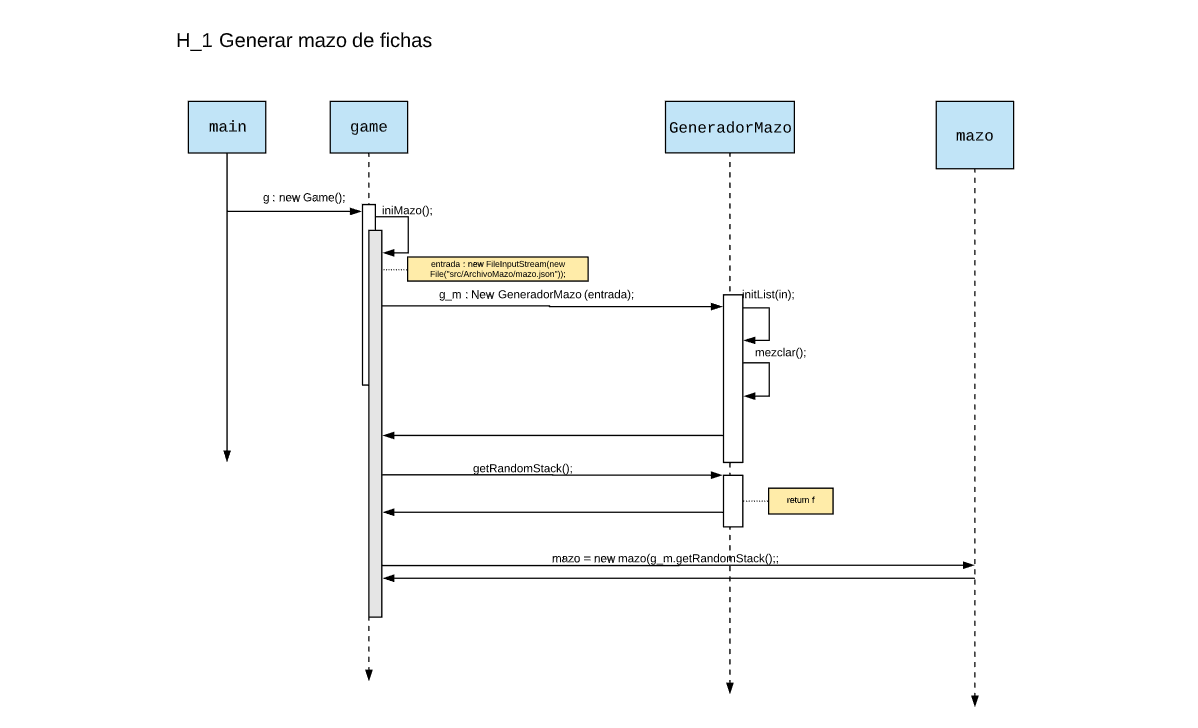
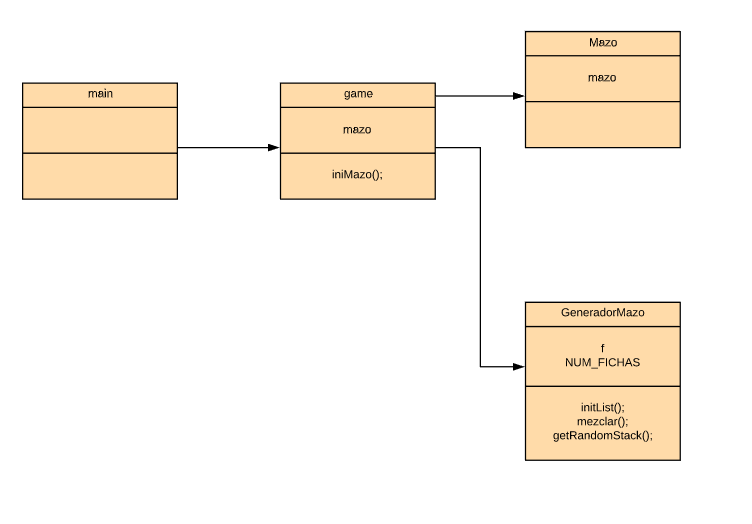
**DIAGRAMAS UML**

1. Generar mazo de fichas
2. Diagrama de secuencias
3. Diagrama de clases



1. Explicación

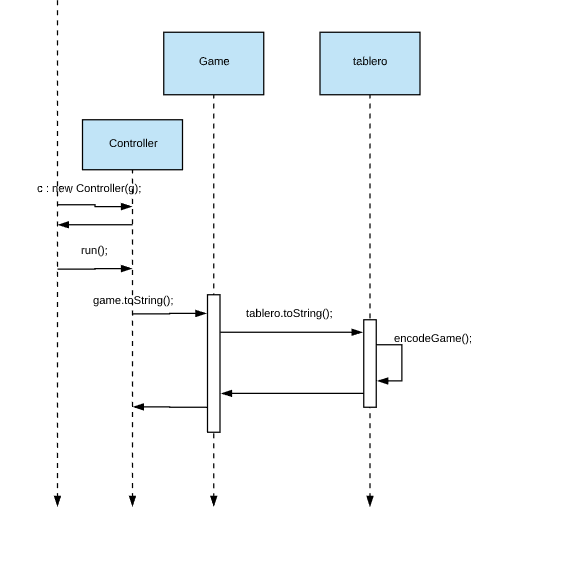
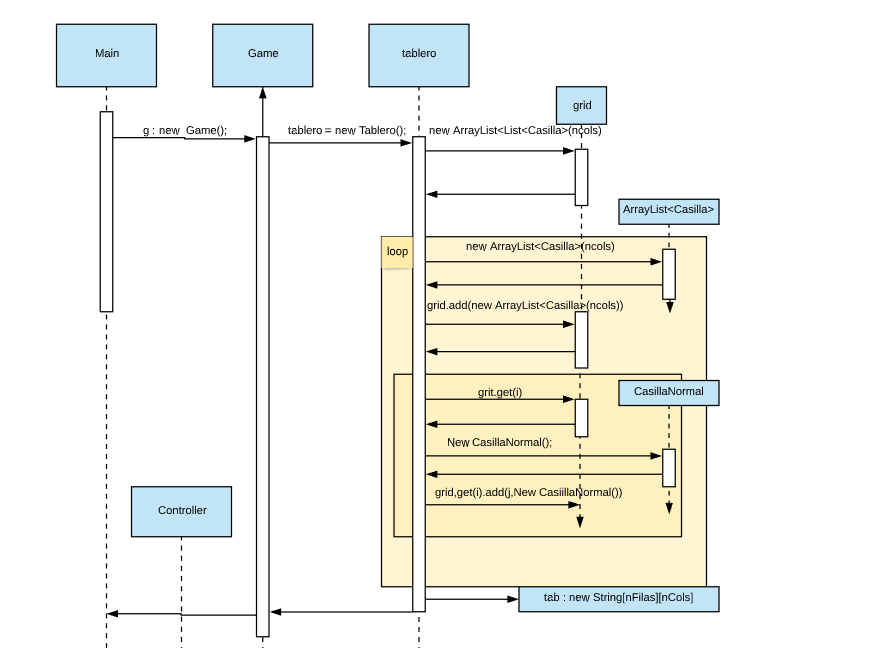
Para generar un mazo aleatorio de fichas, la clase Game llama a un método (iniMazo()) que primero guarda en un FileInputStream el archivo JSON donde tenemos la información sobre las fichas. A continuación, crea una instancia de GeneradorMazo.

Esta clase, es su método initList(in), a partir de un partir de un parámetro de tipo FileInputStream, construye un mazo de Fichas según la información de un archivo JSON, en el que para cada letra del abecedario teniendo en cuenta su frecuencia en el juego, construye una Ficha y la añade a la lista de fichas f.

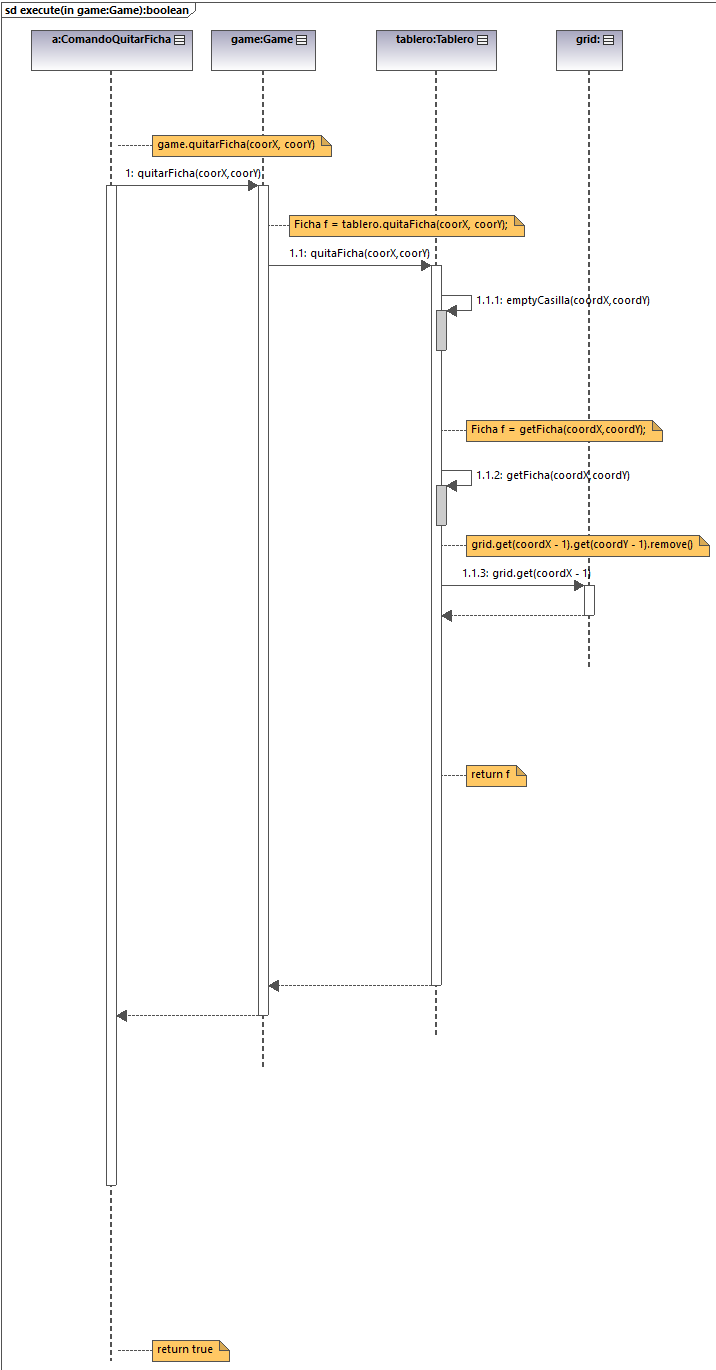
Para terminar, como queremos que sea un mazo aleatorio, llamamos al método mezclar() que realiza una llamada al método shuffle().

De vuelta al Game, creamos un Mazo al que le pasamos una Colección aleatoria de fichas. De esta forma obtenemos un mazo aleatorio de Fichas.

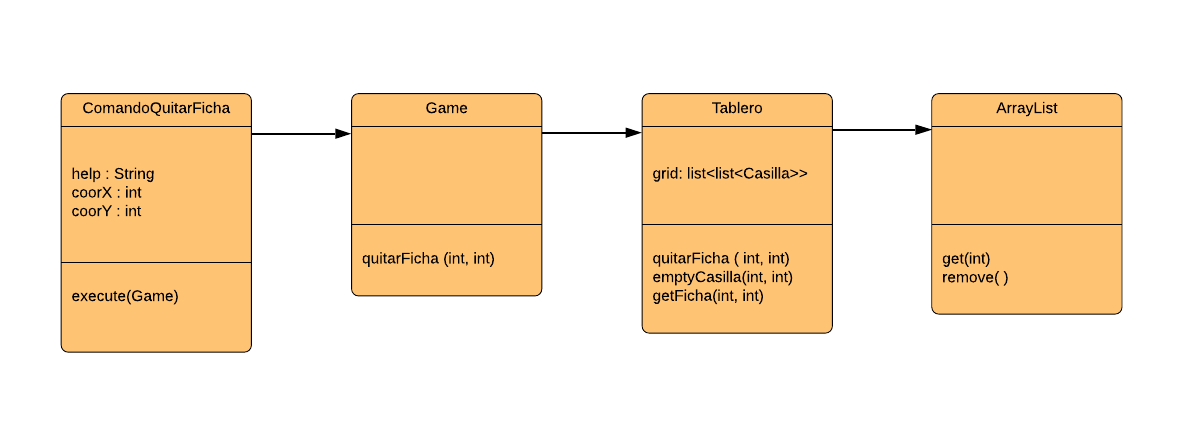
1. Ver el tablero
2. Diagrama de secuencia



1. Quitar ficha
2. Diagrama de secuencia



1. Diagrama de clases



1. Explicación

Para quitar una ficha del tablero, el método execute de la clase ComandoQuitarFicha (que recibe un parámetro de tipo Game) llama al método quitarFicha de la clase Game usando como parámetros los atributos coorX y coorY de la clase ComandoQuitarFicha, posteriormente, el método quitarFicha de la clase Game llama al método quitarFicha de la clase Tablero pasándole dichos parámetros.

El método quitarFicha de la clase Tablero comprueba a través del método emptyCasilla (que recibe como parámetros los parámetros de quitarFicha) si la Casilla seleccionada está vacía, en caso de que no esté vacía lo que hace el método guardar la ficha a extraer en una variable auxiliar de tipo Ficha (para luego poder incorporarla al mazo del jugador) y posteriormente eliminarla de grid usando el método remove (método heredado de la clase ArrayList), en caso de que la Casilla esté vacía el método lanza una excepción.

1. Rellenar Mano
   1. Diagrama de secuencia

Imagen que contiene captura de pantalla

Descripción generada automáticamente

* 1. Diagrama de clases

Imagen que contiene captura de pantalla

Descripción generada automáticamente

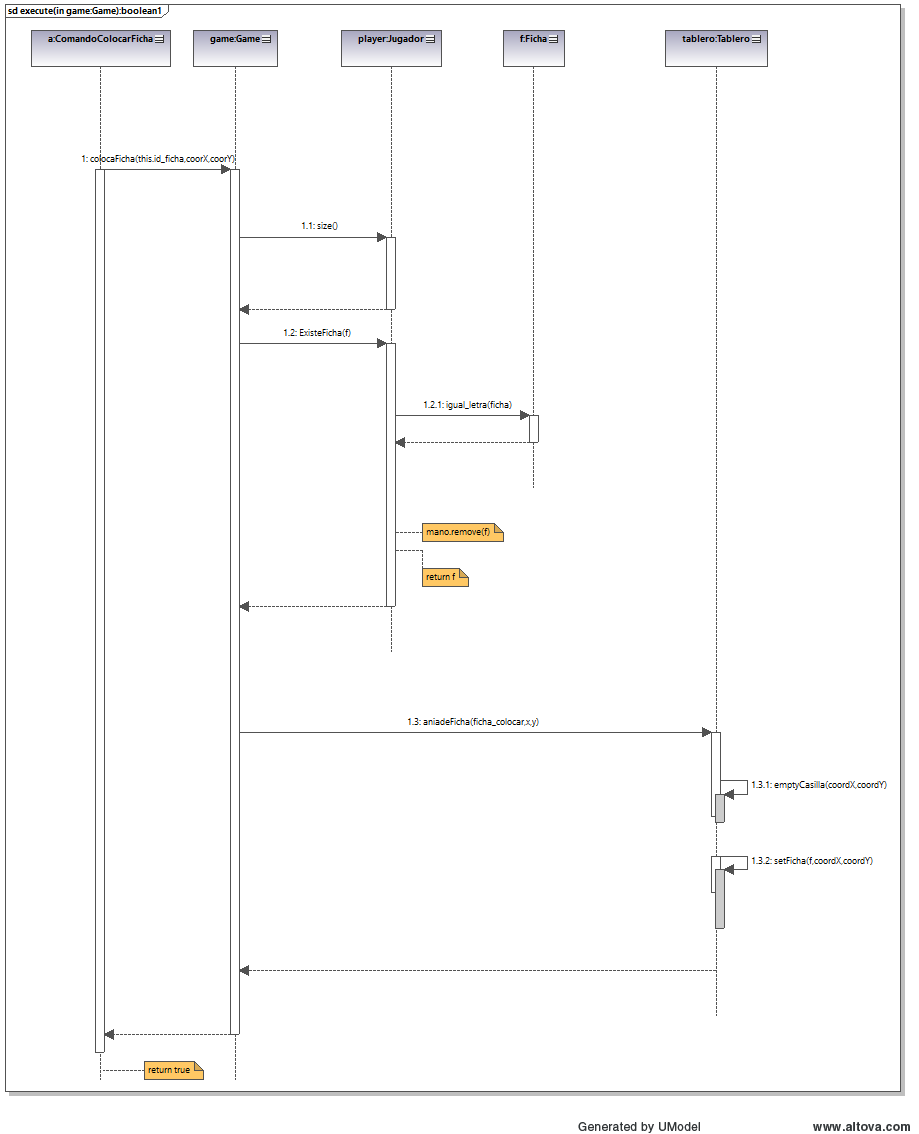
* 1. Explicación

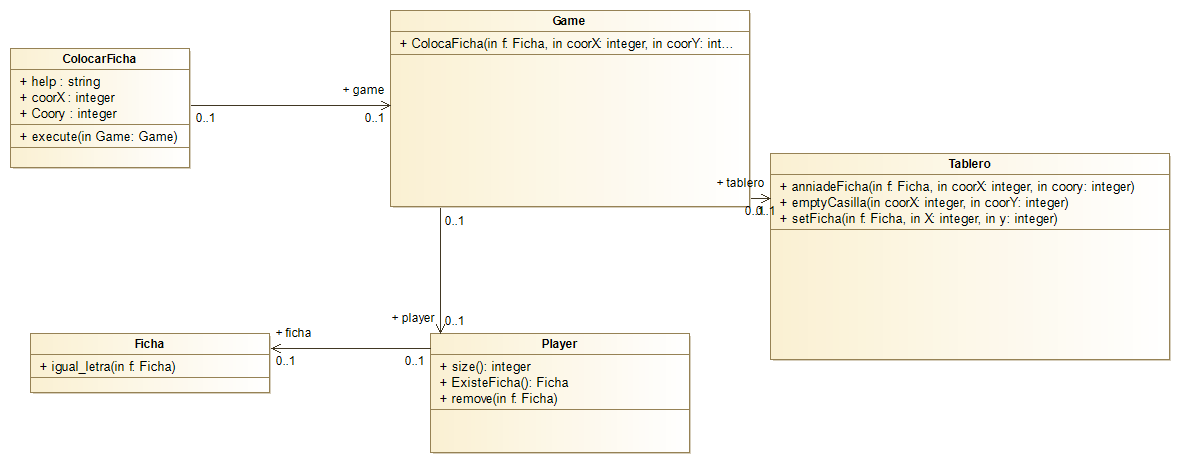
Para rellenar la mano de un jugador, la clase Game llama a un método rellenarManos() que consiste en un bucle while-do que se repite mientras no se llene la mano de un jugador, es decir, mientras no se alcance un máximo que está definido por una constante (número máximo de fichas que puede tener cada jugador).

En cada vuelta del bucle se pedirá a la clase Mazo una ficha a través del método de la clase Game robar(). Una vez obtenida la ficha se llamará a otro método robar(Ficha f) de la clase Jugador en el que se añade la ficha f a la mano del jugador, implementada con una lista de fichas.

1. Colocar Ficha

a. Diagrama de Secuencia

­­­­b. Diagrama de Clases

c. Explicacion:

Para colocar una ficha en el tablero se llama a un metodo de la clase que comprueba si la casilla está vacía. Si lo está, quita de la mano del jugador la ficha que se pasa por consola (si se encuentra en la mano), y la coloca en la casilla indicada por coorX coorY.

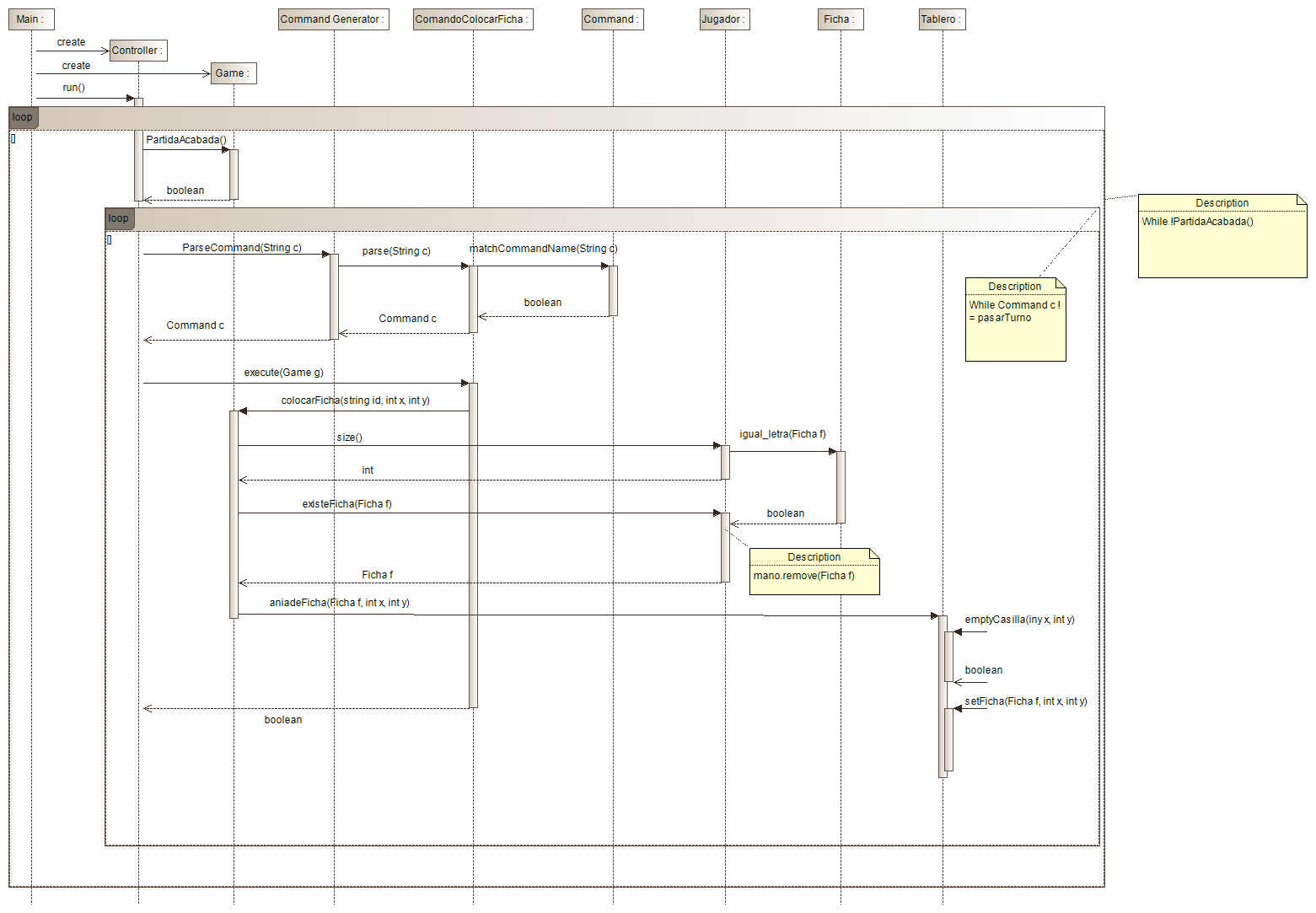
1. Pasar de turno
   1. Imagen que contiene mapa, texto

      Descripción generada automáticamenteDiagrama de secuencia
   2. Captura de pantalla de un celular

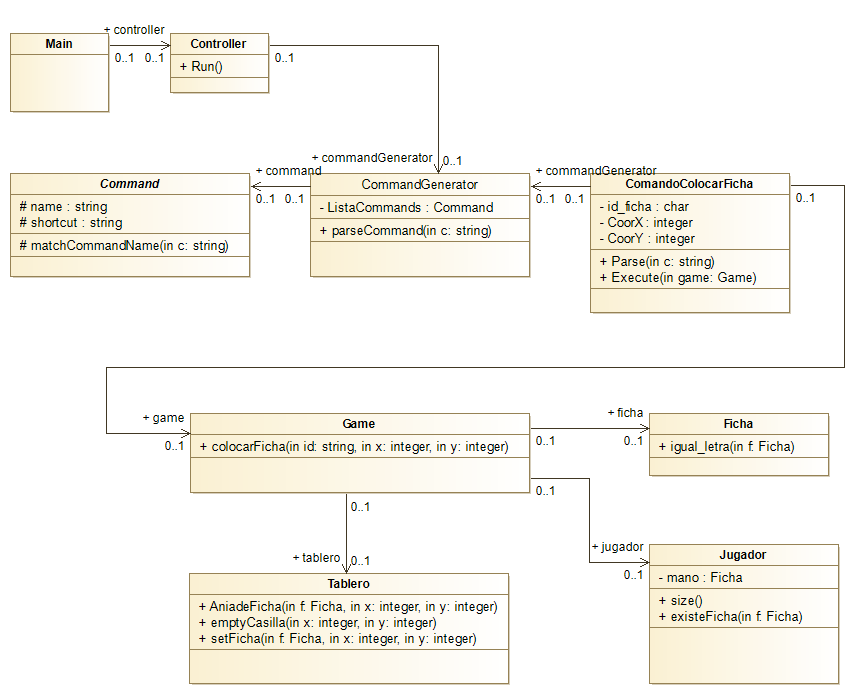
      Descripción generada automáticamenteDiagrama de clases
   3. Explicación

Al iniciar una partida, el Main crea una instancia de Controller e invoca al método run() de Controller. Este método consiste en un bucle while que se ejecuta mientras la partida no finaliza. Dentro del bucle se crea un turno para el jugador actual y se llama al método run() que a su vez llamará al método execute() de la clase ComandoPasarTurno, tras haber parseado el comando. Tras esto se actualiza el turno actual pasando el turno al siguiente jugador y finalmente se guardan los datos de la partida en un fichero.

1. Colocar varias fichas por turno:
   1. Diagrama de secuencia



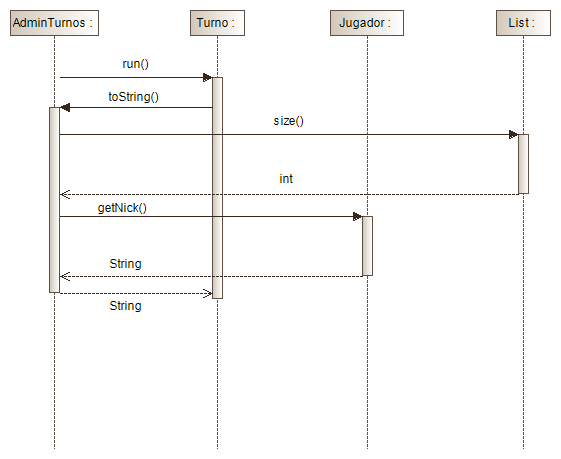
* 1. Diagrama de clases

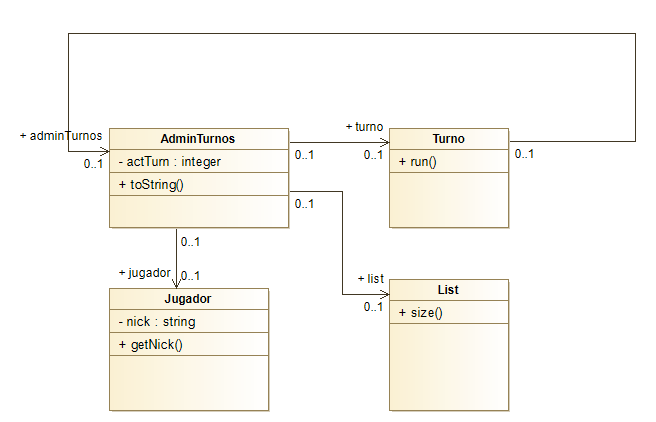


* 1. Explicación

El jugador introduce el comando colocar ficha con parámetros idficha(la ficha que quiere meter en el tablero), posición x y posición, una vez que el CommandGenerator a través de los métodos parse y matchCommandName identifica dicho comando lanza su método execute el cual llama al método colocarFicha de Game, este método llama al método aniadeFicha de Tablero el cual comprueba que la ficha y la posición dada sean válidas y en dicho caso la inserta quitándola de la mano del jugador.

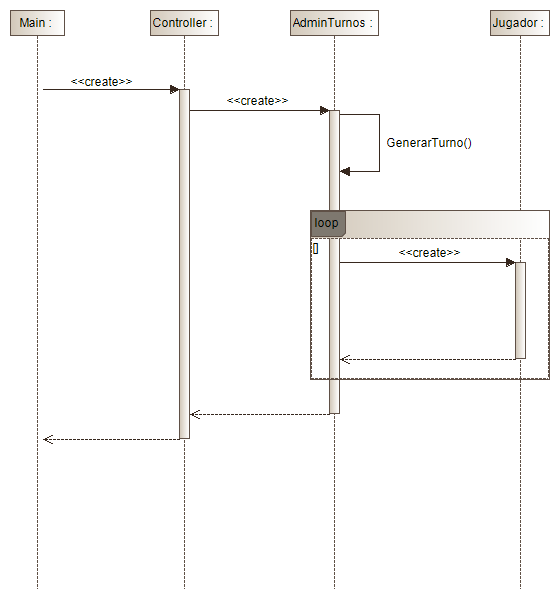
1. Conocer turnos de la partida:
   1. Diagrama de secuencia



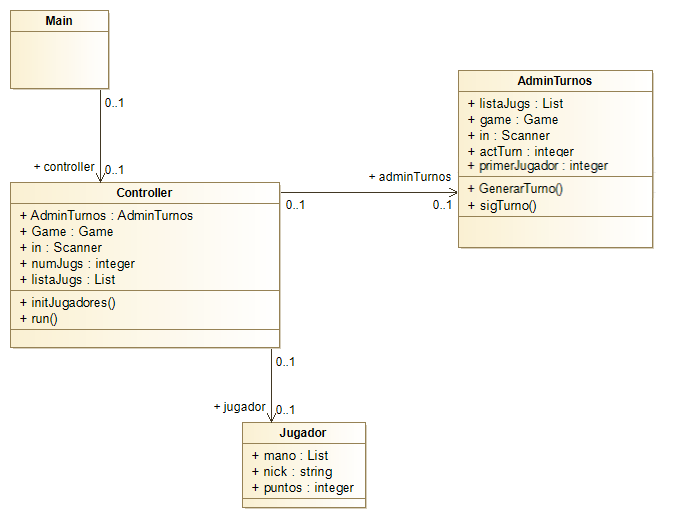
* 1. Diagrama de clases
  2. Explicación

Para mostrar los turnos de los jugadores el método run de Turno llama al método toString de la clase AdminTurnos, dicho método devuelve un String con los nombres de los jugadores en el orden en el que les toca jugar.

1. Saber quién empieza:
   1. Diagrama de secuencia



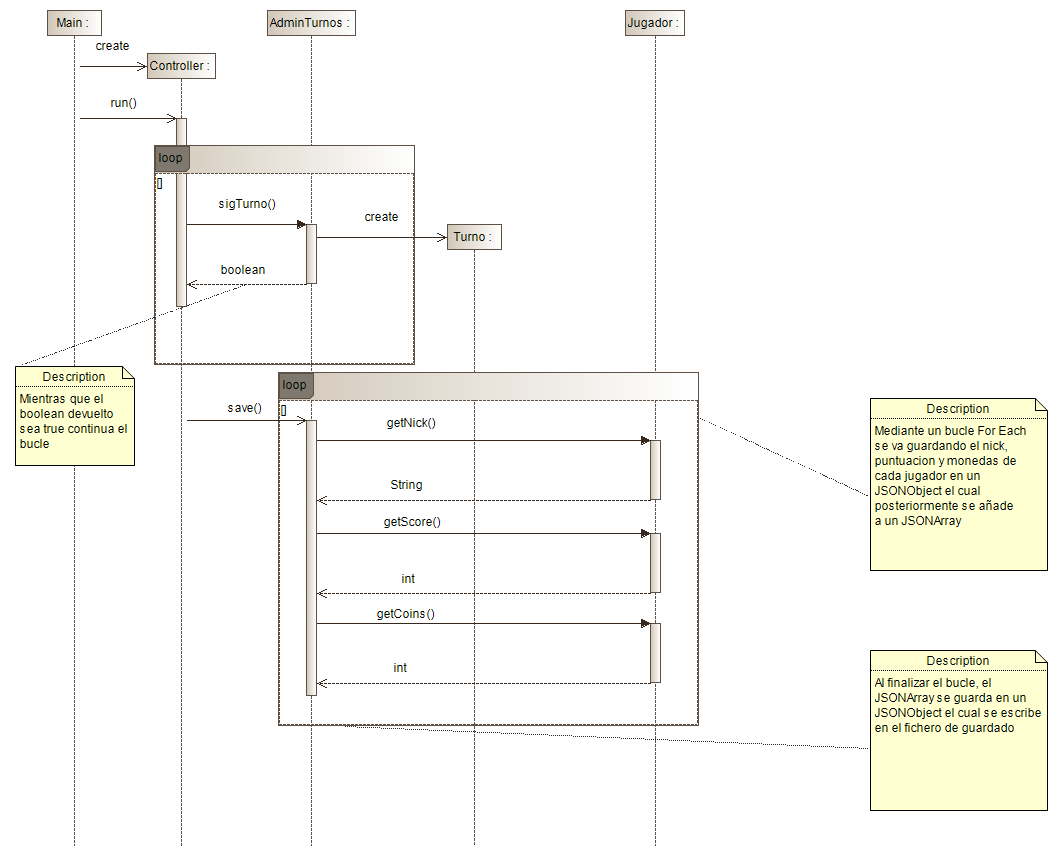
* 1. Diagrama de clases

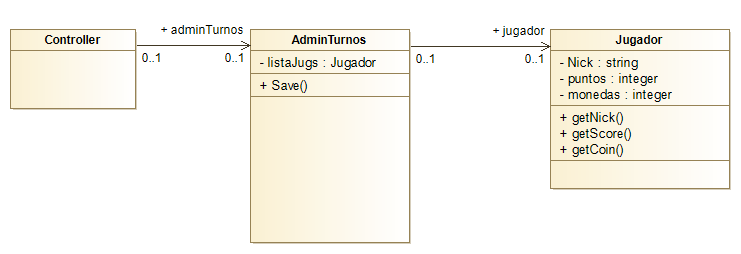


* 1. Explicación

Cuando se crea el administrador de turnos se llama al método generarTurno que devuelve un entero que indicará el jugador que tendrá el primer turno y se guardará en el atributo primerJugador.

1. Guardar partida:
   1. Diagrama de secuencias

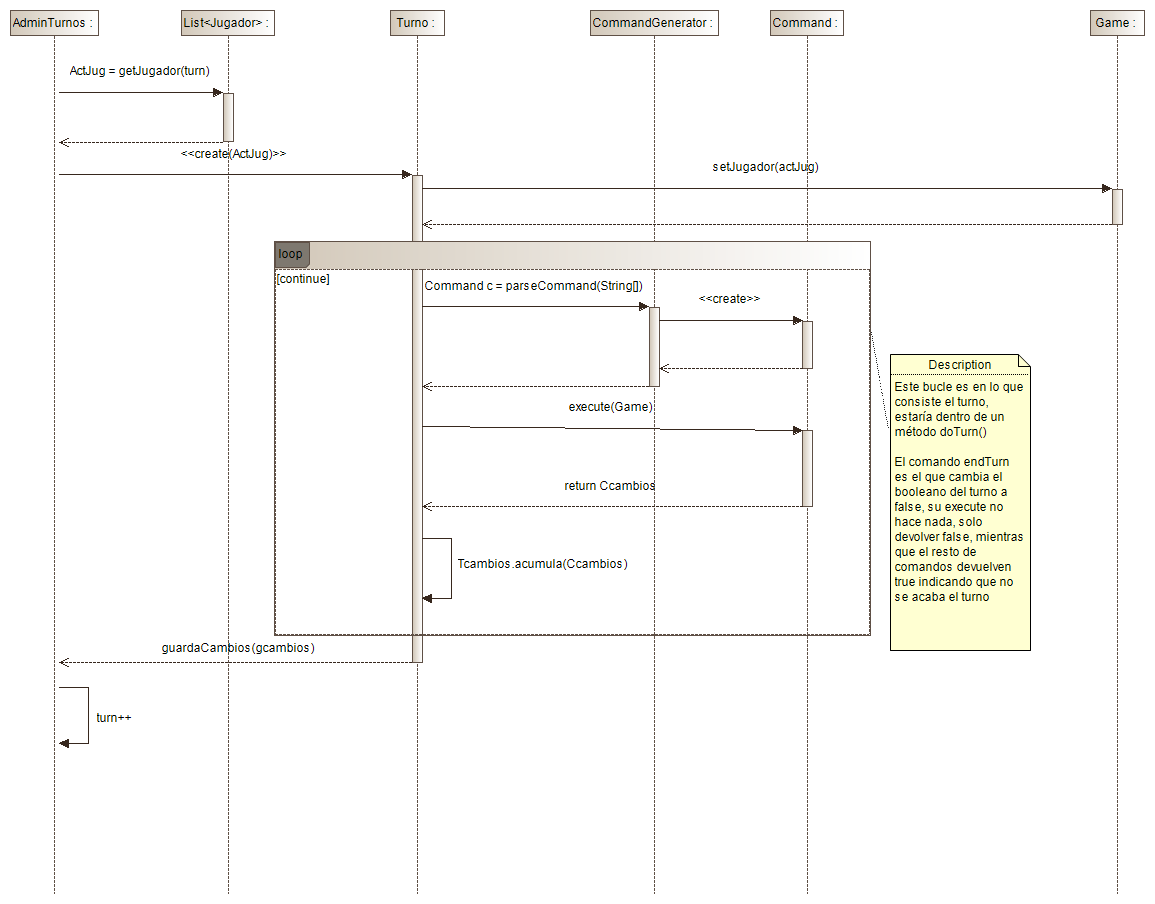


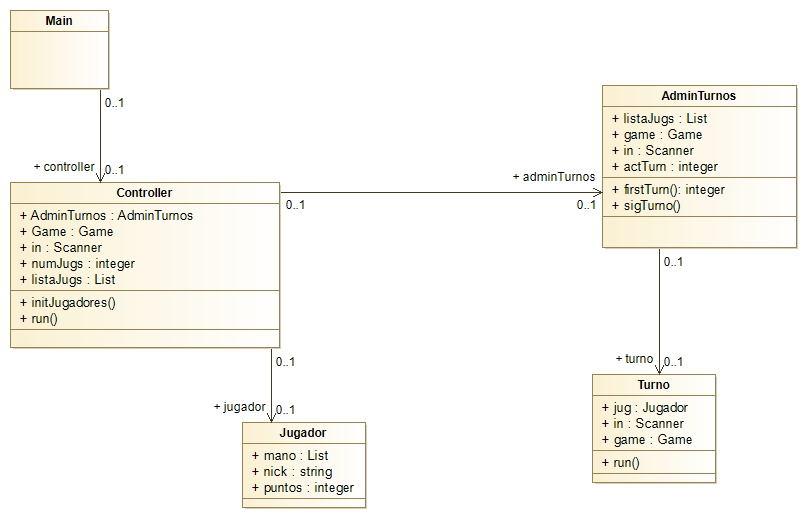
* 1. Diagrama de secuencias
  2. Explicación

Una vez que se sale del bucle de partida del controller se llama al método save de AdminTurnos, una vez dentro de este método se le da la posibilidad al usuario de elegir el nombre del fichero de guardado y se procede a guardar la información de los jugadores, para ello, se recorre la lista de jugadores extrayendo nick, puntuación y monedas de cada jugador y guardándolo primero en un JSONObject y posteriormente en un JSONArray, una vez que se sale del bucle el JSONArray se guarda en otro JSONObject y este último se procede a guardar en el fichero que ha seleccionado el usuario.

11. Siguiente Turno

a. Diagrama de Secuencia

 b. Diagrama de Clases

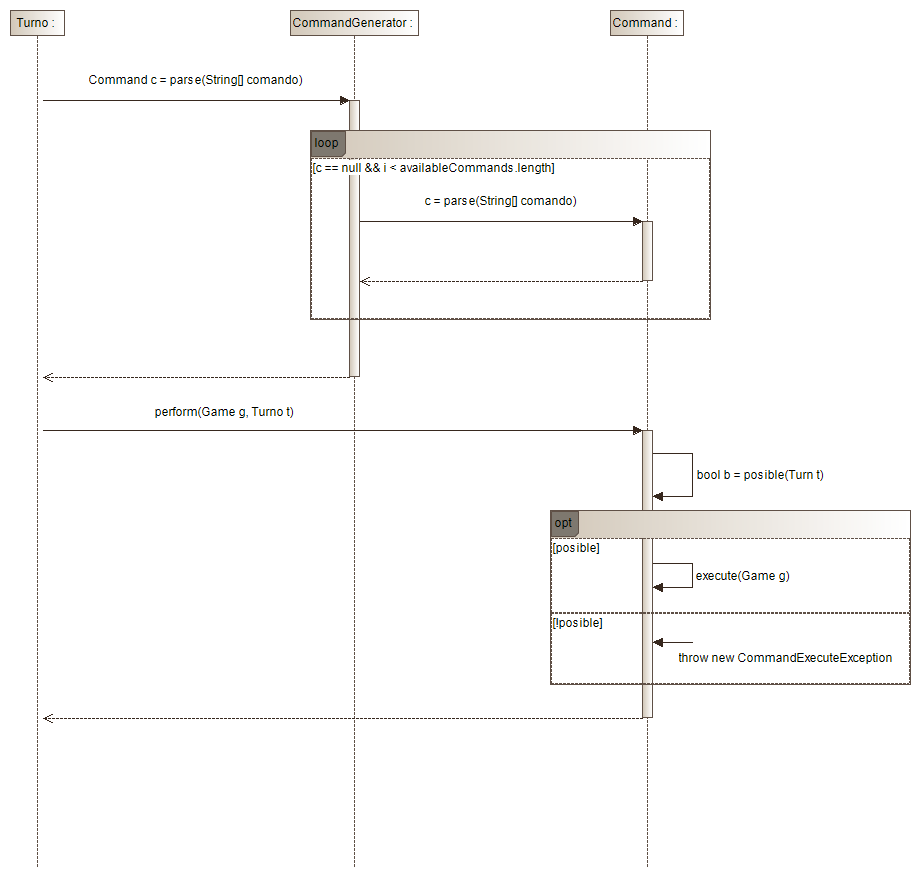


c. Explicación

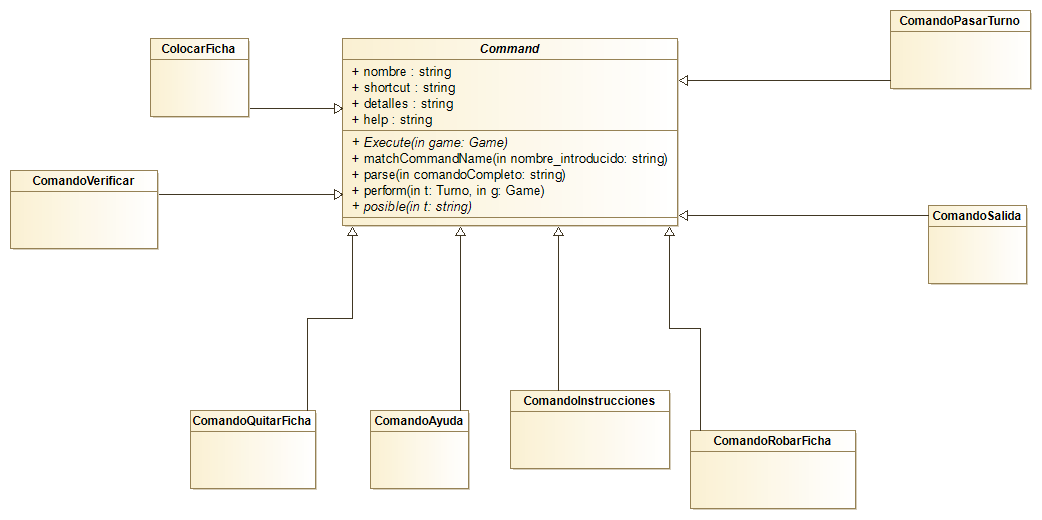
Se ha creado una clase AdminTurnos que se encarga de gestionar los turnos de los jugadores y su orden. Tiene como atributos la instancia de Game única, la lista de jugadores y un indicador de quien es el jugador que tiene el turno. Cada vez que el administrador de turnos otorga un turno a un jugador, se crea una instancia de la clase turno, que encapsula la información de lo sucedido durante un turno. Al comienzo de cada turno, se le indica al Game quien está jugando, alterando su parámettro player, y durante la ejecución , los jugadores tienen la capacidad de ejecutar comandos para interactuar con el juego.

12. Extensión del Patrón Comando

a. Diagrama de Secuencia

.

b. Diagrama de Clases

 c. Explicación:

Se han añadido dos métodos más a los comandos:

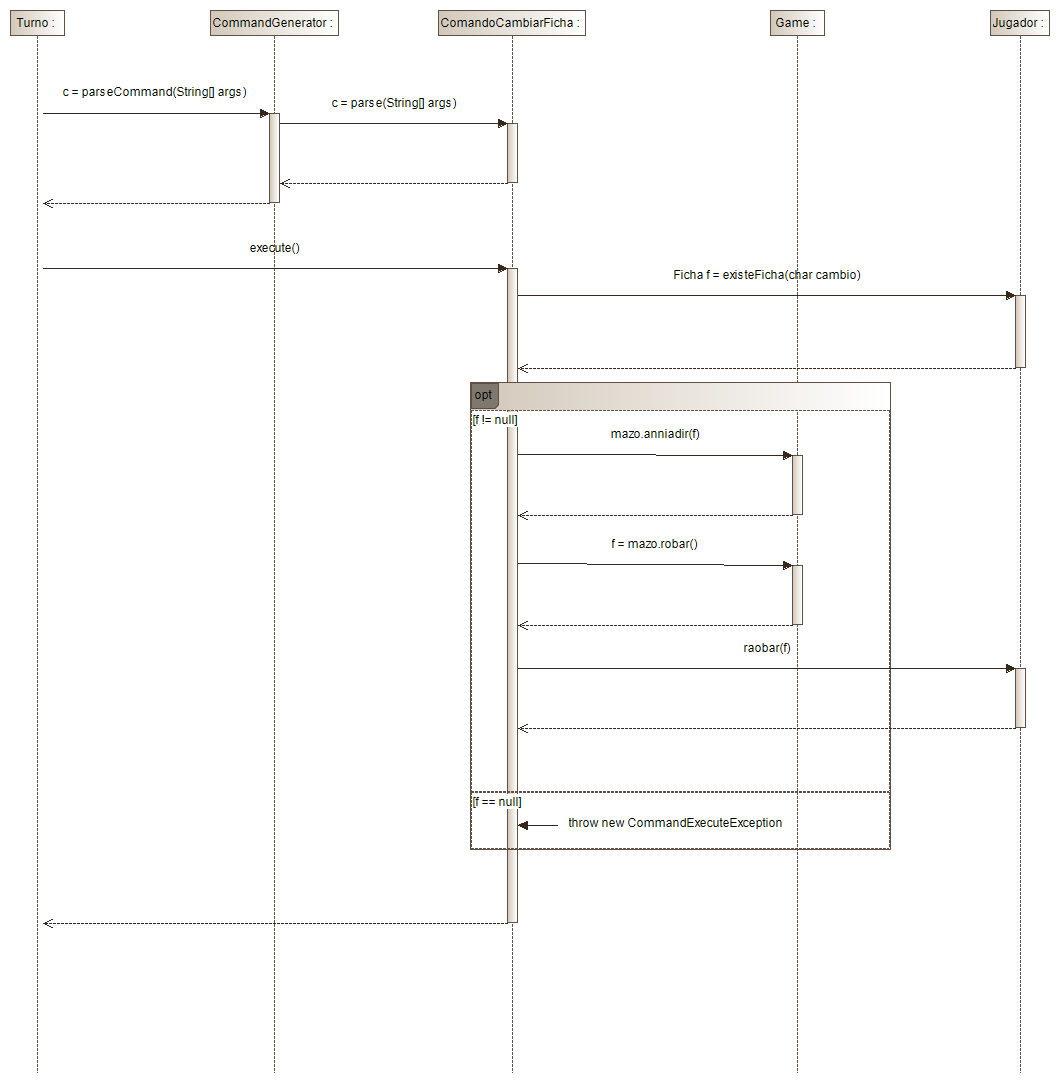
- public boolean Perform(Game g, Turn t) : comprueba que el comando puede ejecutarse dadas las condidiones del turno actual, y si es posible, se ejecuta.

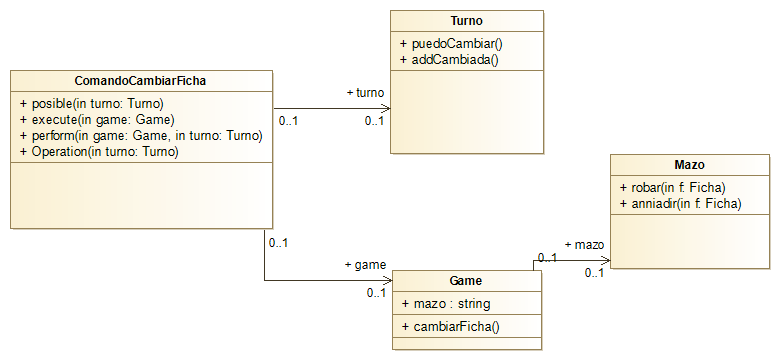
- private abstract boolean Posible(Turn t) : comprueba que el comando puede ejecutarse con respecto a las condiciones actuales del turno. Se hace llamando a un método del turno que se le pasa como parámetro.

El execute y el MatchCommandName se mantienen igual.

13. ComandoCambiarFicha

a. Diagrama de Secuencia:

 b. Diagrama de Clases

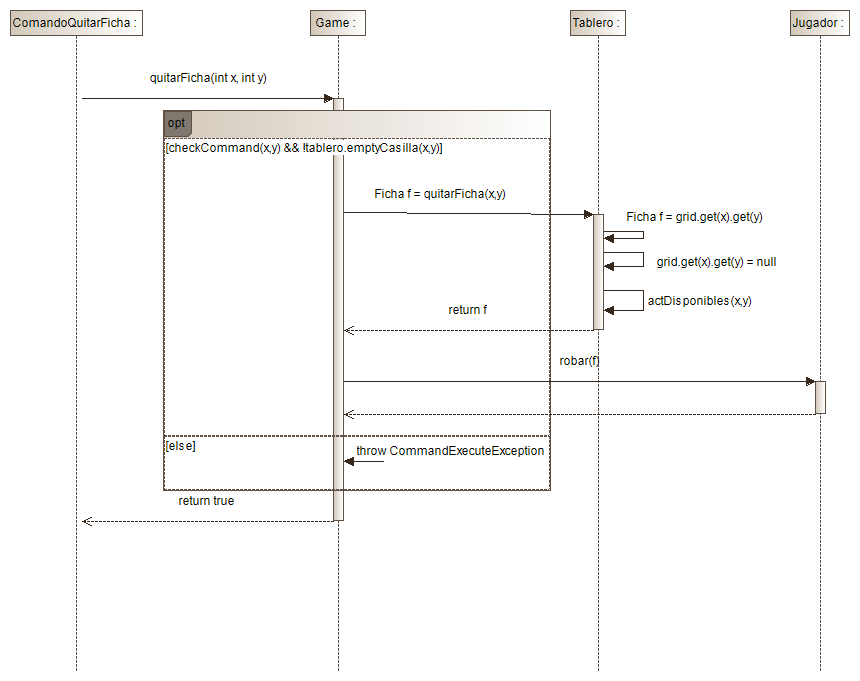


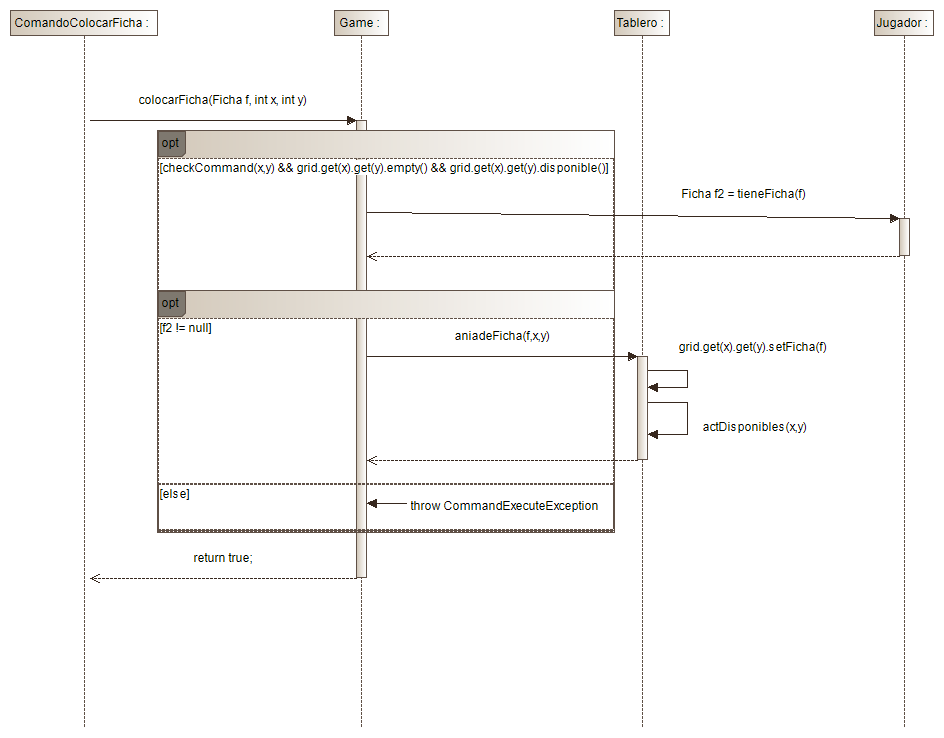
c. Explicación

Se ha añadido un nuevo comando cambiar ficha. Su condicion para poder ejecutarse es que se hayan cambiado menos de 7 fichas durante el turno, y su ejecución consiste en meter la ficha que se quiere cambiar en el mazo, sacar una ficha del mazo y ponerla en la mano del jugador

14.Cambios a colocar/quitar ficha

a.Diagrama de Secuencia:

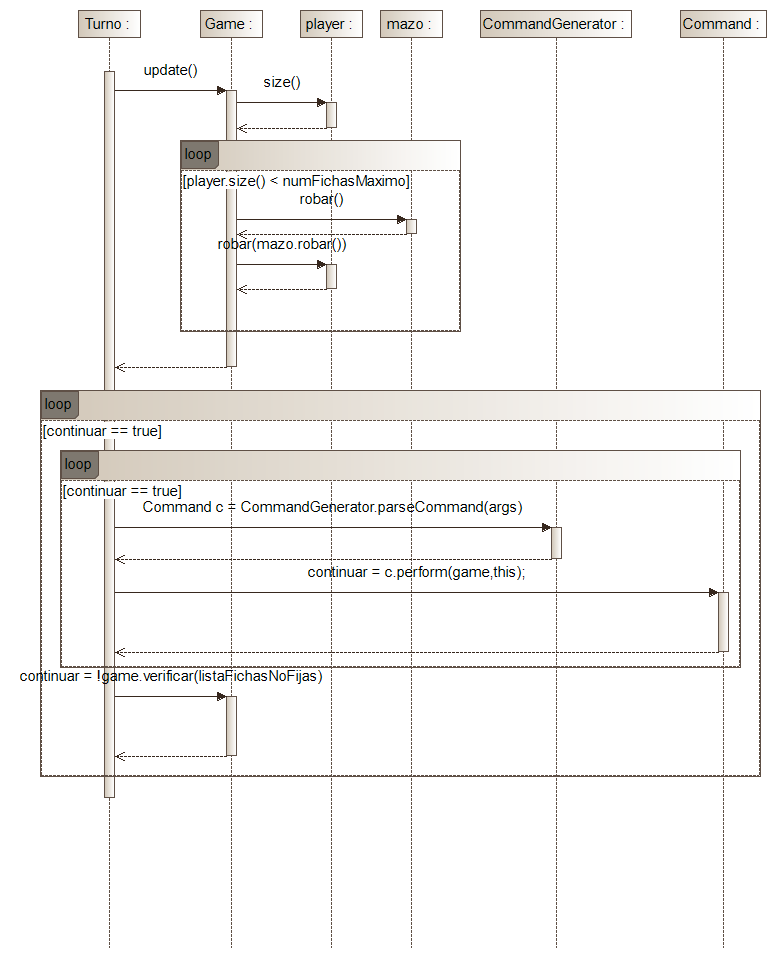


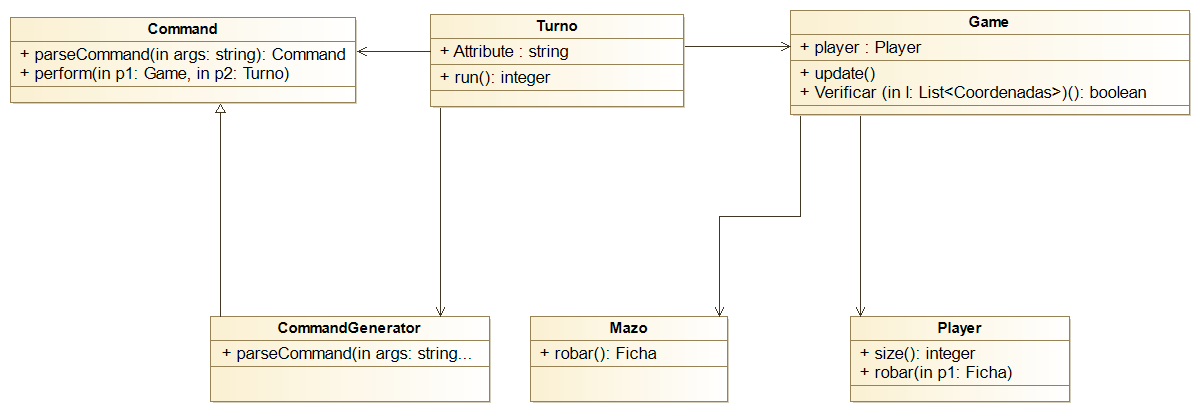
c. Explicación:

Se ha añadido un atributo a la clase casilla que indica si es posible o no poner en ella una ficha en ese momento (boolean disponible). Para manipular ese atributo, cada vez que se coloca o quita una ficha, se calcula dicho atributo para la casilla en la que se está colocando, y para las 4 adyacentes. En concreto, una casilla está disponible si existe alguna adyacente a ella que está llena y si la casilla está vacía.

15. No pasar de turno si no he verificado una palabra

a. Diagrama de Secuencia:

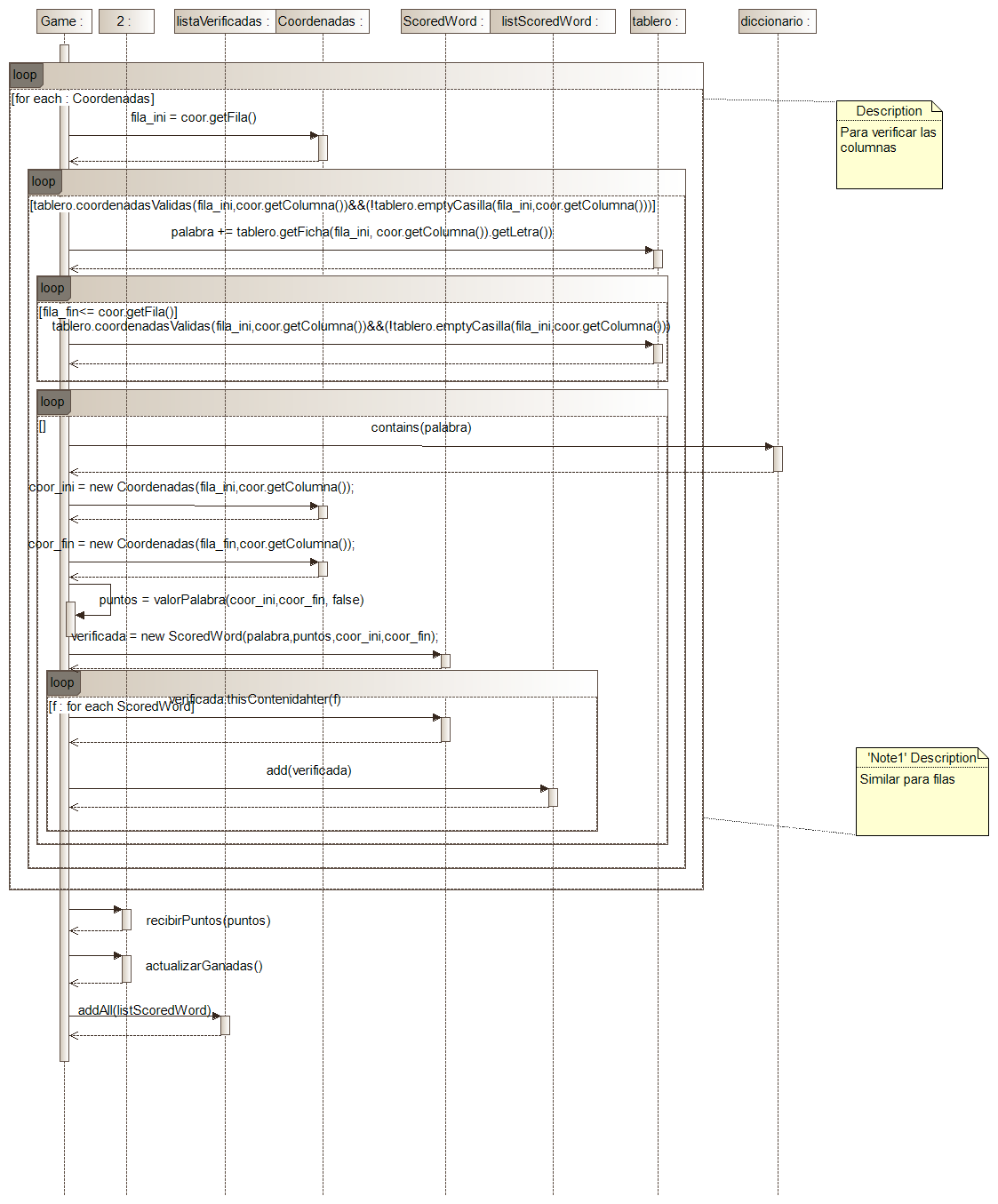


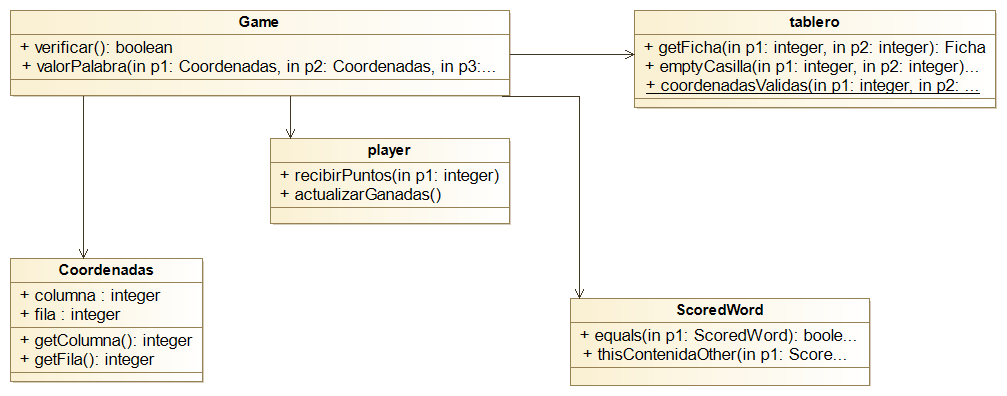
1. Diagrama de Clases:
2. Explicación:

Cuando el jugador introduce el comando de pasar de turno, la variable booleana

continuar se cambia a true si es posible pasar el turno. Entonces se llama al método verificar de game. Este método devuelve true si la listaFichasNoFija que le has pasado es vacia o si ha verificado alguna palabra (la lista ScoredWord no es vacia). Si devuelve true podemos pasar de turno saliendo del segundo bucle while. Si no continuar se pone a true y no podemos pasar de turno.

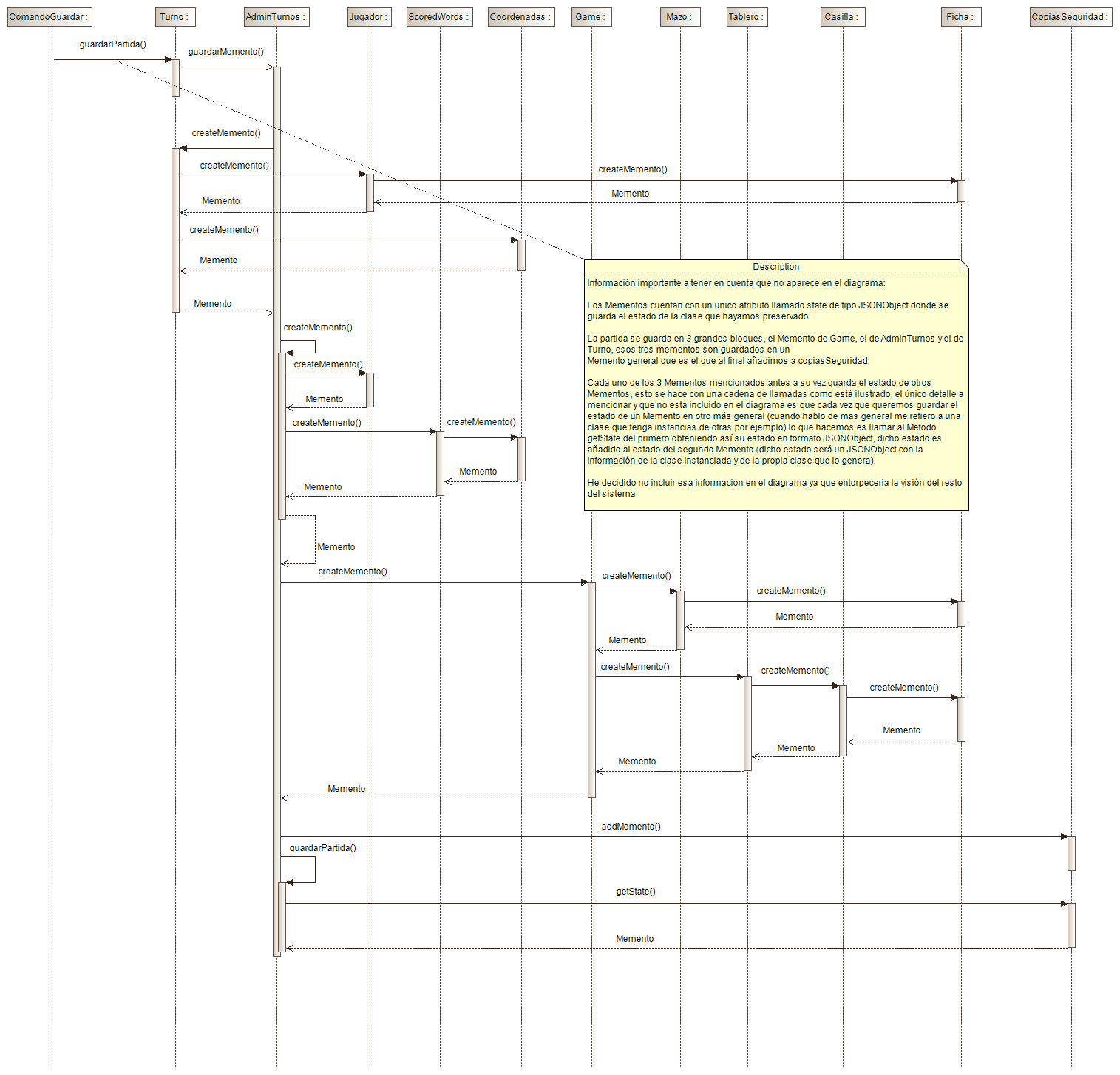
16. Verificar automáticamente

 a. Diagrama de Secuencia:

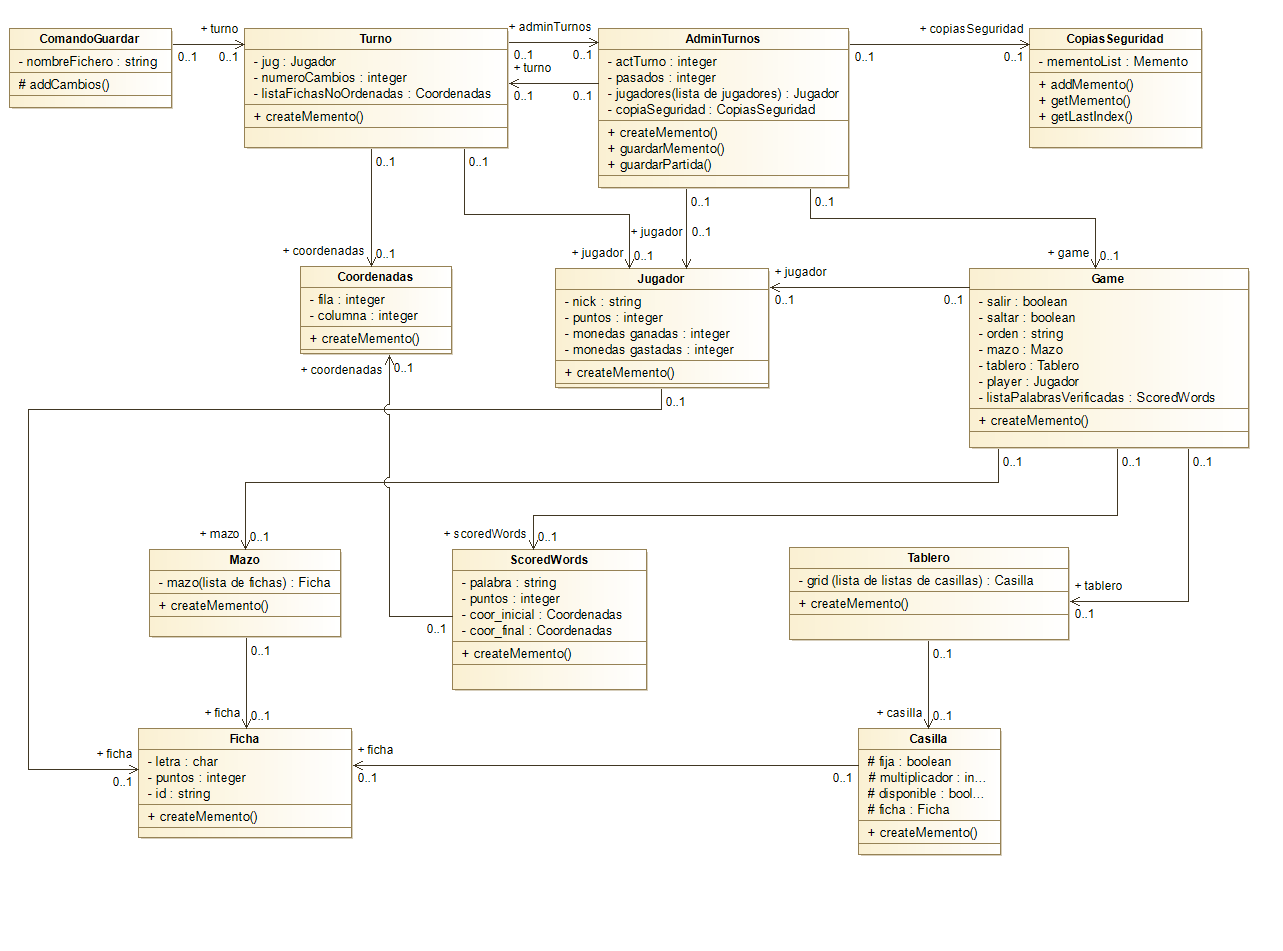
 b. Diagrama de Clases:

16. Guardar Partida, patrón Memento

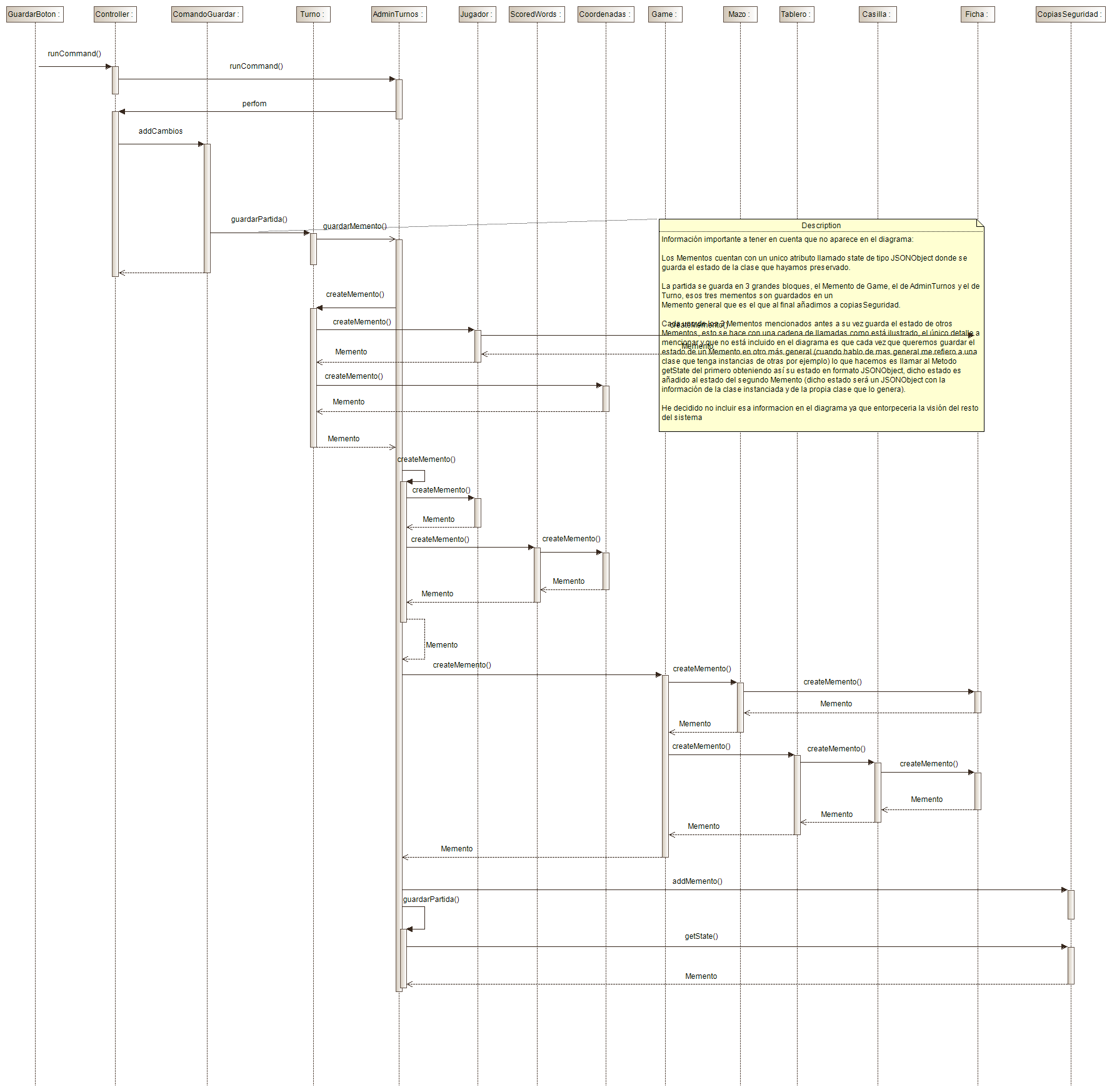
a. Diagrama de Secuencia:



b. Diagrama de Clases

17. Guardar Partida, patrón Memento vista

* 1. Diagrama de Secuencias



* 1. Diagrama de Clases

