**FIUBA - 75.07**

**Algoritmos y programación III**

*Trabajo práctico 2: AlgoPoly*

2do cuatrimestre, 2017

(trabajo grupal de 4 integrantes)

Alumnos:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Padrón** | **Mail** |
| Dan Tchechenistky | 100117 | dtcheche@gmail.com |
| Andres Visciglio | 100118 | atvisciglio@gmail.com |
| Tomás Sabao | 99437 | tomalon95@gmail.com |
| Luciano Ortiz | 100323 | lucianortiz97@gmail.com |

***Fecha de entrega final***: 30 de Noviembre de 2017

***Tutor***: Pablo Suárez, Matias Leal Bazterrica

***Comentarios***:

## 

## Modelo de dominio:

El programa está compuesto por total de X clases, con el fin de modelar el juego Monopoly. El siguiente listado menciona y detalla la responsabilidad de cada una de ellas.

* Carcel: Encargada de modificar el movimiento del jugador y de cobrar la fianza.
* Quini6: Clase cuya responsabilidad es premiar al jugador dependiendo de sus recompensas anteriores.
* Policia: Su responsabilidad es enviar al Jugador a la Carcel.
* Caible: Clase abstracta cuyo fin es agrupar Carcel, Quini6, Policia, entre otros y unificar la recepcion de un determinado mensaje, en este caso accionar( ).
* Movimiento Normal Y Carcel encapsulan la movilidad del jugador, de esta forma no es necesario que el jugador conozca la movilidad dentro del tablero.
* RetrocesoDinamico: Su responsabilidad es retroceder una cantidad determinada de casilleros al jugador que caiga en él, dependiendo de lo que saco en sus dados.
* AvanceDinamico: Su responsabilidad es avanzar una cantidad determinada de casilleros al jugador que caiga en él, dependiendo de lo que saco en sus dados.

## Detalles de implementación:

### Casilleros:

Al momento de plantear la implementación de los casilleros surgió la idea de hacer una clase abstracta Caible que tenga como atributo una posición en el tablero y un método que la accione. De esta clase heredarian entonces los diferentes tipos de casilleros. Por un lado estarían los casilleros que el jugador puede comprar, estos se representan con una clase abstracta Propiedad. Esta contiene los atributos generales a todas las propiedades. Y de ella heredan Barrio y Compañía. Por otro lado se encuentran los casilleros que no pueden ser comprados por el jugador como lo son por ejemplo Carcel, Policia, Avance Dinámico… Todos estos son representados por clases individuales ya que su lógica es independiente.

### Estados de propiedad

Para determinar el comportamiento de las propiedades que poseen o no un dueño, decidimos aplicar el state pattern. Se creó una clase abstracta “EstadoDuenio” de la cual se producen otras dos clases “ConDuenio” y “SinDuenio”. Dado que el comportamiento de las propiedades se encuentra determinado por si tienen o no un propietario, haciendo que al caer los jugadores se mande un mensaje directamente al estado de la propiedad nos ahorra la utilización de numerosos if, haciendo que el código sea más claro y legible, y al mismo tiempo nos otorga más flexibilidad ante cambios en los requisitos propuestos.

### Quini 6

Para la implementación del mecanismo de ganar el quini6, decidimos emplear una lista enlazada en la cual se encuentran almacenados los distintos estados de ganancia monetaria establecidos en el diagrama. Usando un iterador para señalar cuál es el siguiente premio que se obtendrá al caer nuevamente en el casillero, al momento de caer se obtiene el premio monetario y se avanza al siguiente estado. Al final de la lista, el último premio que es de 0 pesos tiene como referencia a un próximo estado a sí mismo.



La misma se encuentra almacenada en el jugador que cae en el quini 6. Consideramos hacerlo de esta manera para no atentar contra la extensibilidad del código. Si hubiéramos querido que sea el propio quini quien guarde las listas de premios (una para cada jugador), si tuviéramos de repente un cambio en las especificaciones y el número de jugadores que el programa acepta son 4, tendríamos una mayor cantidad de código para reescribir.

### Avance y retroceso dinamico:

El avance y retroceso dinamico son muy similares. Al caer un jugador en este casillero se determina cuantas posiciones debera avanzar/retroceder mediante if’s, luego cuentan con un método que hará respectivamente avanzar/retroceder esa cantidad de casilleros al jugador.

## Excepciones:

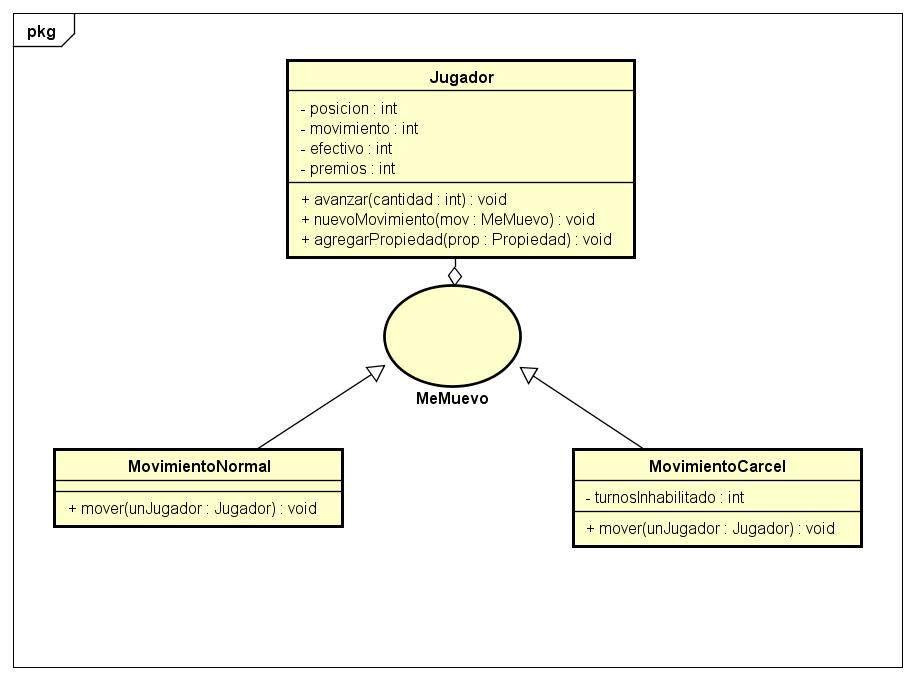
Se implementaron dos excepciones para cubrir posibles errores; MontoInsuficienteException y FianzaInhabilitadaException.

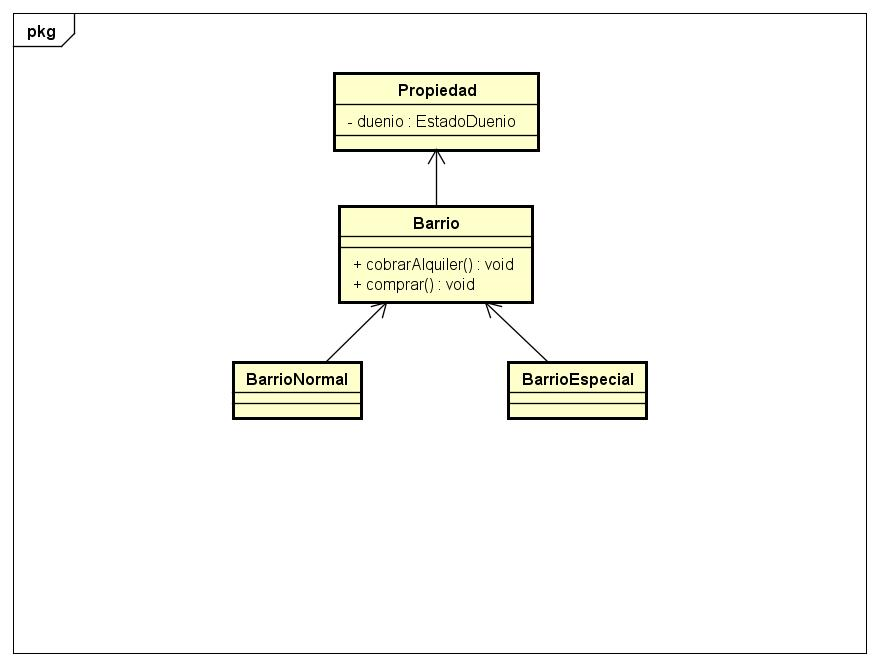
Como es de esperar en el desarrollo del juego, existe la posibilidad de que el jugador no tenga el dinero suficiente para pagar lo que él desea. Es por eso que MontoInsuficienteException aparece en esos casos, para evitar que se realicen operaciones invalidas y perjudiquen la integridad del juego.

Por otro lado tenemos FianzaInhabilitadaException, la cual surge en los casos en que el jugador intenta pagar la fianza no habiendo dejado pasar un turno obligatorio en la Carcel. De esta forma se respetan las reglas establecidas por el juego.

## Diagramas de clase

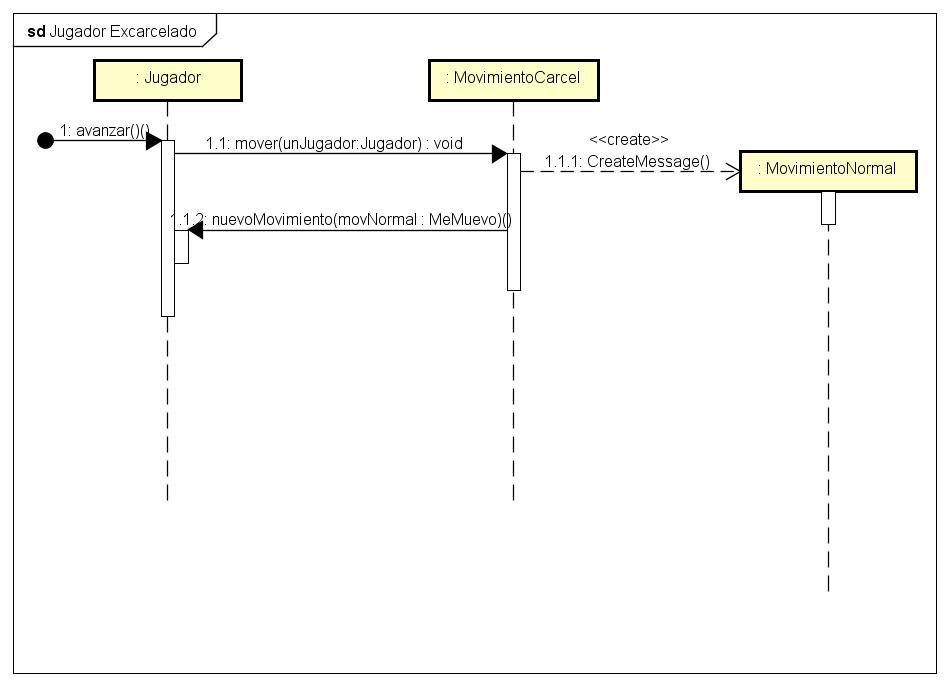
Diagrama que muestra el movimiento del jugador a traves de una interfaz *MeMuevo*.



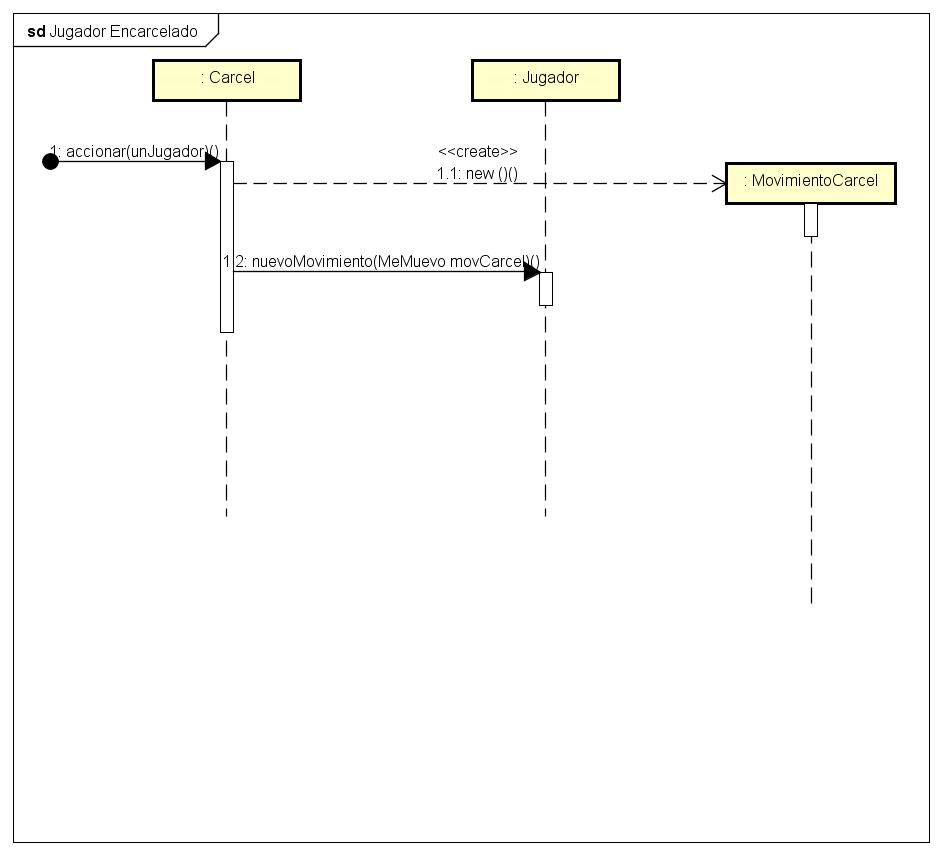
Jerarquia del esquema Barrio - Propiedad

## Diagrama de secuencia

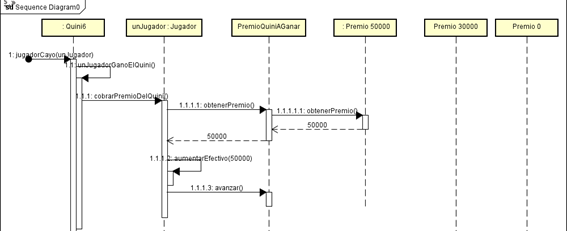
Luego de 4 turnos en la Cárcel, el jugador recupera su movilidad en el juego

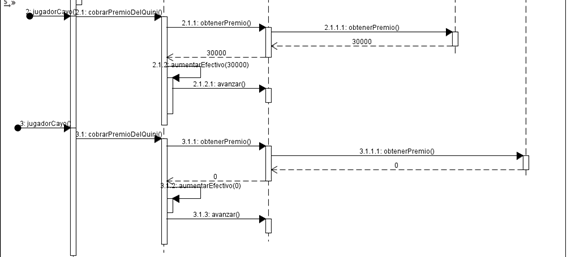


El Jugador es encarcelado y no puede avanzar



Jugador cae en quini6 3 veces





Jugador cae en una propiedad sin dueño y la compra. Él pasa a ser el dueño

