Trabajo Práctico 2: AlgoPoly

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

Nicolás Daniel Vazquez
100338
vazquez.nicolas.daniel@gmail.com

Eliana Gamarra
100016
elianagam2@gmail.com

Javier Albarracín
97568
jalbarracn@gmail.com

Camila Serra
97422
camilaserra5@gmail.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 aplicando los conceptos enseñados en la materia a la resolución de un problema, trabajando en forma grupal y utilizando un lenguaje de tipado estático (Java).

2. Supuestos

- Cuando un jugador cae en **Avance Dinámico** o **Retroceso Dinámico** el mismo será movido hacía otro casillero. Al ser movido, el jugador visitará la nueva casilla. Es decir, si un jugador cae en **Avance Dinámico** y avanza 3 casilleros, caerá en **Impuesto de Lujo** y tendrá que pagar el impuesto.
- Cuando un jugador visita **Avance Dinámico** habiendo sacado 11 o 12, la cantidad de casilleros que debería avanzar se calcula: tirada - cantidadDePropiedades. Se asume que si el jugador tiene más propiedades que su tirada (11 o 12), retrocederá esa cantidad de casilleros.
- Cuando un jugador visita **Retroceso Dinámico** habiendo sacado 2,3,4,5 o 6, la cantidad de casilleros que debería retroceder se calcula: tirada - cantidadDePropiedades. Se asume que si el jugador tiene más propiedades que su tirada (2,3,4,5 o 6), avanzará esa cantidad de casilleros.

3. Modelo de dominio

En primer lugar, se creó la clase **Jugador**, para representar a un jugador del AlgoPoly. Esta clase se encarga de mantener el capital de un jugador, sus propiedades adquiridas, y además tiene diferentes estados.

Luego, se creo la interfaz **Casillero** para representar a cada uno de los casilleros del juego. Luego, los casilleros tendrán clases especificas que implementen dicha interfaz. Esto se hizo para unificar a todos los casilleros y que todos respondan al mismo mensaje: **recibirJugador**. Cada uno de los casilleros, sobrecargará el método y lo implementará de acuerdo a lo especificado.

Para los casilleros **Avance Dinámico** y **Retroceso Dinámico** se crearon las clases: **AvanceDinamico** y **RetrocesoDinamico**. Estas clases lo que hacen es, recibir al jugador, y en base a la última tirada del jugador, modificar su posición.

Por otro lado, se creó la clase **Quini6** para representar su casillero. Está clase lo que hace es incrementar el capital del jugador (pero sólo las primeras 2 veces).

Además, se crearon las clases **Carcel** y **Policia**. La clase **Carcel** representa su casillero, y lo que hace es modificar el estado del **Jugador** de forma que éste queda inhabilitado al visitar la **Carcel**. El jugador luego es el encargado de saber si se puede mover, si está habilitado para pagar la fianza, o si no puede hacer ninguna acción. La clase **Policia**, está relacionada con **Carcel**, ya que, al visitar policía, el jugador es enviado a la cárcel.

Para representar las propiedades, se creo la clase **Propiedad**. Esta clase tiene como atributo "propietario"que indica que **Jugador** la posee. Además, el jugador tiene una lista de propiedades de la cuál es dueño.

4. Diagramas de clase

El diagrama muestra las relaciones entre las clases creadas.

Se puede apreciar que la interfaz **Casillero** tiene una dependencia con la clase **Jugador**. Esto es porque los casilleros deben conocer al jugador para así poder modificarlo como corresponda.

Por otro lado, se puede ver como las clases **Quini6**, **AvanceDinamico**, **RetrocesoDinamico**, **Propiedad**, **Carcel** y **Policia** implementan a la interfaz **Casillero** tal como fue explicado anteriormente.

La interfaz **Estado** sirve para manejar el estado de un jugador. Este puede ser **Habilitado** o **Encarcelado**. Estas clases sobrecargan los métodos: **puedeMoverse**, **puedeEjecutarAcciones** e **iniciarTurno**.

Por último se ve que el **Casillero** de tipo **Propiedad** tiene una asociación con **Jugador**, y a su vez, **Jugador** con **Propiedad**. La propiedad tiene como propietario a un jugador, y el jugador tiene una lista de propiedades de la cual es propietario.

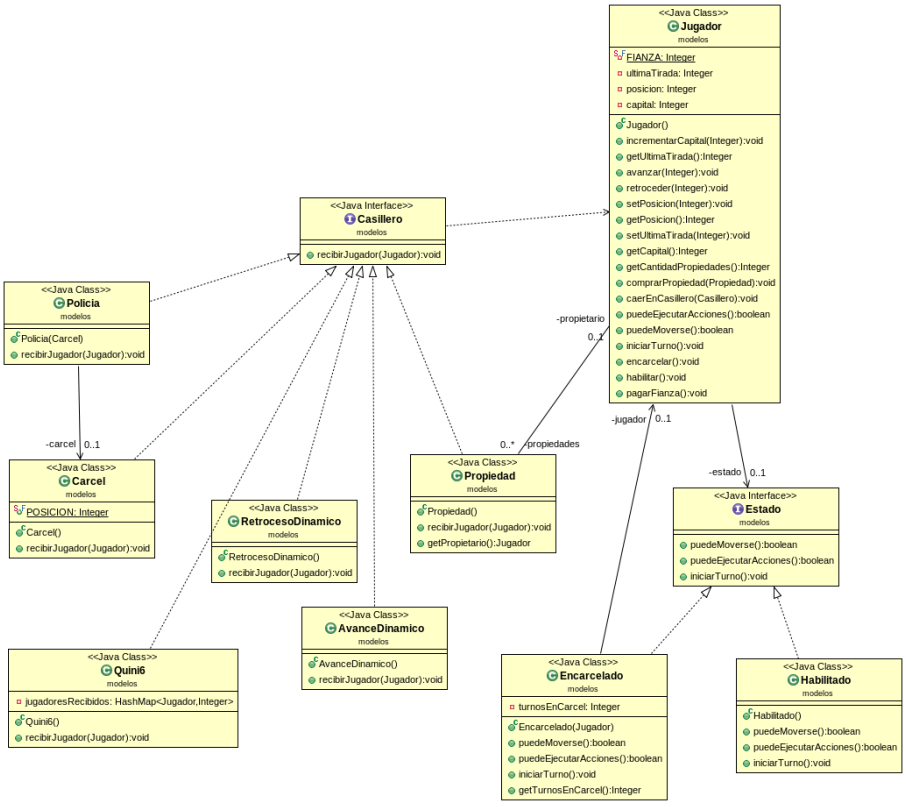

Figura 1: Diagrama del AlgoPoly.

En más detalle la implementación de los Casilleros.

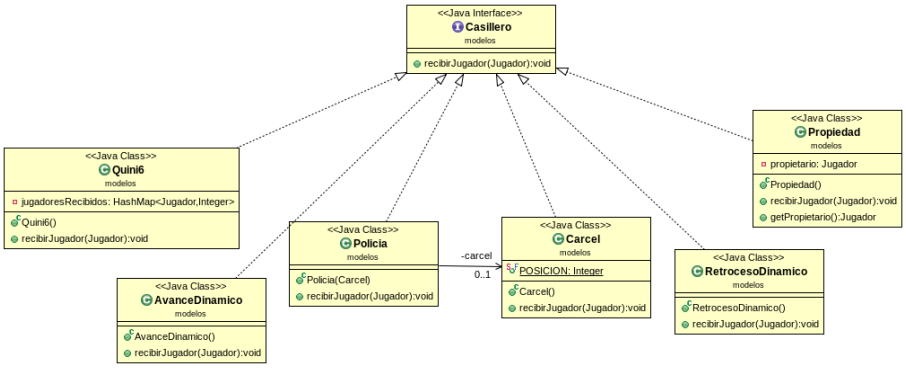

Figura 2: Diagrama de los casilleros.

5. Detalles de implementación

5.1. Interfaz Casillero

La interfaz **Casillero** se creó para que todas las casillas sigan el contrato que se plantea, ya que, por más que cada casilla tenga una acción diferente sobre el jugador, representan lo mismo. Se decidió que el mensaje a sobrecargar sea **recibirJugador**, ya que cuando un jugador cae en la casilla, la casilla lo recibe y luego actúa sobre él.

```
public interface Casillero {
    void recibirJugador(Jugador jugador);
}
```

5.2. Calculo premio Quini6

Para calcular que premio recibe el jugador que está visitando la casilla **Quini 6** se implementó un mapa que guarda la cantidad de visitas de los jugadores. En base a esto se calcula cuanto será el premio. Si es la primera o segunda vez, se incrementará el capital del jugador, pero de la tercera vez en adelante, no pasará nada.

```
jugadoresRecibidos.put(jugador,
   jugadoresRecibidos.containsKey(jugador) ? jugadoresRecibidos.get(jugador) + 1 : 1);
if (jugadoresRecibidos.get(jugador) == 1) {
    jugador.incrementarCapital(50000);
} else if (jugadoresRecibidos.get(jugador) == 2) {
    jugador.incrementarCapital(30000);
}
```

5.3. Interfaz Estado

Para manejar los estados del jugador se creó la interfaz **Estado**. Esta interfaz tiene los siguientes métodos:

```
public interface Estado {
    boolean puedeMoverse();
    boolean puedeEjecutarAcciones();
    void iniciarTurno();
}
```

En principio, se crearon las clases **Habilitado** y **Encarcelado**, que implementan esta interfaz. La clase habilitado es el estado normal de un Jugador, en el cual se puede mover y ejecutar acciones normalmente. Por el contrario, al tener como estado Encarcelado, el jugador no se podrá mover ni ejecutar acciones en el primer turno, y luego podrá ejecutar acciones (pagar fianza), y por último luego de 3 turnos podrá moverse y volverá a estar Habilitado.

Habilitado:

```
@Override
public boolean puedeMoverse() {
    return true;
}

@Override
public boolean puedeEjecutarAcciones() {
    return true;
}
```

Encarcelado:

```
@Override
public boolean puedeMoverse() {
    return false;
}

@Override
public boolean puedeEjecutarAcciones() {
    return this.turnosEnCarcel > 1;
}
```

6. Diagramas de secuencia

En el siguiente diagrama se muestra como funciona la casilla **Avance Dinámico** cuando un jugador cae en ella habiendo sacado 8 con los dados. El jugador avanzará la cantidad de casilleros equivalentes a hacer el módulo entre su capital y su tirada.

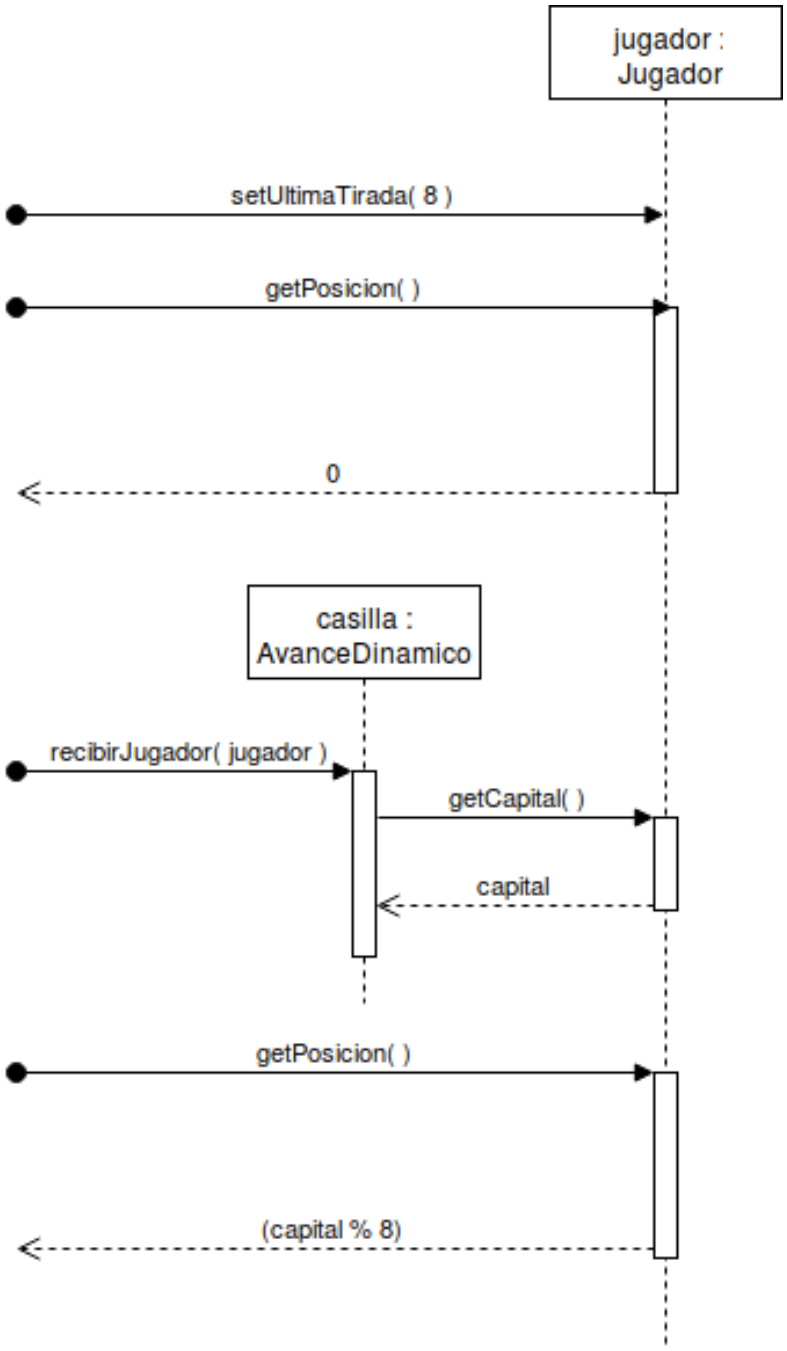

Figura 3: Avance Dinámico.

En este diagrama se muestra cómo funciona la casilla **Quini6** cuando un jugador cae en ella cuatro veces. Su capital sólo se verá afectado en la primer y segunda visita.

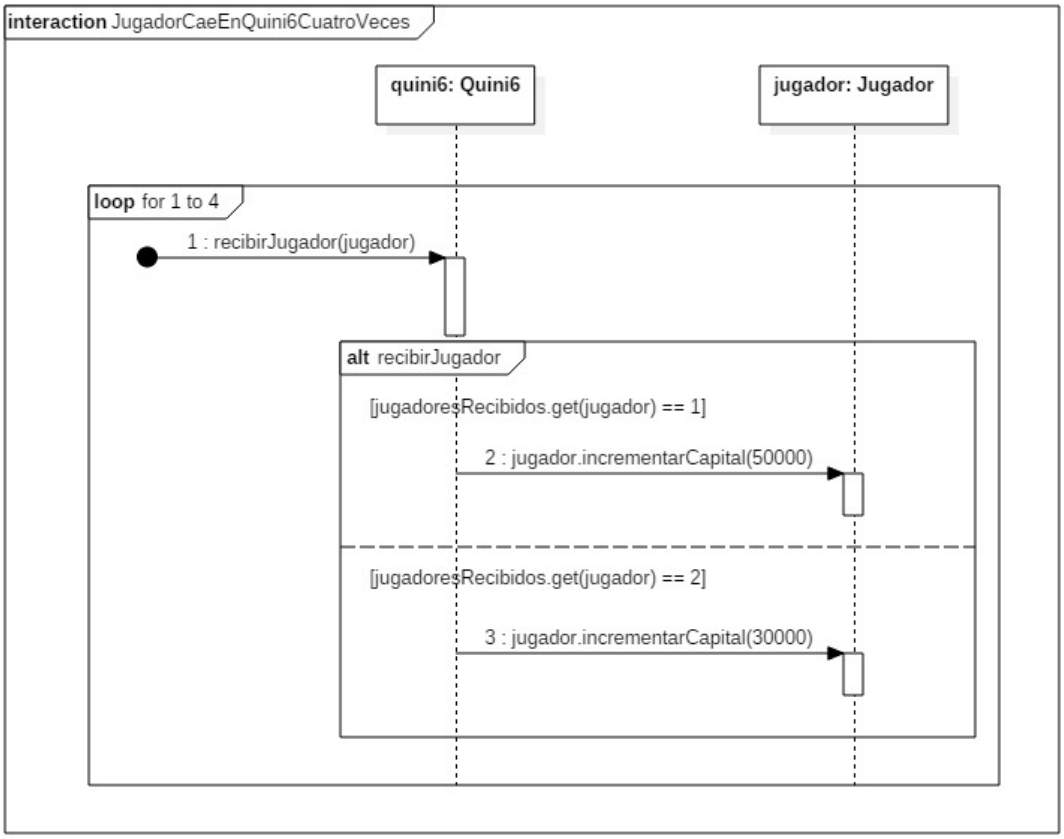

Figura 4: Quini6.

El siguiente diagrama muestra la interacción entre el **Jugador** y el casillero **Carcel** luego de caer en dicho casillero. El método encarcelar asigna una instancia de **Encarcelado** al estado del **Jugador**.

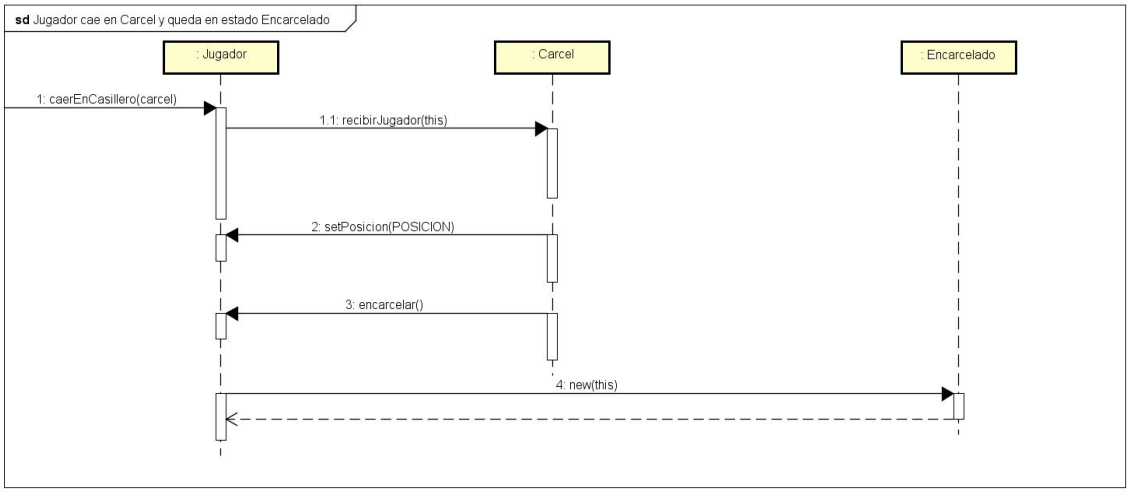

Figura 5: Jugador cae en Carcel.

El siguiente diagrama muestra como luego de pagar una fianza se modifica el estado del **Jugador** a **Habilitado**. El diagrama supone que el **Jugador** se encuentra en condiciones de pagar la fianza, esto es, transcurrió al menos un turno desde que fue encarcelado y tiene un capital mayor a 45000.

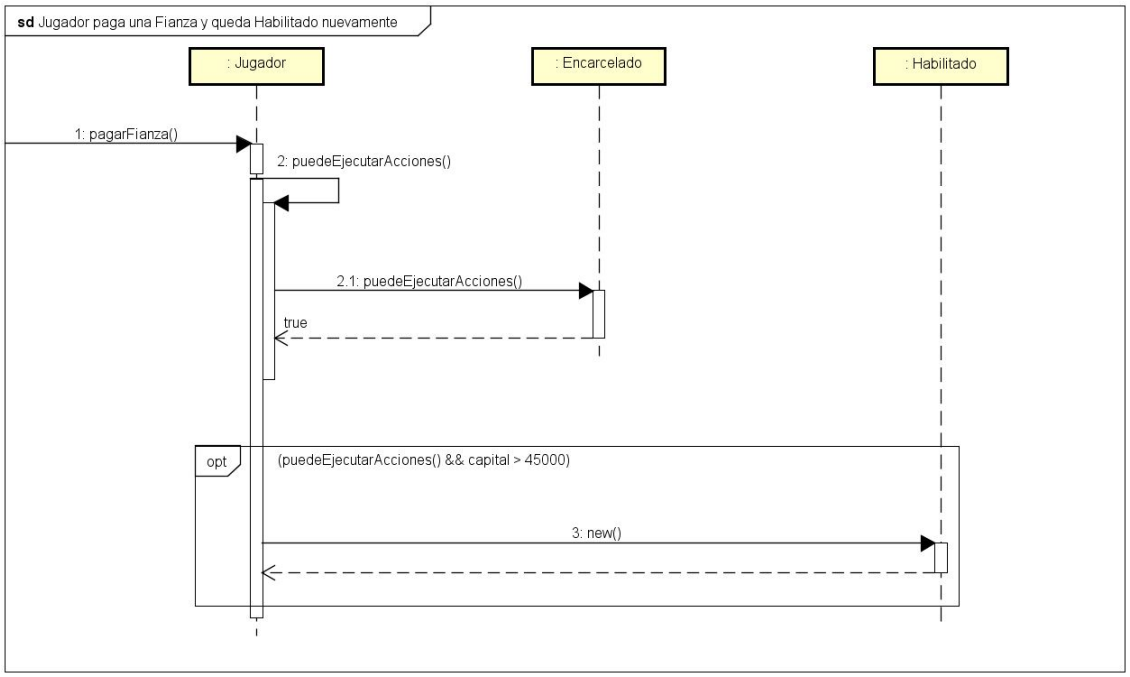

Figura 6: Jugador paga una fianza.

El siguiente diagrama muestra como luego de transcurridos cuatro turnos encarcelado se modifica el estado del **Jugador** a **Habilitado**. El diagrama supone que el método inicial iniciarTurno() está siendo invocado por cuarta vez desde que el **Jugador** fue encarcelado.

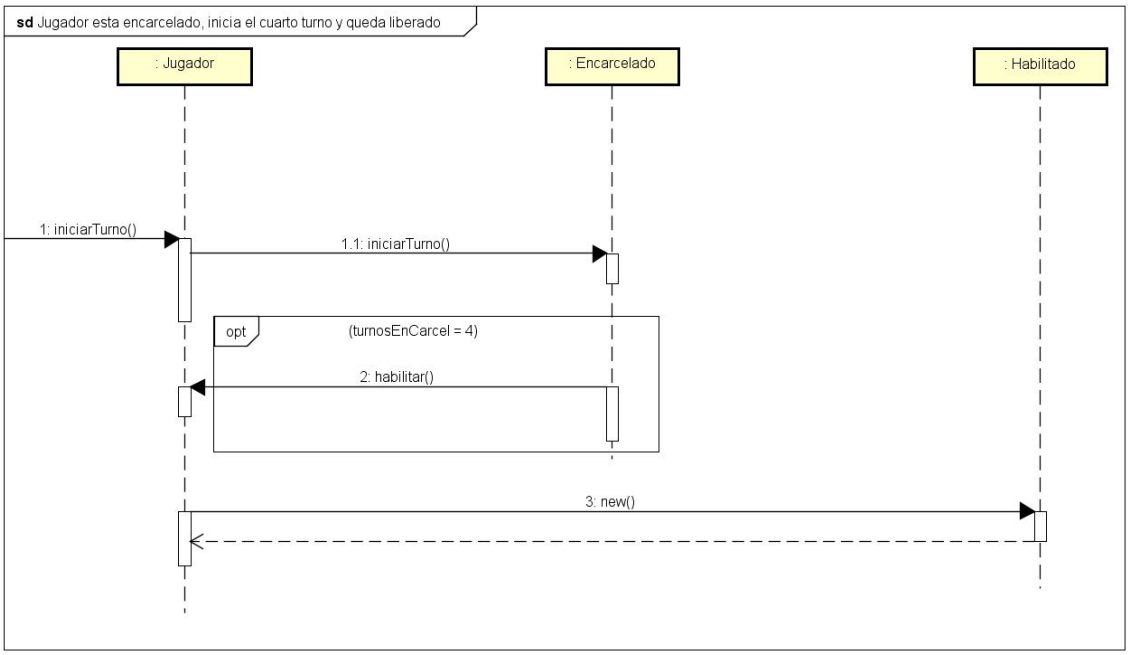

Figura 7: Jugador queda liberado luego de cuatro turnos.

7. Diagramas de paquete

En el siguiente diagrama se muestra como se realizó la estructura de los paquetes. En principio se creó el paquete algobay para agrupar todo. Después se dividió en modelos/vistas/controladores. Luego, dentro de modelos, se dividió en jugador y tablero. En el paquete jugador está la clase Jugador y sus posibles estados. En el paquete tablero, están todos los casilleros.

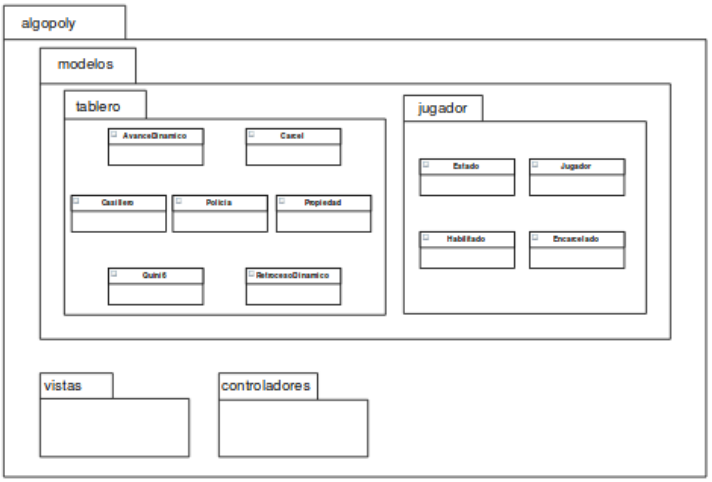

Figura 8: Diagrama de Paquetes.