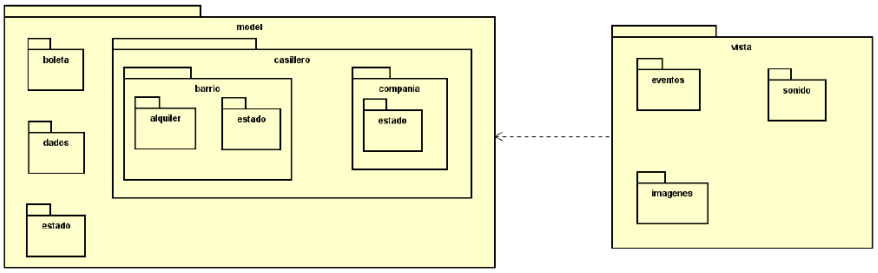

Figura 12: Diagrama De Paquetes del juego.

7. Diagramas de estado

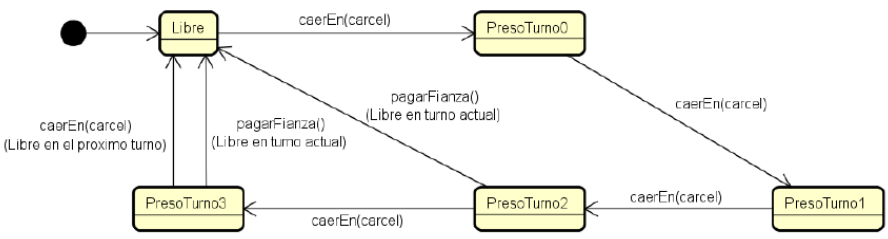

Figura 13: Diagrama de estados del jugador.

8. Detalles de implementación

8.1. Clase Dados implementa el patrón Singleton

La clase dados con constructor privado, guarda una instancia de sí mismo, y además, una instancia de la clase dado, por cada dado que se utilice en el juego, en este caso, dos.

public TiroDeDados tirar() return new TiroDeDados(dadoUno.tirar(), dadoDos.tirar());

8.2. Cobrar alquiler en compañias

Cada compañia tiene un estado que puede ser comprada, o no comprada. Si la compañia está comprada, el actuar sobre del jugador delega el cobrar alquiler al estado, y este mismo a la clase Servicios, que puede ser de transporte o públicos. En esta clase se verifica si el dueño de la compañia en cuestion, tambien posee la compañia relacionada (En el caso de tren, es el subte, y en el caso de aysa es edesur, y viceversa). Luego se llama al cobrar simple o doble segun corresponda.

public void cobrar(CompaniaComprada companiaComprada, Jugador jugador) Jugador duenio = companiaComprada.getPropietario(); if (duenio.esPropietarioDe(subte) && duenio.esPropietarioDe(tren)) companiaComprada.cobrarDoble(jugador); else companiaComprada.cobrarSimple(jugador);

8.3. Cobrar alquiler en barrios divididos

El barrio, al igual que compañia, tiene un estado comprado que se encarga de cobrar alquiler. En este estado se guardan las distintas clases correspndientes a los distintos tipos de alquiler según la cantidad de casas y hoteles, que heredan de AlquilerBarrio. En la instancia de alquiler actual es donde se guardan los precios que corresponden a casa barrio en particular.

public abstract class AlquilerBarrio

public abstract void cobrarAlquiler(Jugador jugador);

public abstract void cambiarProximoAlquiler ();

8.4. Construccion de casas en barrios divididos

Cada barrio dividido (norte y sur) se relacionan mediante una region. Los métodos agregar casa y agregar hotel, se delegan a esta clase, donde verifica las condiciones necesarias para construir en este tipo de barrio. ejemplo: construir casa.

public void agregarCasa (BarrioDividido barrio, Jugador jugador, Dinero costoCasa) throws CapitalInsuficienteException

if (jugador.esPropietarioDe(barrioSur) && jugador.esPropietarioDe(barrioNorte)) jugador.decrementarCapitalEn barrio.sumarCasa();

8.5. Eliminacion de jugador

Cuando se produce la eliminacion de un jugador, al ser un caso particular, el turno pasa automaticamente al jugador siguiente, sin necesidad de decirle al juego que lo haga.

9. Excepciones

ElDineroNoPuedeSerNegativoException: Esta excepción lanza una señal cuando se intenta decrementar la cantidad de dinero por debajo de cero.

CapitalDelJugadorInsuficiente: Esta excepción lanza una señal cuando al jugador no le alcanza el dinero para realizar una acción.

ElJugadorNoTieneCapitalSuficienteParaPagarFianza: Esta excepción lanza una señal cuando el jugador quiere pagar la fianza y el precio de la fianza es mayor a su capital.

ElJugadorDebeVenderPropiedadesPorCapitalInsuficienteException: Esta excepción lanza una señal cuando el jugador cae en un barrio o una compañía y no tiene capital suficiente para pagar el Alquiler (que dependerá de las construcciones que hayan en ese casillero) o lo que deba pagar en la compañía.

NoSePermiteConstruirMasDeDosCasasEnBarrioDivididoException: Esta excepción lanza una señal cuando el jugador intenta comprar una nueva casa en un barrio que ya tiene dos casas, y no puede tener más de dos casas.

NoSePermiteConstruirMasDeUnaCasaEnBarrioSimpleException: Esta excepción lanza una señal cuando el jugador intenta comprar una nueva casa en un barrio que ya tiene una casa, y no puede tener más de una casa.

NoSePermiteConstruirMasDeUnHotelEnBarrioDivididoException: Esta excepción lanza una señal cuando el jugador intenta comprar un hotel en un barrio que ya tiene un hotel, y no puede tener más de un hotel.

NoSePuedeConstruirUnHotelEnUnBarrioSimpleException: Esta excepción lanza una señal cuando el jugador intenta comprar un hotel en un barrio en el que no está permitido comprar un hotel.

JugadorPresoNoSePuedeMoverException: Esta excepción lanza una señal cuando el jugador intenta mover cuando está en la cárcel.

SinPropietarioException: Esta excepción lanza una señal cuando se intenta obtener el propietario de un casillero apropiable que no tiene.

JugadorDebeComprarElBarrioParaPoderConstruirException: Esta excepcion lanza una señal cuando el jugador intenta construir pero no es propietario de la propiedad.

JugadorNoPuedeConstruirCasaSiNoAdquiereLosDosBarriosException: Esta excepcion lanza una señal cuando el jugador intenta construir en un barrio dividido pero no es propietario de ambas partes de la region.

JugadorNoPuedeConstruirHotelSiNoSeConstruyeElMaximoNumeroDeCasasException: Esta excepcion lanza una señal cuando el jugador intenta construir un hotel pero todavia no construyo todas las casas necesarias.

NoSePuedeComprarUnaCompaniaYaCompradaException: Esta excepcion lanza una señal cuando el jugador intenta comprar una compania que ya esta comprada y/o tiene propietario.

NoSePuedeComprarUnBarrioYaCompradoException: Esta excepcion lanza una señal cuando el jugador intenta comprar un barrio que ya esta comprado y/o tiene propietario.