FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Java

[7507/9502] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre de 2017

Alumno:	FERNANDEZ, Olivia
Número de padrón:	99732
Alumno:	LOIS, Lucas
Número de padrón:	98923
Alumno:	MARIJUAN, Magali
Número de padrón:	100070
Alumno:	MENDOZA, Kevin
Número de padrón:	98038

Índice

1. Introducción **2**

2. Supuestos **2**

3. Modelo de dominio **2**

4. Diagramas de clase **3**

5. Detalles de implementación **6**
5.1. Tablero 6
5.2. Dados 7
5.3. BarrioSimple y BarrioDoble 7
5.4. Controlador de Propiedades 7
5.5. Carcel y Jugador 7
5.6. Interfaz Gráfica 7

6. Excepciones **8**

7. Diagramas de secuencia **9**

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, .ªlgopoly", en Java utilizando los conceptos del paradigma de la orientación a objetos , los conceptos de técnicas de programación (patrones de diseño), los conceptos de integración continua y, por último, el desarrollo de una interfaz gráfica.

2. Supuestos

- Cuando se vende una propiedad que tiene un hotel se tiene en cuenta el dinero que se utilizó para construir las casas.
- Los servicios, para cobrarle al jugador, considera los dados que tiene al llegar al casillero y no se vuelven a tirar los dados.

3. Modelo de dominio

El trabajo práctico consta de dos partes importantes, el modelo y la vista. Esta última corresponde a la interfaz gráfica que lo dejarmos para la sección siguiente: "Detalles de Implementación".

Lo que concierne al modelo hemos decidido detallar en esta sección las siguientes clases:

- Jugador: esta es la clase principal del juego, mayor parte de su comportamiento lo delega. Es el encargado de moverse en el tablero y avanza acorde a lo que le dicen los dados o algún casillero en específico. Tambien puede comprar barrios o servicios.
- Tablero: el tablero se encarga de conectar los casilleros, es decir, marcar las referencias de quién está detrás y quién le sigue a cada casillero. Conoce directamente a la carcel para que la policía pueda encarcelar a los jugadores que lo merecen y conoce la salida para que comience el juego.
- Dados: devuelve la suma de dos numeros random del 1 al 6 para que el jugador pueda avanzar esa cantidad de casilleros.
- Casillero: es cada pequeña porción del tablero, puede ser comprable (edificable o no edificable) o puede ser un algún casillero especial. En ellos, en los casilleros, el jugador se ve afectado para bien o para mal. Lo más general, de los casilleros, es que pueden ser "pisados"por el jugador.
  - Salida: es el primer casillero del tablero. Allí se ubican al principio los jugadores y se elige de manera random el orden.
  - Policía: este casillero se encarga de mandar a la carcel a los jugadores que lo pisen.
  - Carcel: este casillero retiene a los jugadores, que lo pisen o que pisen la policía, por tres turnos. Tambien negocia con los jugadores una fianza.
  - Quini6: este casillero premia a los jugadores que lo pisen (sólo las dos primeras veces).
  - AvanceDinamico: este casillero mueve hacia adelante a los jugadores una cantidad de casilleros deteminada por el número que obtuvieron con los dados y el dinero que poseen.
  - RetrocesoDinamico: este casillero mueve hacia atrás a los jugadores una cantidad de casilleros determinada por el número que obtuvieron con los dados y las propiedades que poseen.
  - Barrio: Este casillero representa un terreno en el que el jugador puede cobrar rentas. Estas rentas aumentan si el jugador edifica casas u hoteles.

- Barrio Simple: Este casillero representa un único terreno en el que sólo se puede construír una casa.
- Barrio Doble: Este casillero representa un único terreno pero impone que para la edificación su casillero asociado deba pertenecerle al mismo jugador.

4. Diagramas de clase

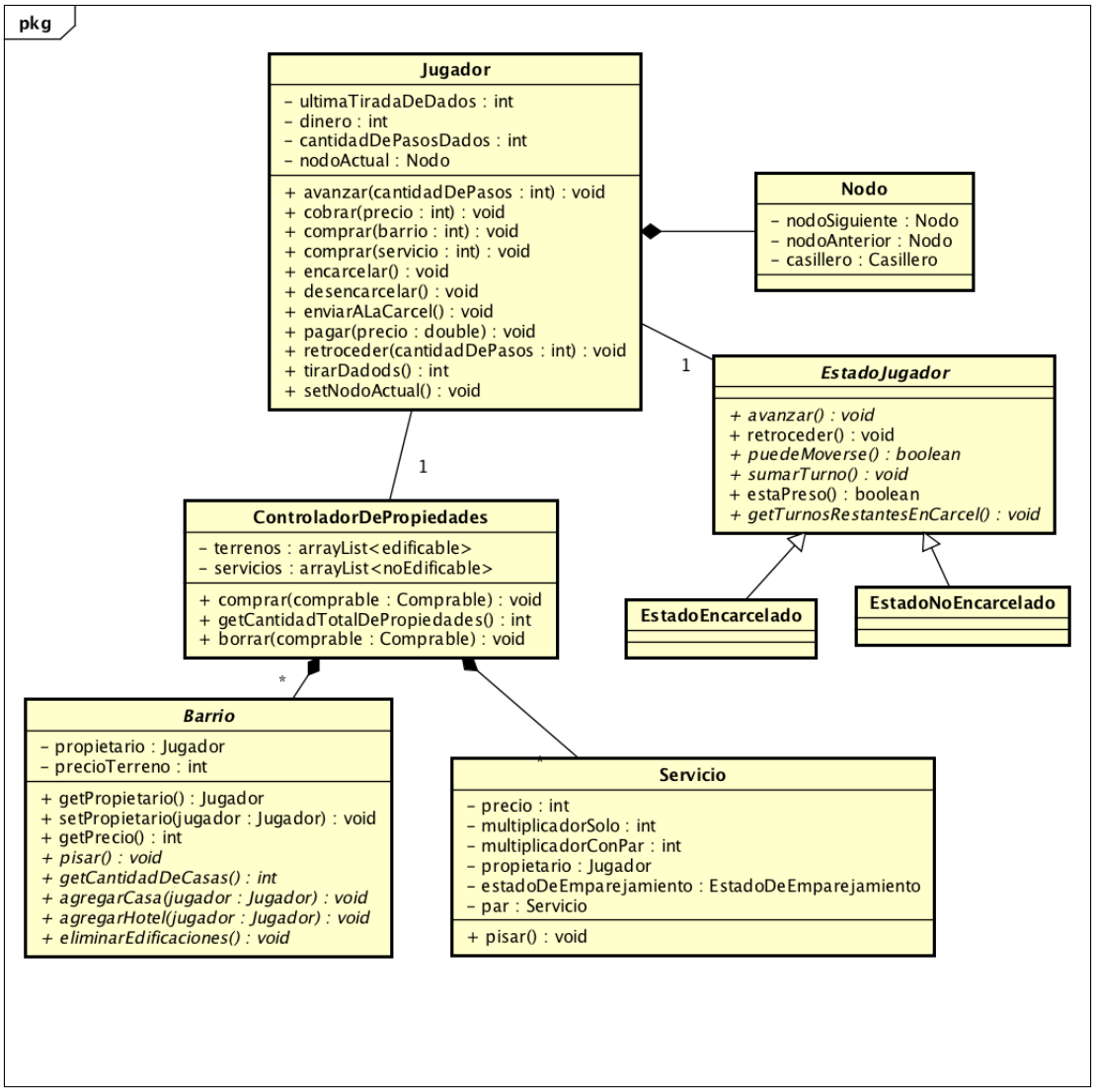

Figura 1: Diagrama de clases del jugador.

El jugador conoce el nodo en el que está, puede tomar dos estados a lo largo del juego (encarcelado y no encarcelado). Luego posee un controlador de propiedades que lo que hace es organizar sus propiedades e interacutar con ellas para la compra y la venta.

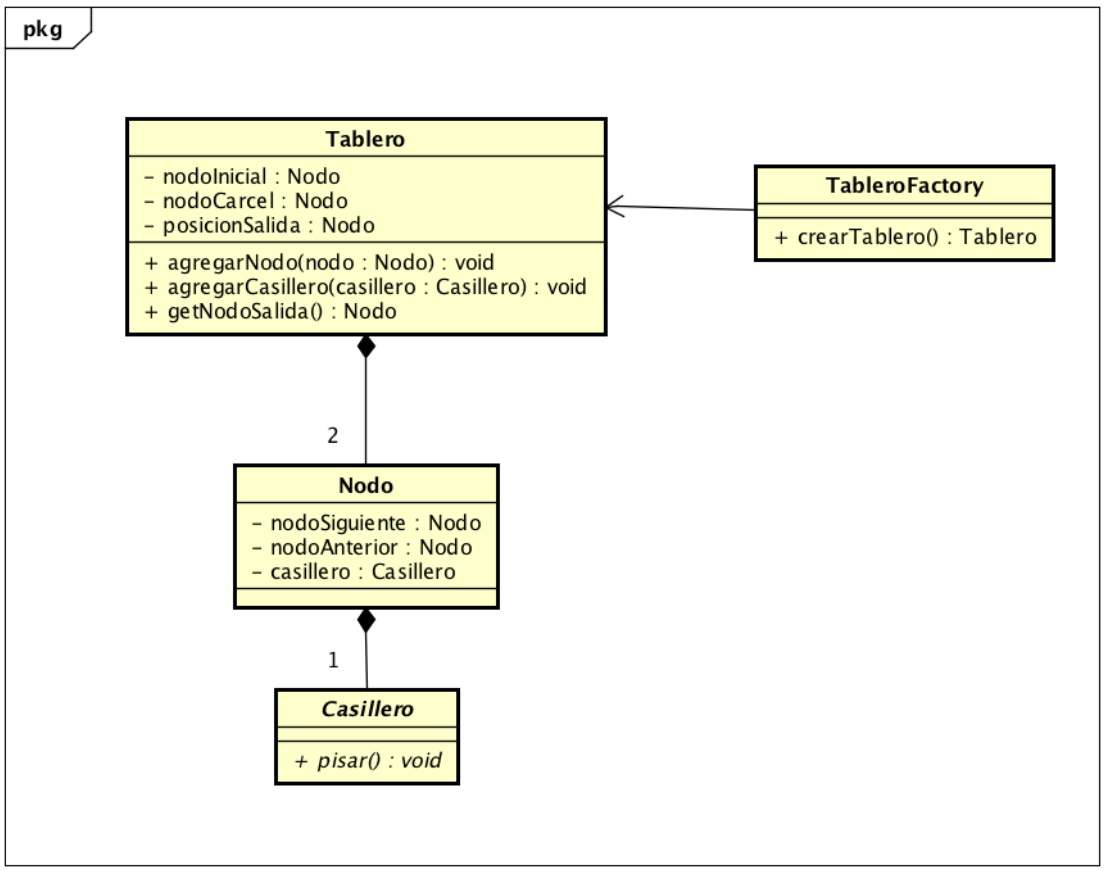

Figura 2: Diagrama de clases del tablero y los nodos.

Para el tablero utilizamos un patrón creacional(factory) que en este caso está representado por TableroFactory (en la sección siguiente se detallara el uso de este patrón). El tablero contiene 2 nodos unicamente, carcel e inicial. Los nodos a su vez estan compuestos por casilleros.

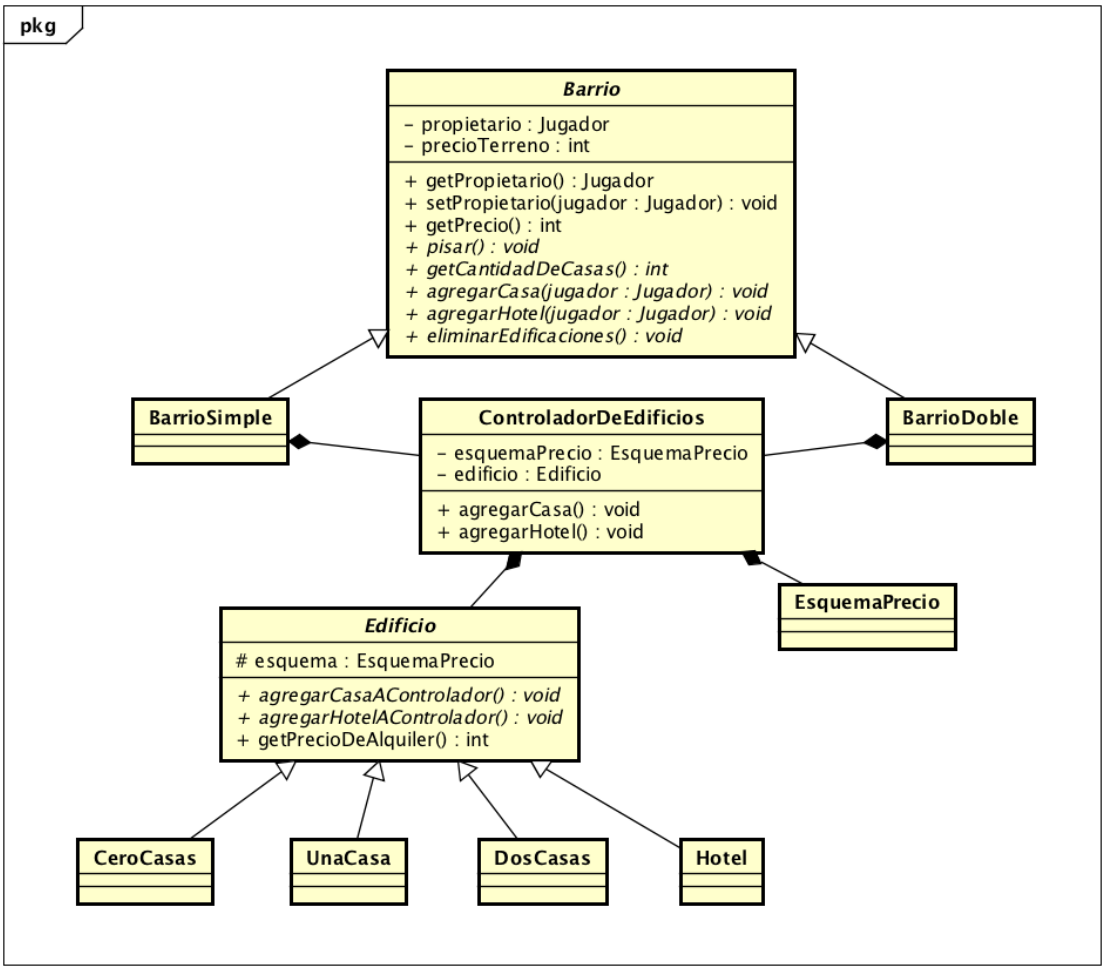

Figura 3: Diagrama de clases del Barrio.

Barrio es una clase abstracta de la cual heredan BarrioSimple y Barrio Doble, ambas se caracterizan por tener un controlador de edificios que se encarga principalmente de conocer un atributo del tipo Edificio que cumple con dos patrones de diseño que son: **Double Dispatch pattern** y **Null pattern**. De esta manera barrio puede pedirle a su controlador de edificios que agregue casas o las quite y éste sabrá que hacer.

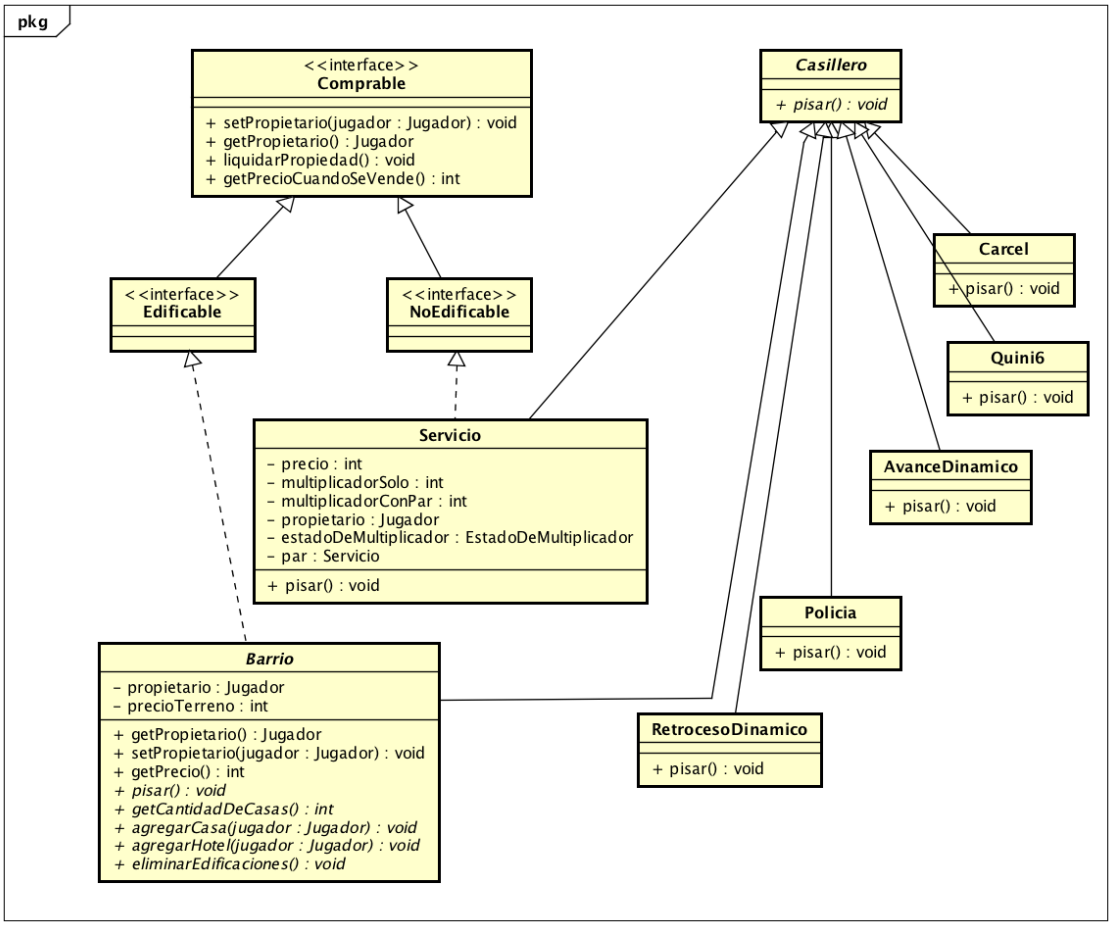

Figura 4: Diagrama de clases de Casillero.

Este diagrama busca mostrar la relacion de clases abstractas e interfaces que intervienen en los casilleros. Ya que, no todos los casilleros tienen el mismo comportamiento cuando son pisados.

En particular, todos los casilleros finales heredan de Casillero y redefinen pisar. Luego existe una interfaz comprable (que tal como puede comprarse puede venderse). Los únicos dos casilleros que implementan esta interfaz son Barrio y Servicio.

Para obligar a Barrio edificar hago que implente la interfaz edificable y como Servicio no puede edificar hago que implemente la interfaz NoEdificable y ambas interfaces heredan de Comprable para evitar reescribir código de lo que es igual.

5. Detalles de implementación

5.1. Tablero

Para la implementación del tablero necesitabamos que sea circular para que el jugador pueda dar vueltas alrededor de él; lo implementamos con una lista circular doblemente enlazada para que puede ir para atras y para adelante sin ningun problema donde el dato del nodo es el casillero. Ésto es importante ya que significa que el casillero no tiene por qué saber que tiene adelante y atras sino que se lo pone en un gran nodo que sí sabe.

Para la creacion del tablero utilizamos un TableroFactory que crea el tablero y agrega los casilleros en orden y con los datos correspondientes para poder arrancar a jugar.Esto es un **patrón**

creacional.

El jugador sabe en que tablero está y tambien en que nodo actual se encuentra; de ésta manera es el Jugador quien avanza atravez del tablero la cantidad de pasos que sean necesarios.

5.2. Dados

Los dados fueron implementados con un Singleton para que pueadan ser accesible por cualquier casillero y por cualquier jugador en cualquier momento del juego. Es en plural porque hace referencia a los 2 dados que se tiran en simultáneo.

5.3. BarrioSimple y BarrioDoble

Apesar de que ambos son Barrios, y cada uno representa un único casillero, les dimos nombres y, por lo tanto, clases distintas por diferir en el comportamiento, y los hicimos heredar, de su clase madre Barrio, en aquello en lo que actuaban igual. Las dos diferencias mas grandes son: el BarrioDoble tiene una referencia a su "par"para saber si el propietario es dueño de ambos terrenos (en BarrioSimple no existe ese "par"); por otro lado en BarrioSimple solo se pueden construir una casa por terreno a diferencia del BarrioDoble que se puede dos casa y despues un hotel.

5.4. Controlador de Propiedades

Esta clase es de las más importantes para explicar el comportamiento de Barrio. Lo primero que pensamos fue cómo implementar el método agregar casa y agregar hotel siendo tan distintas las condiciones que se pedían para la contrucción de cada una de ellas en cada tipo de barrio. Decidimos solucionar esta situación con el patrón **Double Dispatch**.

Para ello, existe la clase abstracta Edificio de la cual heredan CeroCasas, UnaCasa, DosCasas y Hotel. El controlador de edificios comienza con CeroCasas y cuando algún jugador pisa el barrio el controlador de edificios le pide a su edificio el precio de alquiler (en este caso contiene a CeroCasas que le cobra el alquiler por sólo pisar). Cuando se quiere comprar una casa, el controlador de edificios le pide a edificio (CeroCasas) que le agregue a él mismo UnaCasa. Y esto mismo sucede con los demás edificios. Si el controlador de edificios tiene CeroCasas y recibe el mensaje agregarHotel, el controlador de edificios le manda a CeroCasas agregarHotel en cotrolador y este lanza una excepción.

Por otro lado, el objeto CeroCasas que también podría llamarse NoCasas, implementa el patrón **NullPattern**. Este patrón lo que hace es representar un valor neutral, es decir, cuando se compra un barrio vacío.

5.5. Carcel y Jugador

Para implementar la carcel y poder retener al jugador por 3 turnos utilizamos un **patron state** en Jugador que dependiendo del patrón el jugador, puede o no, moverse. Inicialmente está en EstadoNoEncarcelado por lo tanto puede moverse pero una vez que cae en la carcel, ésta le manda el mensaje mandarALaCarcel que cambia el estado del jugador a EstadoEncarcelado y, por otro lado, se inicializa un contador de turnos para poder despues volver al EstadoNoEncarcelado (amenos que pague la fianza antes).

5.6. Interfaz Gráfica

Esta se compone de dos escenas, inicio y main. En inicio se le piden los datos a los jugadores y en main se desarrolla el juego.

La mainScene contiene al campo juego, al contenedor principal y la información del jugador.

El contenedor principal hereda de BorderPane, en el cual se inserta el campo juego y el player information.

El campo juego hereda de GridPane, que nos permite realizar una grilla en la cual colocamos las vistas de los casilleros. De esta manera cada casillero posee una posición única. Luego, procedimos por realizar una función que itera de manera circular por esta grilla permitiendo mover las fichas.

Por otro lado, decidimos que el tablero que contiene las vistas y una clase llamada AlgoPoly sean singelton, de esta manera, logramos obtener una instancia de ellas en cualquier parte del juego y logramos el acceso de los datos.

AlgoPoly contine la listaUsuarios y establece una relación entre el jugador y su usuario mediante un hash y establece una relación entre el nodo(el del modelo) y su vista con otro hash. Esto es, cada nodo posee su propia vista que personaliza los botones y las actividades que puede realizar el usuario.

El usuario, es mucho más que el jugador del modelo, este contiene al jugador, su ficha y su posición en el tablero gráfico. La relación antes mencionada del nodo con su vista, sirve para pedirle la vista que posee tal nodo, entonces, por ejemplo, cuando el jugador se mueve de la policía a la carcel su ficha puede moverse, sino sólo avanzaría lo que le dicen los dados.

Exite también, CajasVistas, que lo que hacen es mostrar las distintas opciones que tiene el usuario en cada casillero como también mostrar la información del mismo casillero. Estas casas llaman al método pisar del casillero y atrapan excepciones que puedan surgir, perminitiendo desarrollar el juego.

6. Excepciones

PropiedadNoPuedeConstruirHotelException Ésta excepción fue creada para el caso que se desee construir hotel en un tipo de propiedad que no admite la construcción de hoteles, tanto como un Servicio como por ejemplo un BarrioSimple que sólo permite la construcción de una sóla casa.

CantidadInsuficienteDeCasasExcepcion Contempla el caso en el que se quiera construir un hotel en un BarrioDoble pero todavía no se tenga el maximo de casas posibles (2 en cada barrio).

JugadorNoEsDuenioDeAmbasPropiedadesExcepcion En el caso que el jugador no tenga las dos propiedades del BarrioDoble y desee construir una casa en el que sí tiene.

JugadorNoEsDuenioDePropiedadExcepcion Este caso es el mismo que el anterior solo que sin tener ninguna de las propiedades (sirve para los BarrioSimple más que nada).

JugadorNoPuedeMoverseException En el caso que el jugador este encarcelado y desee avanzar se lanza ésta excepción.

JugadorNoTieneDineroException Tanto para cuando quiera comprar una propiedad o cuando pisa un casillero que requiere gasto y no tiene el dinero suficiente se lanza ésta excepción.

PrecioNegativoException Si se settea una propiedad con un precio negativo se lanza la excepción.

PropiedadYaTieneDuenioException Cuando se desea comprar una propiedad que ya tiene dueño.

7. Diagramas de secuencia

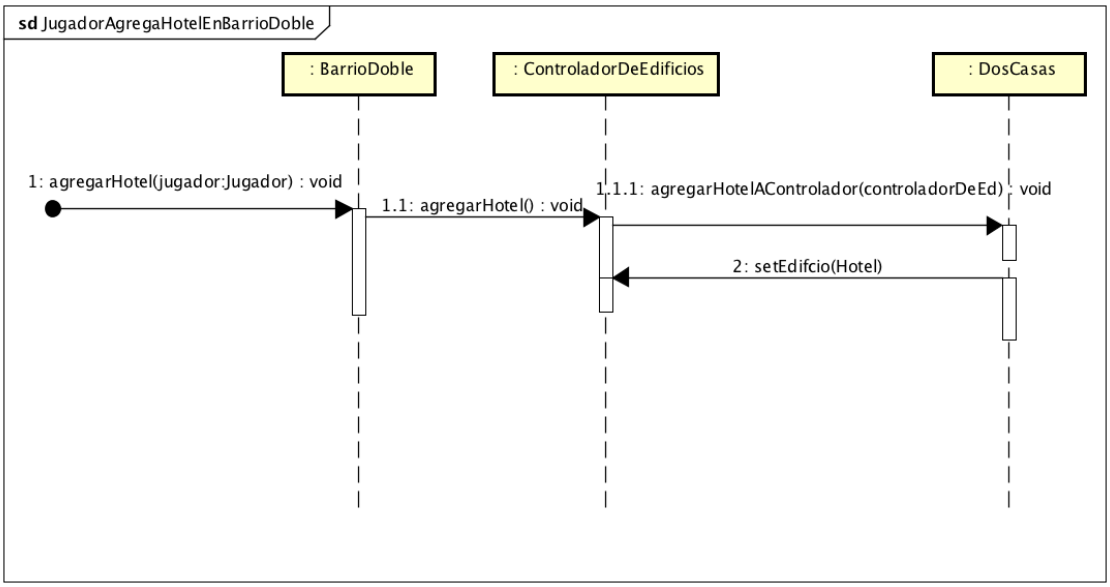

Figura 5: Jugador agrega hotel en barrio doble.

Este diagrama muestra cuando el jugador desea agregar un hotel en un BarrioDoble, este le delega este comportamiento al controlador de edificios. Este último, le pide a su Edificio que le agregue un hotel.

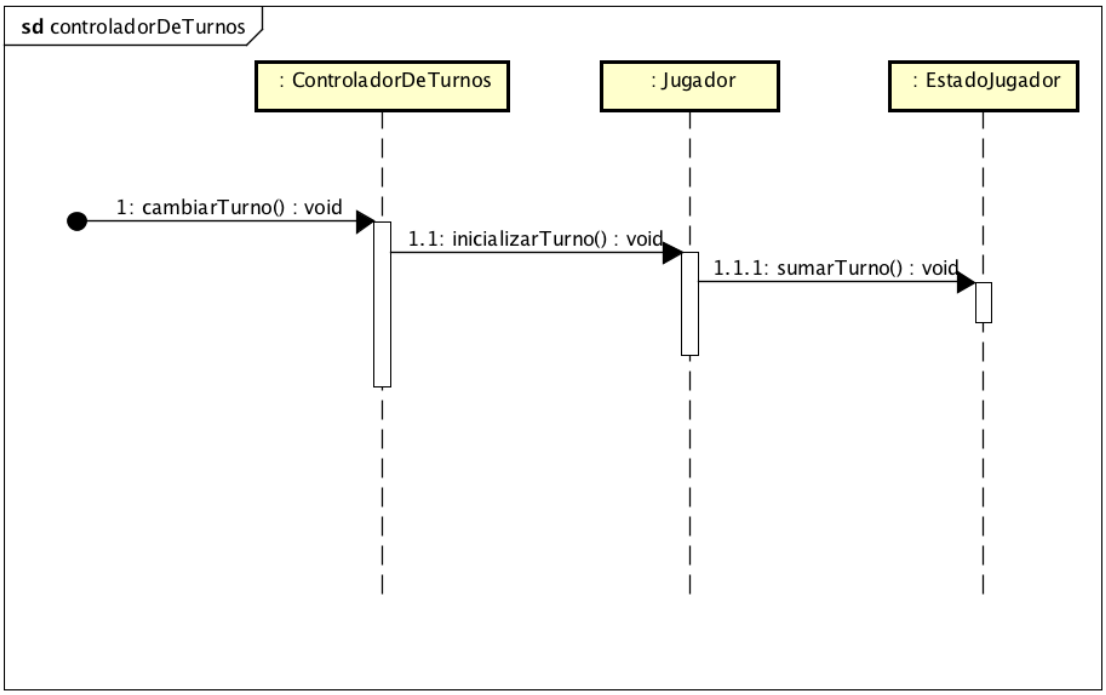

Figura 6: ControladorDeTurnos.

Este diagrama muestra lo que sucede cuando se cambia el turno. El controlador de turnos le pide al jugador que inicialice su turno y este a su le dice al estado que su sume un turno. Esta última actividad tiene sentido en el EstadoEncarcelado ya que le resta un turno pendiente en la carcel.

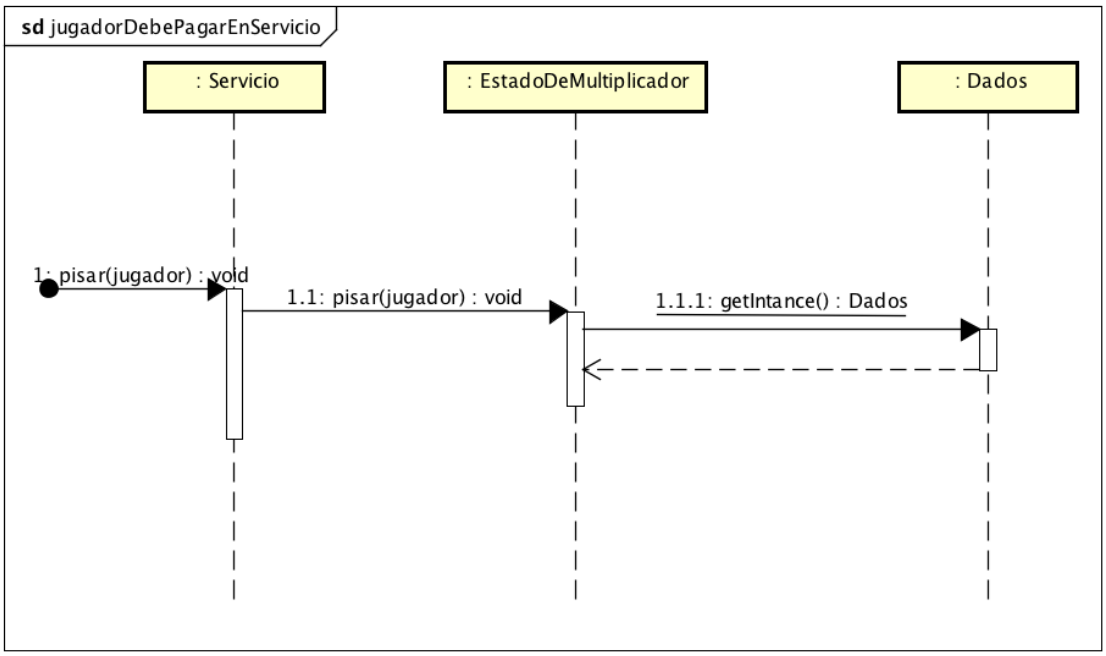

Figura 7: Jugador debe pagar en servicio lo que sale de los dados.

Este diagrama muestra cuando un jugador pisa un servicio. Este último llama a su EstadoMultiplicador (que posee el porcentaje que se utilizará) y le pide a la clase Dados su única instancia y a esta le pide el último valor obtenido.

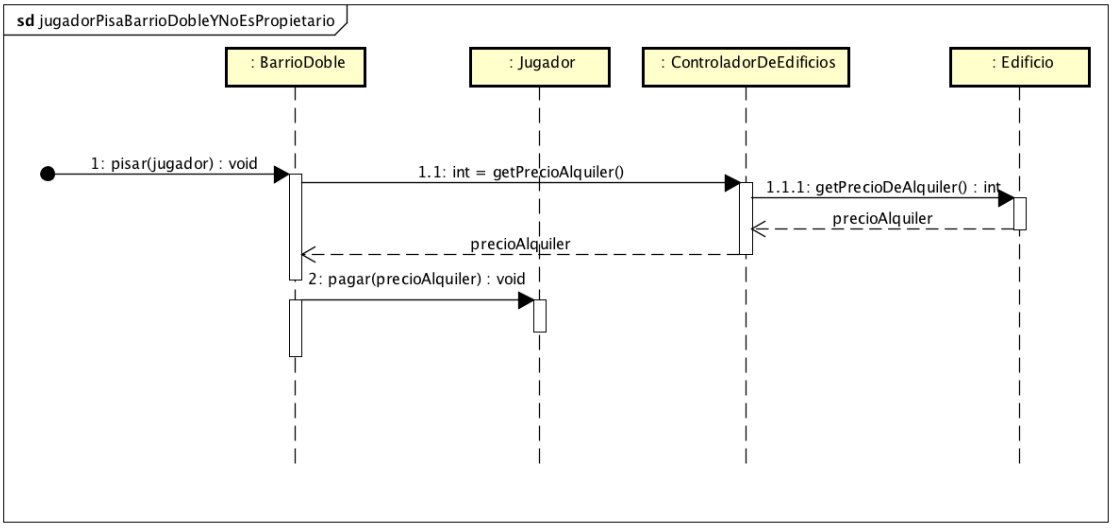

Figura 8: Jugador pisa BarrioDoble y no es propietario.

Este diagrama muestra cuando un jugador que no es propietario pisa un Barrio. Este le pide al controlador de turno el precio del alquiler que a su vez, el controlador delega este comportamiento al edificio. Luego el barrio doble le dice al jugador que pague el alquiler.

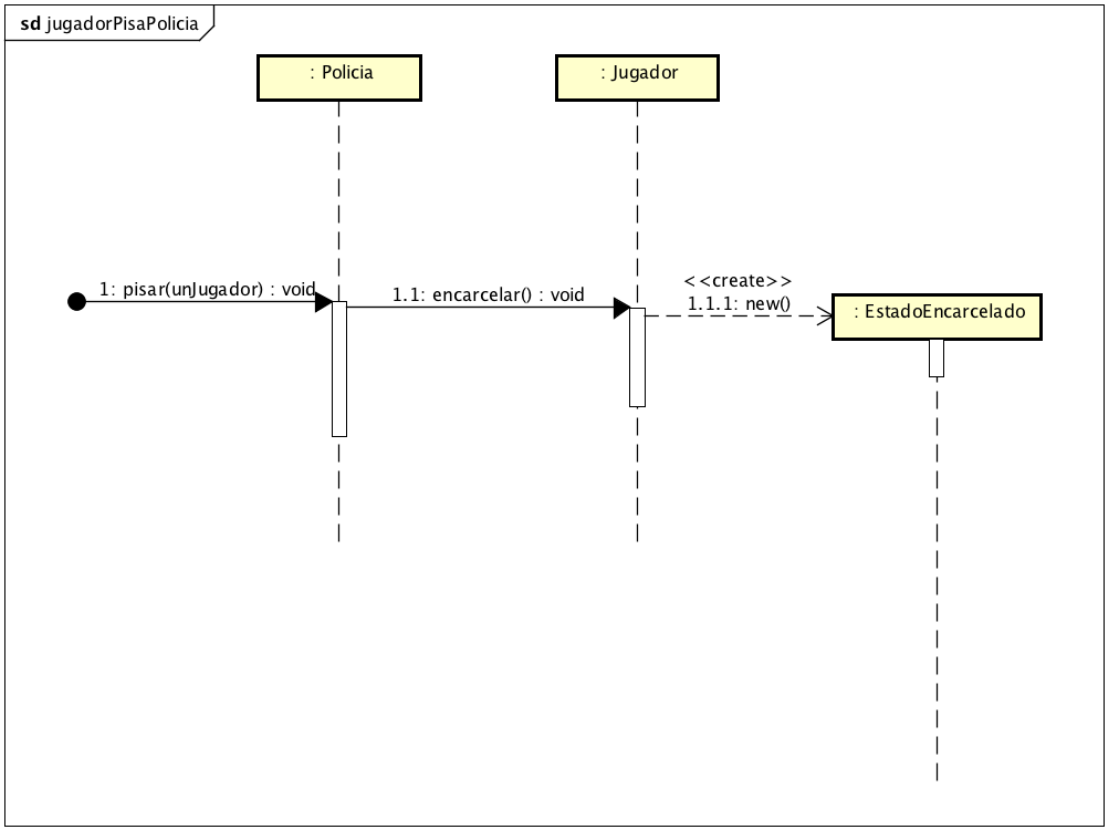

Figura 9: Jugador pisa policía y va a la carcel.

Este diagrama muestra cuando un jugador pisa la Policía y esta lo encarcela. EL método encarcelar del jugador crea una nueva instancia de EstadoEncarcelado y lo guarda como atributo.