FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# Trabajo práctico 2: AlgoPoly

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


| Alumno: | Número de Padrón: | Email: |
|---|---|---|
| Lucas Etcheverry | 94427 | lucas.etcheverry28@gmail.com |
| Diego Esposito | 95669 | diego92_79@hotmail.com |
| Guido Di Paolo | 95346 | guido.dipaolo@gmail.com |

# Índice

# 1. Introducción

## 1.1. Objetivos

Aplicar los conceptos enseñados en la materia a la resolución de un problema, trabajando en forma grupal y utilizando un lenguaje de tipado estático (Java)

## 1.2. Consigna General

Desarrollar la aplicación completa, incluyendo el modelo de clases e interfaz gráfica. La aplicación deberá ser acompañada por pruebas unitarias e integrales y documentación de diseño. En la siguiente sección se describe la aplicación a desarrollar.

## 1.3. Descripción de la aplicación a desarrollar

Se deberá desarrollar una aplicación que implemente un juego relacionado con el clásico juego de mesa MonoPoly. En el juego habrá 3 jugadores. Cada jugador comenzará con la misma cantidad de dinero equivalente a $100000 (cien mil pesos) desde la posición de salida. El tablero tendrá la siguiente disposición:

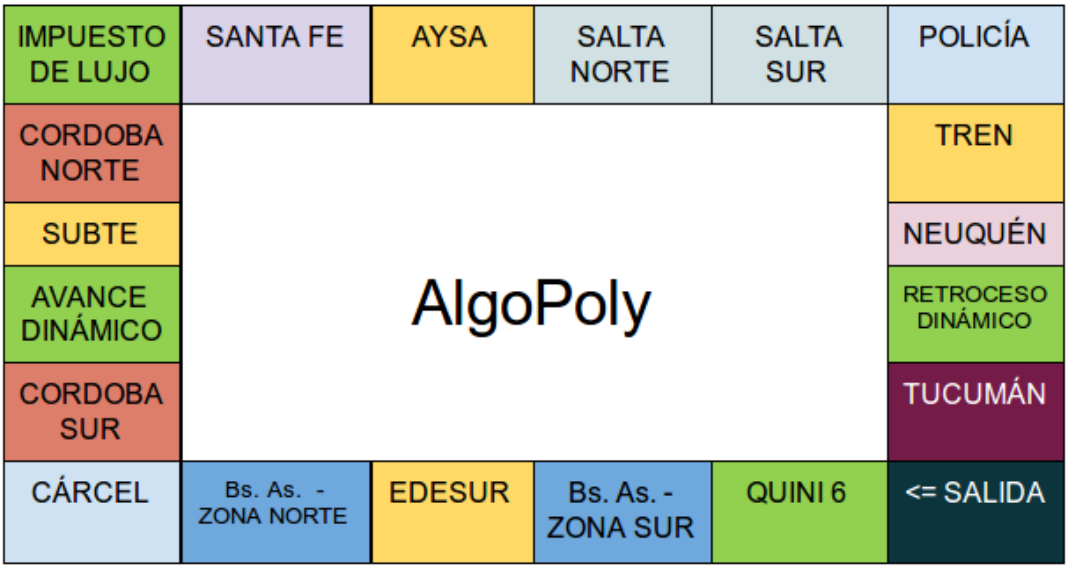


Figura 1: Tablero de la aplicación.

### 1.3.1. Casilleros

1. **Salida**: Los 3 jugadores comienzan el juego en el mismo lugar. Se elige aleatoriamente quién tira primero, segundo y tercero.

2. **Quini 6**: El jugador que caiga en esta casilla recibe un premio de $50000 (cincuenta mil pesos). Si ese jugador ya ganó una vez el Quini 6, la segunda vez que caiga en esa casilla recibe un premio de $30000 (treinta mil pesos). Las sucesivas veces que el mismo jugador caiga en este casillero no cobrará ningún dinero.

3. **Buenos Aires Sur:**:

   - Precio terreno: $20000 (veinte mil pesos)
   - Alquiler: $2000
   - Alquiler con 1 casa: $3000

- Alquiler con 2 casas: $3500
- Alquiler con Hotel: $5000
- Construir casas cuestan $5000 y hotel $8000

4. **EDESUR**:
   - Comprar la compañía: $35000
   - Cobra 500 veces lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados =>deberá pagar: 12 x 500 = $6000
   - Si también tiene la compañía AYSA cobrará 1000 veces lo sacado en los dados.
5. **Buenos Aires Norte:**:
   - Precio terreno: $25000 (veinte mil pesos)
   - Alquiler: $2500
   - Alquiler con 1 casa: $3500
   - Alquiler con 2 casas: $4000
   - Alquiler con Hotel: $6000
   - Construir casas cuestan $5500 y hotel $9000
6. **Cárcel:**: un jugador al caer en este casillero debe esperar 3 turnos para salir. Es decir, recién cuando le toque por 4ta vez podrá moverse. Salvo que cuando cuando ya haya pasado 1 turno de la cárcel (o sea en el turno 2 y 3 de espera) pague una fianza de $45000.
7. **Córdoba Sur:**:
   - Precio terreno: $18000
   - Alquiler: $1000
   - Alquiler con 1 casa: $1500
   - Alquiler con 2 casas: $2500
   - Alquiler con Hotel: $3000
   - Construir casas cuestan $2000 y hotel $3000
8. **Avance dinámico:**El jugador avanzará tantos casilleros como lo indica la siguiente lógica:
   - Si sacó 2,3,4,5 o 6 =>entonces avanza el número sacado menos 2 unidades
   - Si sacó 7,8,9 o 10 =>entonces avanza #cantidad de efectivo del jugador
   - si sacó 11 o 12 =>avanza el numero sacado menos la sumatoria de propiedades del jugador. Tanto los terrenos como las casas como los hoteles suman como propiedad.
9. **SUBTE:**
   - Comprar la compañía: $40000
   - Cobra 600 veces lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados =>deberá pagar: 12 x 600 = $7200
   - Si también tiene la compañía TRENES cobrará 1100 veces lo sacado en los dados.
10. **Córdoba Norte:**
    - Precio terreno: $20000

- Alquiler: $1300
- Alquiler con 1 casa: $1800
- Alquiler con 2 casas: $2900
- Alquiler con Hotel: $3500
- Construir casas cuestan $2200 y hotel $3500

11. **Impuesto al Lujo:** El jugador que caiga en esta casilla debe pagar el 10

12. **Santa Fe:**

- Precio terreno: $15000
- Alquiler: $1500
- Alquiler con 1 casa: $3500
- Construir la única casa posible cuesta $4000

13. **AYSA:**

- Comprar la compañía: $30000
- Cobra 300 veces lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados =>deberá pagar: 12 x 300 = $3600
- Si también tiene la compañía EDESUR cobrará 500 veces lo sacado en los dados.

14. **Salta Norte:**

- Precio terreno: $23000
- Alquiler: $2000
- Alquiler con 1 casa: $3250
- Alquiler con 2 casas: $3850
- Alquiler con Hotel: $5500
- Construir casas cuestan $4500 y hotel $7500

15. **Salta Sur:**

- Precio terreno: $23000
- Alquiler: $2000
- Alquiler con 1 casa: $3250
- Alquiler con 2 casas: $3850
- Alquiler con Hotel: $5500
- Construir casas cuestan $4500 y hotel $7500

16. **Policía:** Al caer en este casillero el jugador va a la cárcel.

17. **TRENES:**

- comprar la compañía: $38000
- Cobra 450 veces lo que dice los dados. O sea que si un jugador no dueño de la empresa cae en esa casilla tras haber sacado 12 en los dados =>deberá pagar: 12 x 450 = $5400
- Si también tiene la compañía SUBTES cobrará 800 veces lo sacado en los dados.

18. **Neuquén:**

- Precio terreno: $17000
- Alquiler: $1800
- Alquiler con 1 casa: $3800
- Construir la única casa posible cuesta $4800

19. **Retroceso dinámico:** El jugador avanzará tantos casilleros como lo indica la siguiente lógica:
    - Si sacó 2,3,4,5 o 6 =>retrocede el número sacado menos la sumatoria de propiedades del jugador. Tanto los terrenos como las casas como los hoteles suman como propiedad.
    - Si sacó 7,8,9 o 10 =>retrocede #cantidad de efectivo del jugador
    - si sacó 11 o 12 =>retrocede el número sacado menos 2 unidades
20. **Tucumán:**
    - Precio terreno: $25000
    - Alquiler: $2500
    - Alquiler con 1 edificio histórico: $4500
    - Construir la casita de tucumán (única construcción posible) cuesta $7000

### 1.3.2. Jugabilidad

- Es un juego por turnos. Hay 3 jugadores. En cada turno cada jugador debe lanzar un par de dados (números del 1 al 6 cada dado). El número obtenido será la suma de ambos resultados.
- Si un jugador al tirar los dados obtiene un doble numero (o sea 1 y 1 , o 3 y 3 , etc) entonces tira nuevamente. Si vuelve a sacar otro doble, no tira otra vez, sino que le toca el turno al jugador siguiente.
- El jugador debe mover en el sentido que indica la flecha del casillero de salida.
- Los efectos ocurren al caer en un casillero y no **al pasar por ellos**.
- Para poder edificar tanto en Córdoba como en Bs.As como en Salta es necesario que el jugador compre ambas zonas (norte y sur) para recién poder edificar.
- No se puede edificar en ningún terreno en el mismo turno en que se lo compra Antes de lanzar los dados, cada jugador tiene la posibilidad de edificar si se encuentra en condiciones de hacerlo: Ya sea tanto contar con el dinero que sale + el terreno.
- Para poder edificar un hotel, debe estar lleno de casas en su capacidad máxima. Es decir que si un jugador tiene 2 casas en Bs. As. sur y sólo 1 casa en Bs. As Norte **NO** puede construir hotel. Recién al contar con 2 casas en Bs. As sur y 2 casas en Bs. norte puede empezar a construir los hoteles (máximo 1 por cada terreno).
- Los hoteles reemplazan a las casas, es decir que cuando se edifica un hotel las 2 casas desaparecen.
- Los terrenos que no son dobles (es decir que no tiene un Norte y Sur) no pueden construir hoteles.
- Un jugador puede decidir vender tanto sus terrenos (con las casas u hoteles que tenga edificados) o las compañías antes de lanzar los dados. En ese caso el jugador perderá la titularidad de los mismos, los cuales pasan a estar disponibles para la compra para el resto de los jugadores. El jugador que vende cobra un 15

### 1.3.3. Fin del juego

1. Cuando un jugador se quede sin dinero y sin propiedades quedará eliminado.
2. Si un jugador no tiene dinero, pero sí propiedades y debe afrontar un gasto, está obligado a vender (con la quita del 15
3. Ganará el último jugador restante.

### 1.3.4. Interfaz gráfica

Se debe desarrollar una interfaz visual para la interacción entre los jugadores. En la misma se pondrá mucho énfasis y se evaluará como parte de la consigna la **USABILIDAD** de la misma.

# 2. Supuestos

1. Solo se pueden vender propiedades, y en caso de ser barrios con casas u hoteles se venderan tambien con el terreno.
2. Si posee ambas propiedades de aquellos barrios dobles y construye un hotel en ambos. Al vender uno de los terrenos, sigue conservando el hotel en el otro que no vendió.
3. Si un jugador saca dobles en los dados por primera vez, realiza esa jugada y luego si puede tirar los dados nuevamente.
4. Si un jugador cae en el casillero Cárcel no ocurre nada. Puede jugar normalmente. Solo va a la cárcel cuando cae en el casillero "Policia".
5. Cuando su turno lo permite, un jugador puede comprar todas las casas que desee en diferentes barrios.

# 3. Modelo de dominio

A continuación se detallan las entidades que permiten modelar el juego AlgoPoly.

- **AlgoPoly** : Esta clase se encarga de mantener una relación con los jugadores de la partida y es responsable de hacerlos jugar.
- **Cell** : Abstracción que representa los casilleros del tablero
  - CellGroup : Esta clase se encarga de construir las celdas. Tomas valores definidos en un archivo de configuración.
  - Groupable (Interfaz) : Esta interfaz modela el comporamiento de aquellas celdas que conforman grupos, como los barrios dobles, o los servicios.
  - Owneable (Interfaz) : Esta interfaz modela el comportamiento de aquellas celdas que pueden ser adquiridas.
  - Start Point : Esta clase representa al casillero de salida
  - Quini 6 : Esta clase representa al casillero quini 6 y se encarga de conocer cuales fueron los ganadores durante la partida actual
  - Neighborhood : Esta clase representa a los barrios , Bs As - Sur , Cordoba - Norte, etc . La clase encapsula el efecto que posee cuando un jugador cae en ella.
  - NeighborhoodZone: Esta clase extiende de CellGroup y modela el comportamiento de las zonas conformadas por barrios dobles.
  - Jail : Esta clase representa al casillero Carcel y ademas se encarga de saber cuales son los jugadores que se encuentran presos
  - DynamicForward : Esta clase representa al casillero avance dinámico.
  - DynamicBackward : Esta clase representa al casillero retroceso dinámico.
  - Railway : Esta clase representa a los casilleros Subte y Tren. La clase encapsula el efecto que posee cuando un jugador cae en ella.
  - Service : Esta clase representa a los casilleros Edesur y Aysa . La clase encapsula el efecto que posee cuando un jugador cae en ella.
  - Policia : Esta clase representa a la casilla Policia y además encapsula la lógica que permite enviar a un jugador a la carcel cuando se cae en ella.

- **Board** : Esta clase mantiene la relación con todos los casilleros que conforman el tablero.
- **MotionAlgorithm** : Esta interfaz se encarga de definir un método que luego es implementado por ciertas clases que representan los distintos tipos de movimeintos que un jugador posee.
  - NormalForward : Representa el movimiento normal del jugador, es decir avanza tantos casilleros como haya sido el resultado de los dados.
  - StoppedInJail: Representa a un jugador que no se puede mover debido a que se encuentra en la cárcel.
  - StoppedInBankruptcy : Representa a un jugador que no se puede mover debido a que posee deuda.
  - DynamicForwardPlus2 : Encapsula el algoritmo utilizado para el avance dinámico cuando el jugador cae en la casilla avance dinámico sacando en los dados un resultado entre 2 y 6
  - DynamicBackwardPlus2 : Encapsula el algoritmo utilizado para el retroceso dinámico cuando el jugador cae en la casilla retroceso dinámico sacando en los dados un resultado entre 2 y 6
  - DynamicForwardCash : Encapsula el algoritmo para avance dinámico según el dinero que el jugador posea.
  - DynamicForwardProperties : Encapsula el algoritmo para avance dinámico según la cantidad de propiedades que el jugador posea.
  - DynamicBackward : Esta clase y todas sus similares encapsulan la lógica que le dice a un jugador en que forma debe retroceder cuando se cae en la casilla retroceso dinámico.
  - DynamicForwardAlgorithmFactory : Esta clase funciona como dispatcher, retorna el algoritmo requerido para DynamicForward segun el resultado de los dados.
  - DynamicBackwardAlgorithmFactory : Esta clase funciona como dispatcher, retorna el algoritmo requerido para DynamicBackward segun el resultado de los dados.
  - Defeted : Esta clase modela el movimiento de un jugador que ha sido derrotado.
- **Prisoner** : Esta clase representa a un jugador en la carcel.
- **Player** : Esta clase representa a un jugador.
- **Money** : Esta clase se encarga de representar la plata que se utiliza en el juego para la compra y venta de propiedades.
- **Die** : Esta clase se encarga de simular los dados que se utilizan en el juego.
- **Turn** : Esta clase se utiliza para mentener el contexto de cada turno del juego. Por ejemplo posee el jugador que esta moviendo en el turno actual.
- **Debt** : Esta clase se utiliza para modelar una deuda.

# 4. Diagramas

## 4.1. Diagramas de clase

### Celdas

A continuación se muestra un diagrama de clases relacionado con el modelado de las celdas del juego.

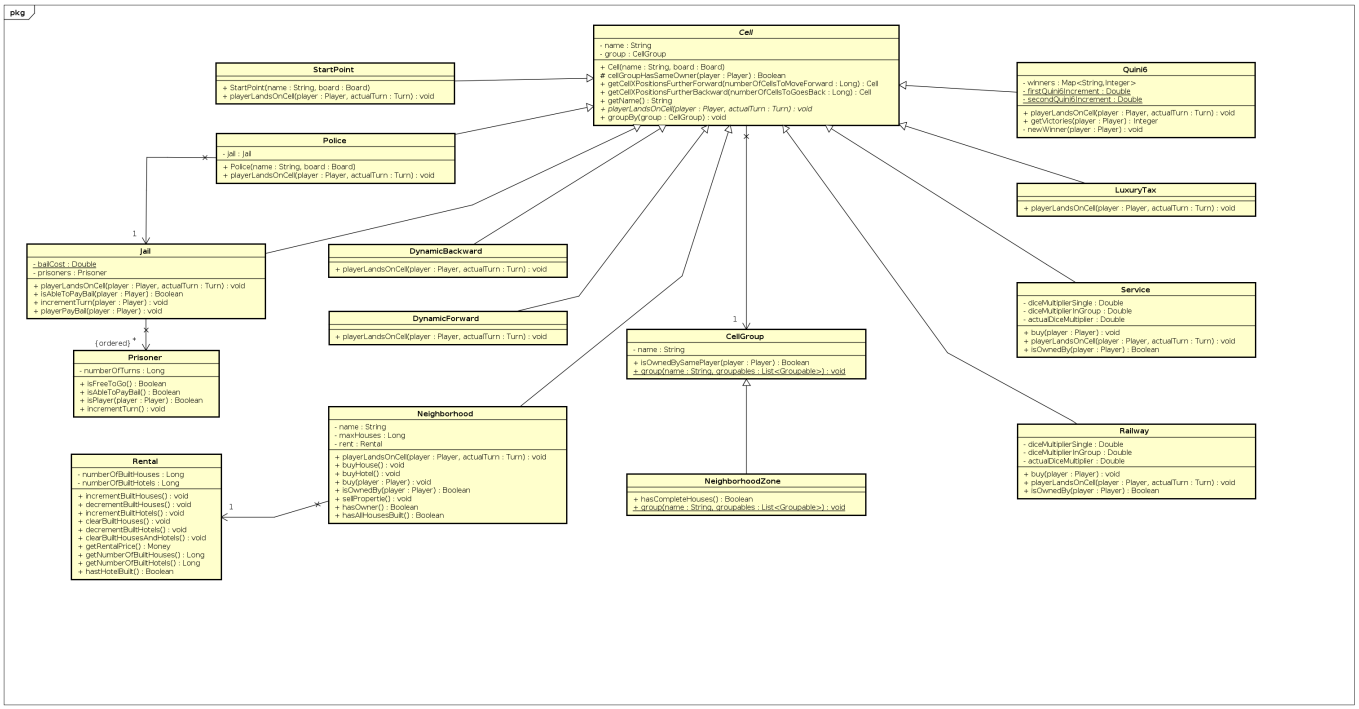


Figura 2: Diagrama de clases - Celdas

## Jugabilidad

A continuación se presentan las clases que definen el compartamiento del jugador con el tablero y tambien las clases que definen las distintas formas de movimiento que posee el jugador

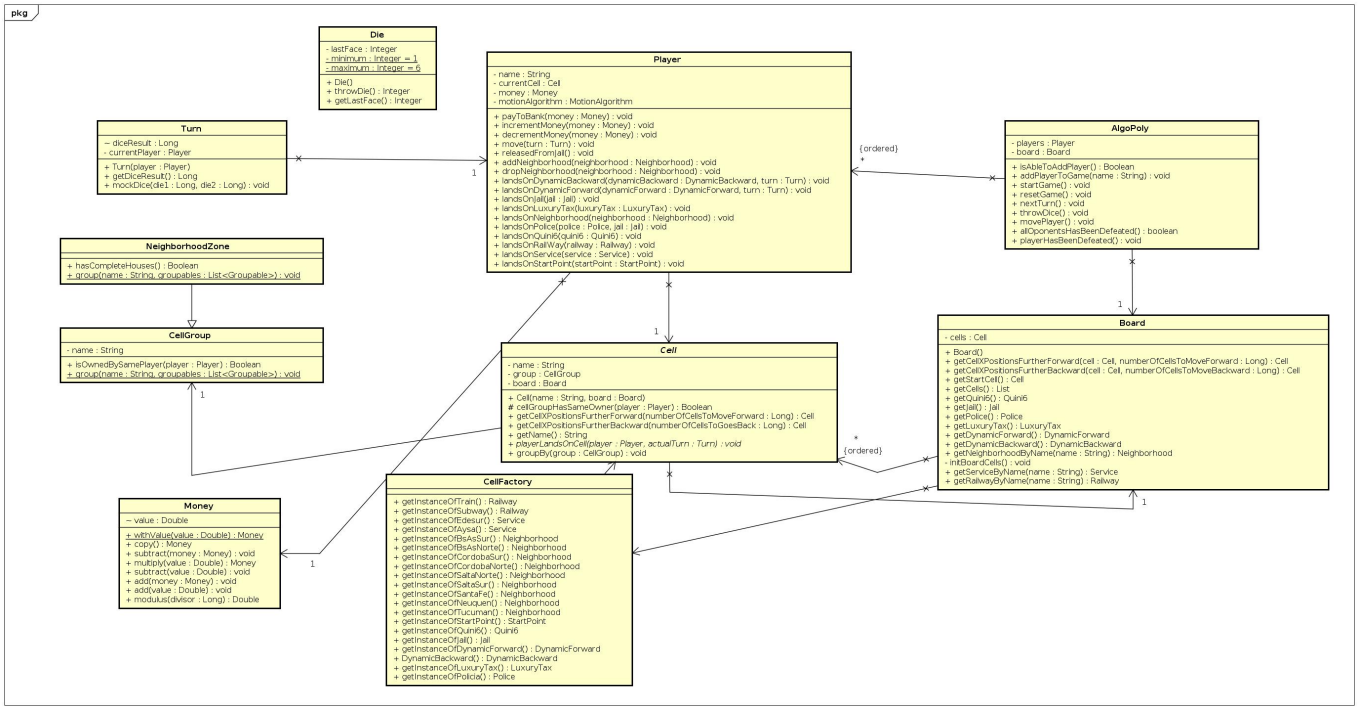


Figura 3: Diagrama de clases - Jugabilidad

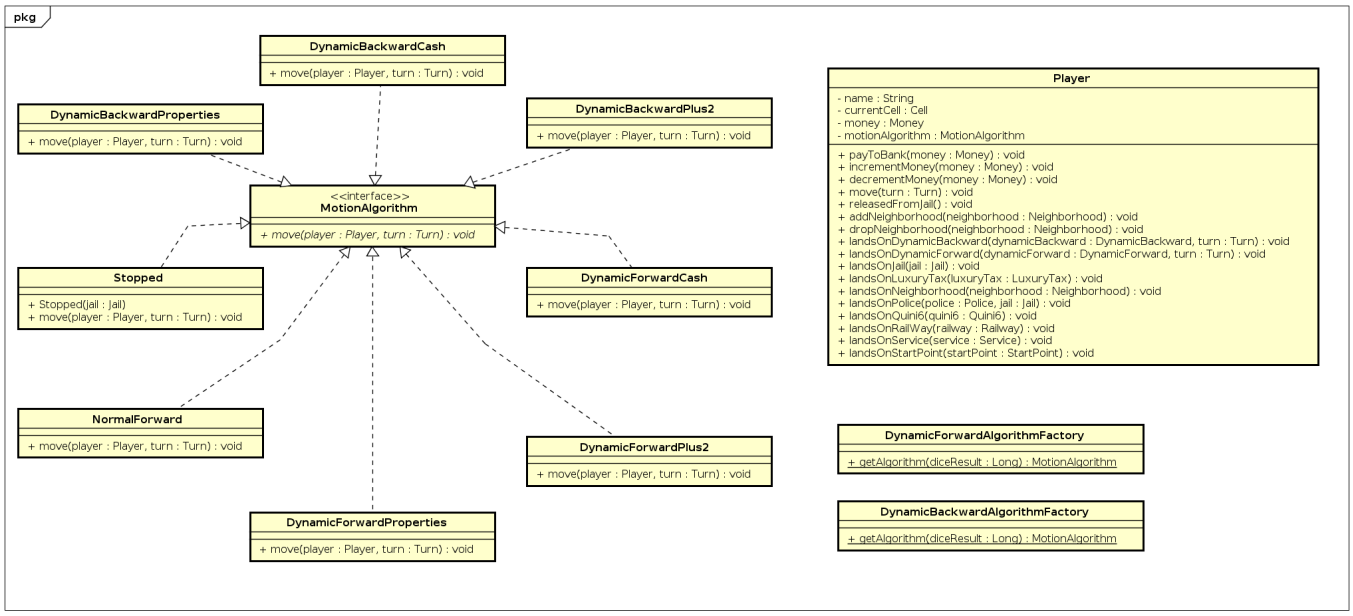


Figura 4: Diagrama de clases - Movimientos

## 4.2. Diagramas de secuencia

Una de las secuencias implementadas es la que se utiliza para comunicar entre si instancias de las clases Cell y Player.

A continuación se muestra dicho diagrama de secuencia:

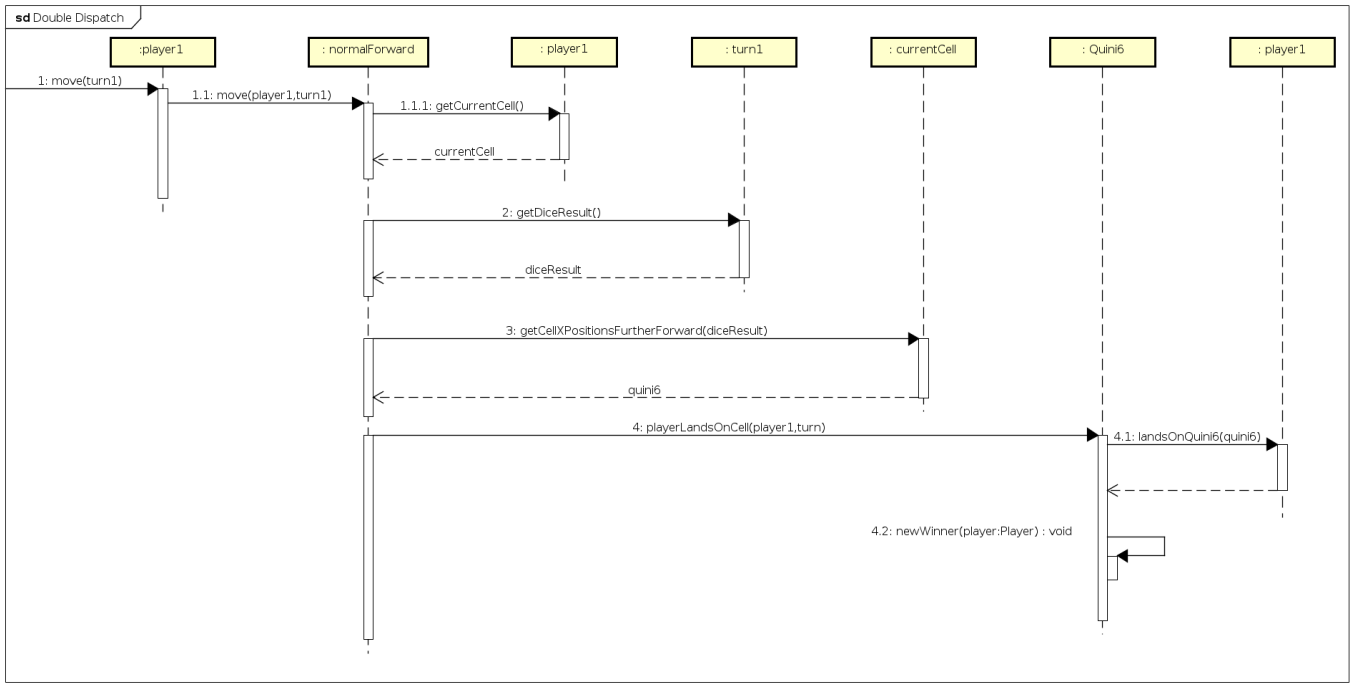


Figura 5: Diagrama de secuencia - Double dispatch

### 4.3. Diagrama de Paquetes

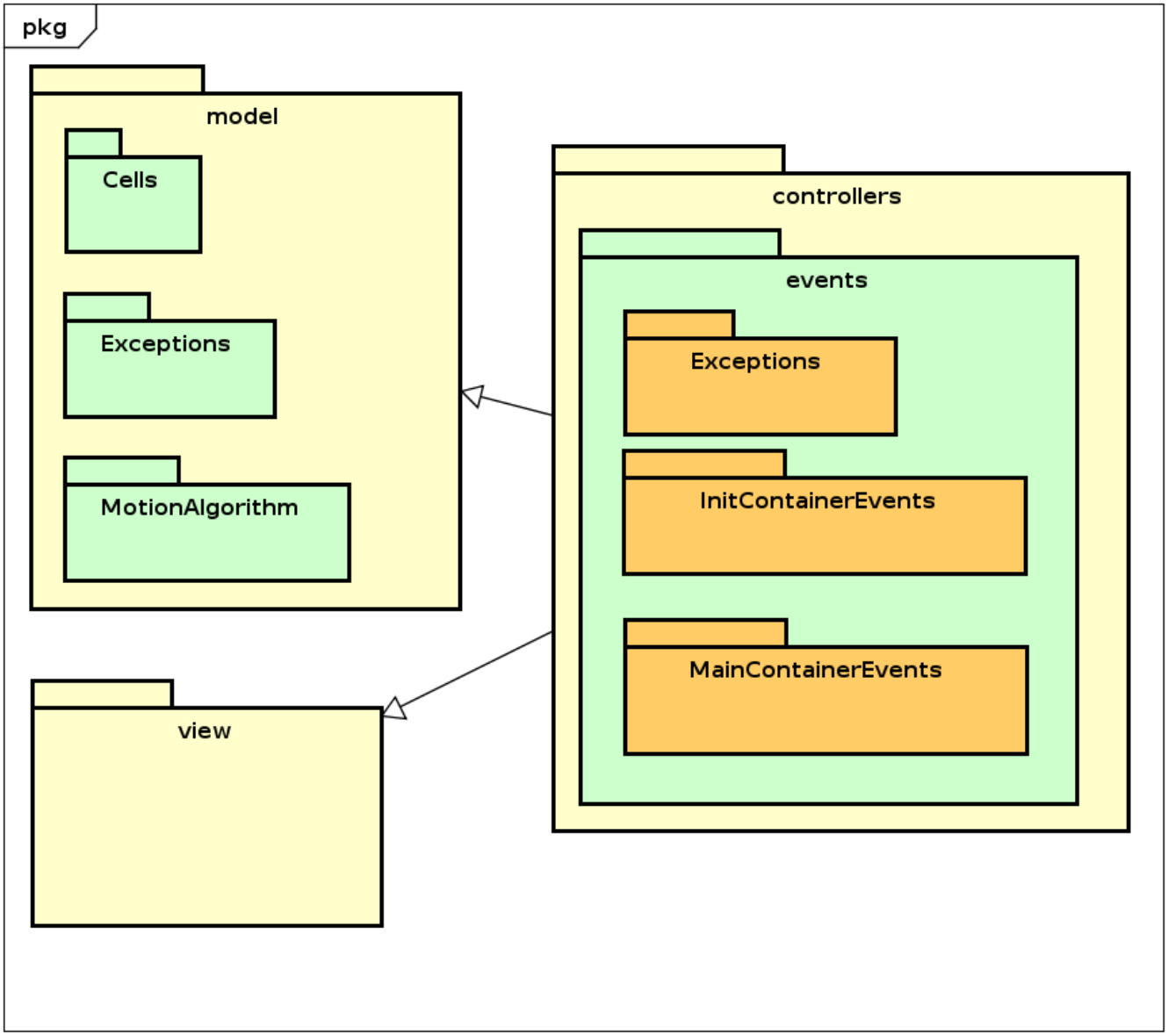


Figura 6: Diagrama de paquetes

## 5. Detalles de implementación

Básicamente se siguió el patrón modelo-vista-controlador.

1. Modelo: Se modelan entidades principales: Player, MotionAlgorithm y Cells, representando al jugar, los posibles movimientos y las celdas del tablero respectivamente. Estas entidades se complementan con otras que logran su funcionamiento, como Board, AlgoPoly, Money, Position, etc.

2. Vista: Representa la interfaz del usuario. Se compone de un tablero (imágen canvas); barra lateral para controlar el transcurso del juego, dados, venta y compra de propiedas; y finalmente una barra lateral para mostrar información del jugador, como por ejemplo las propiedades compradas y el dinero. Utiliza las entidades MainContainer y OwneableHelper.

3. Controlador: Se encarga de orquestar el juego, actuando ante las acciones de los usuarios e invocando clases de la vista y del modelo.

Se modeló el movimiento de los jugadores por medio de una interfaz llamada MotionAlgorithm. Este comportamiento lo adquieren las otras clases del paquete. Permiten definir explicitamente los movimientos al estar en la carcel, con deuda, en bancarrota, adelanto dinámico, atraso dinámico, adelanto normal, etc. Para construir las clases de adelanto y atraso dinámico se utilizó el patrón de diseño Factory.

Las celdas se modelan a partir de una clase abstracta padre llamada Cells. Cada una se construye a partir de métodos de una clase factory llamada CellsFactory. A su vez, cada celda puede adquirir comportamientos definidos por las interfaces Groupable y Owneable. Se definieron clases para agrupar las diferentes celdas: CellGroup y NeighborhoodZone.

Los valores de los diferentes parámetros estáticos definidos por los requerimientos dados se toman de un archivo ubicado en el paquete config. Los mismos se levantan por medio de una clase denominda Global.

Board es la única clase que se encarga de invocar los contructores de las celdas y almacena los estados de las mismas conforme avanza el juego.

AlgoPoly controla el flujo del juego, contiene al tablero y maneja los diferentes turnos.

# 6. Excepciones

**AlgoPolyPlayerQuantityException** Se lanza cuando se intentan agregar más jugadores de los que se había establecido.

**AlgoPolyNoActualTurnException** Se lanza cuando se intenta jugar y no hay un turno corriendo.

**CellNotFoundException** Se lanza ante un desborde de la lista de celdas. Se intenta alcanzar una celda que no existe.

**PlayerExceptions** :

1. InsufficientFundsException: Se lanza cuando se intenta ejecutar una acción que requiere usar más dinero del que se dispone.
2. PlayerInBankruptcyException: Se lanza cuando el jugador intenta realizar una acción teniendo una deuda que saldar.
3. PlayerNotAbleToPayBailException: Se lanza cuando el jugador intenta pagar la fianza y aún no lo tiene permitido.

**NeighborhoodExceptions** :

1. NeighborhoodFullHousesException: Se lanza cuando se intenta construir pero tiene la capacidad ocupada.
2. NeighborhoodWithHotelAlreadyBuiltException: Se lanza cuando el jugador intenta construir mas de 1 hotel.
3. NeighborhoodWithOutAllHousesBuiltException: Se lanza cuando el jugador intenta comprar un hotel sin tener completa su capacidad de casas.
4. NeighborhoodWithOutOwnerException: Se lanza cuando el jugador intenta edificar sin que el terreno tenga dueño.
5. NeighborhoodWithOwnerException: Se lanza cuando el jugador intenta comprar un vecindario con dueño.
6. NeighborhoodZoneWithOutSameOwnerException: Se lanza cuando el jugador trata de edificar sin que posea el vecindario completo.