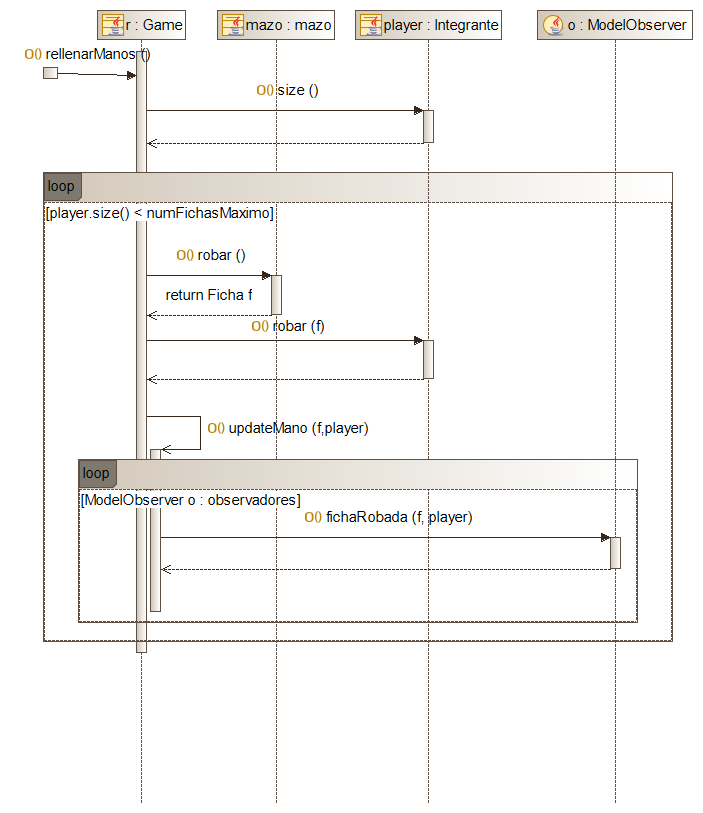
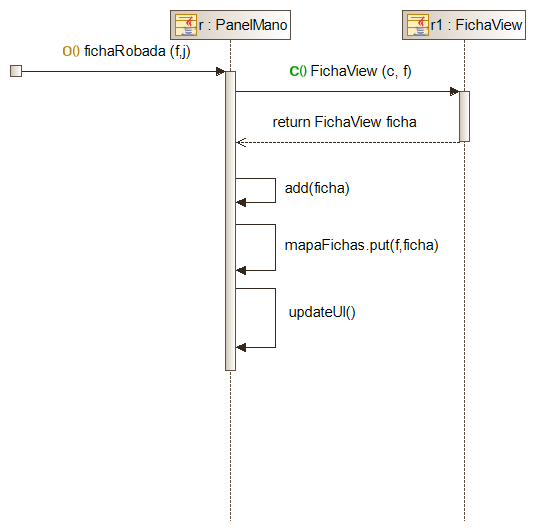
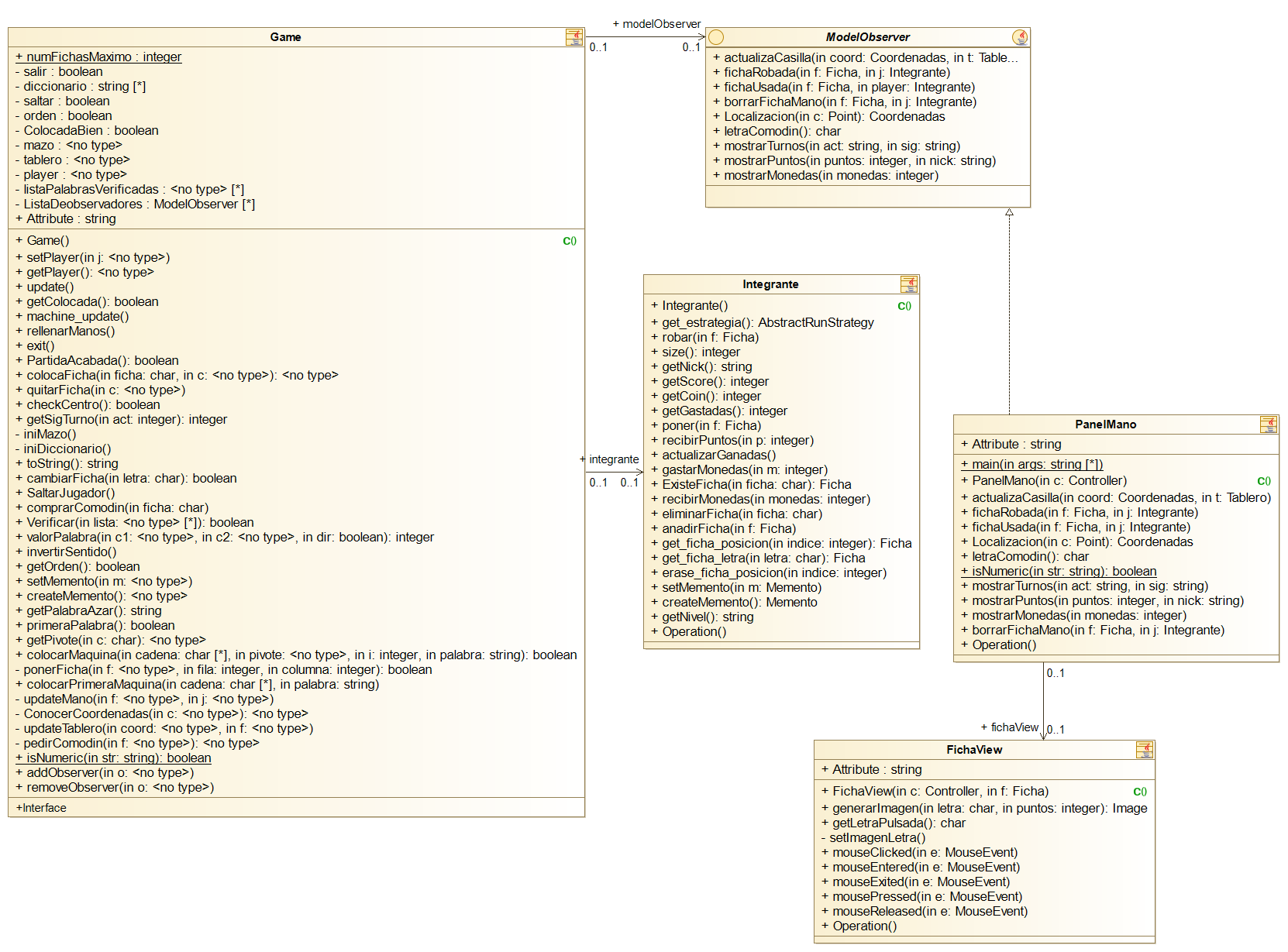
**SPRINT 4 DIAGRAMAS**

1. **Mostrar la mano del jugador en la GUI**
   1. **Diagramas de secuencia**

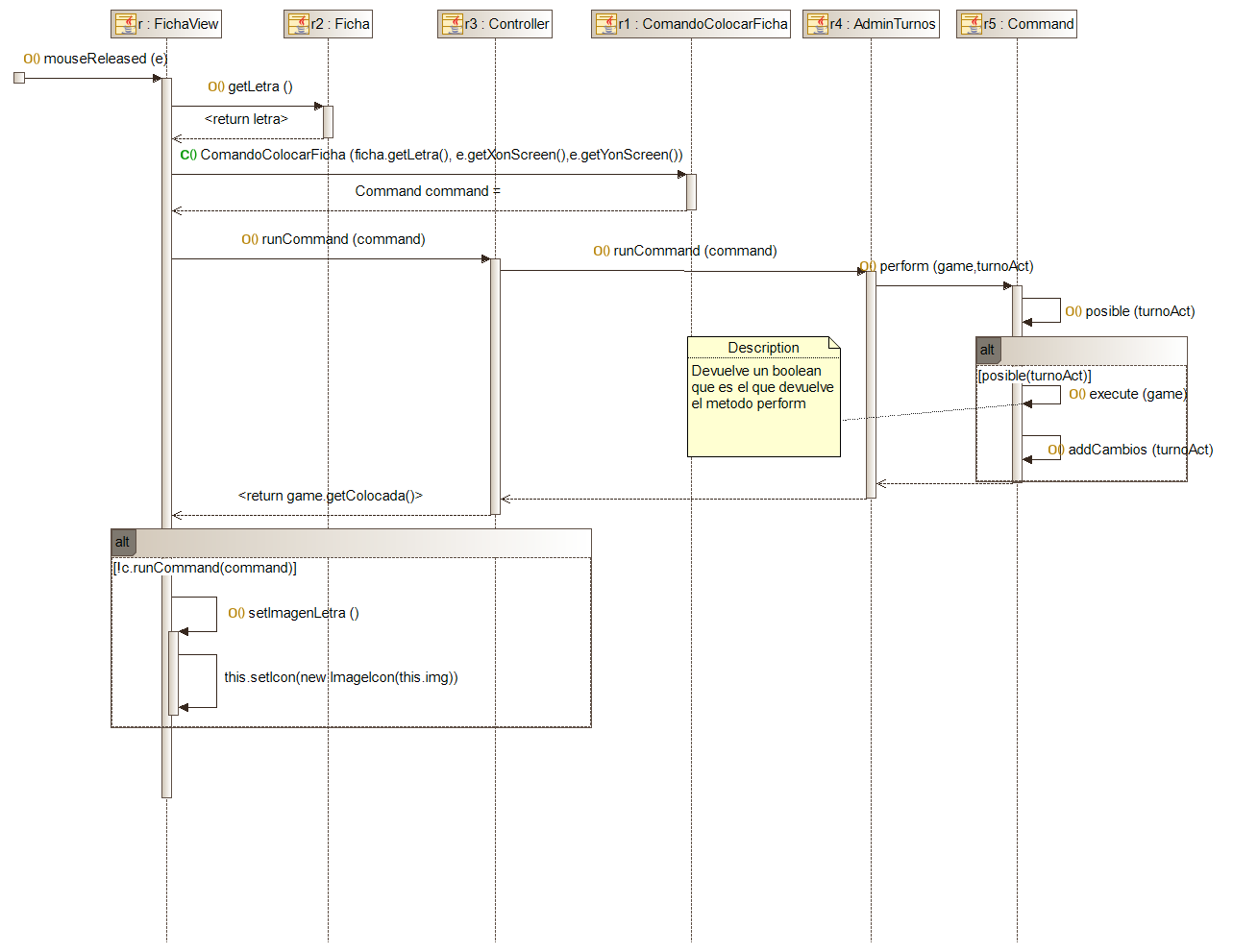
****

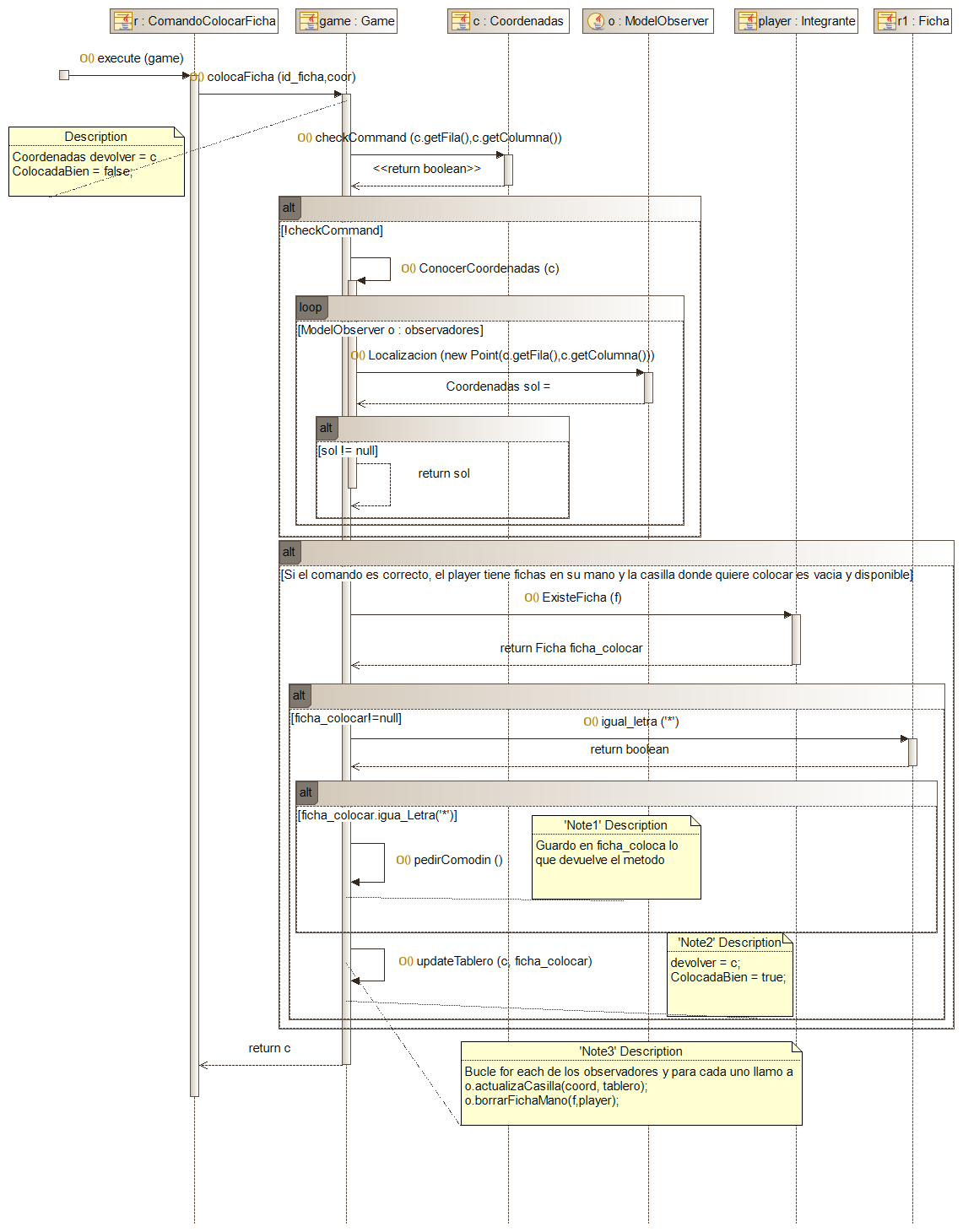


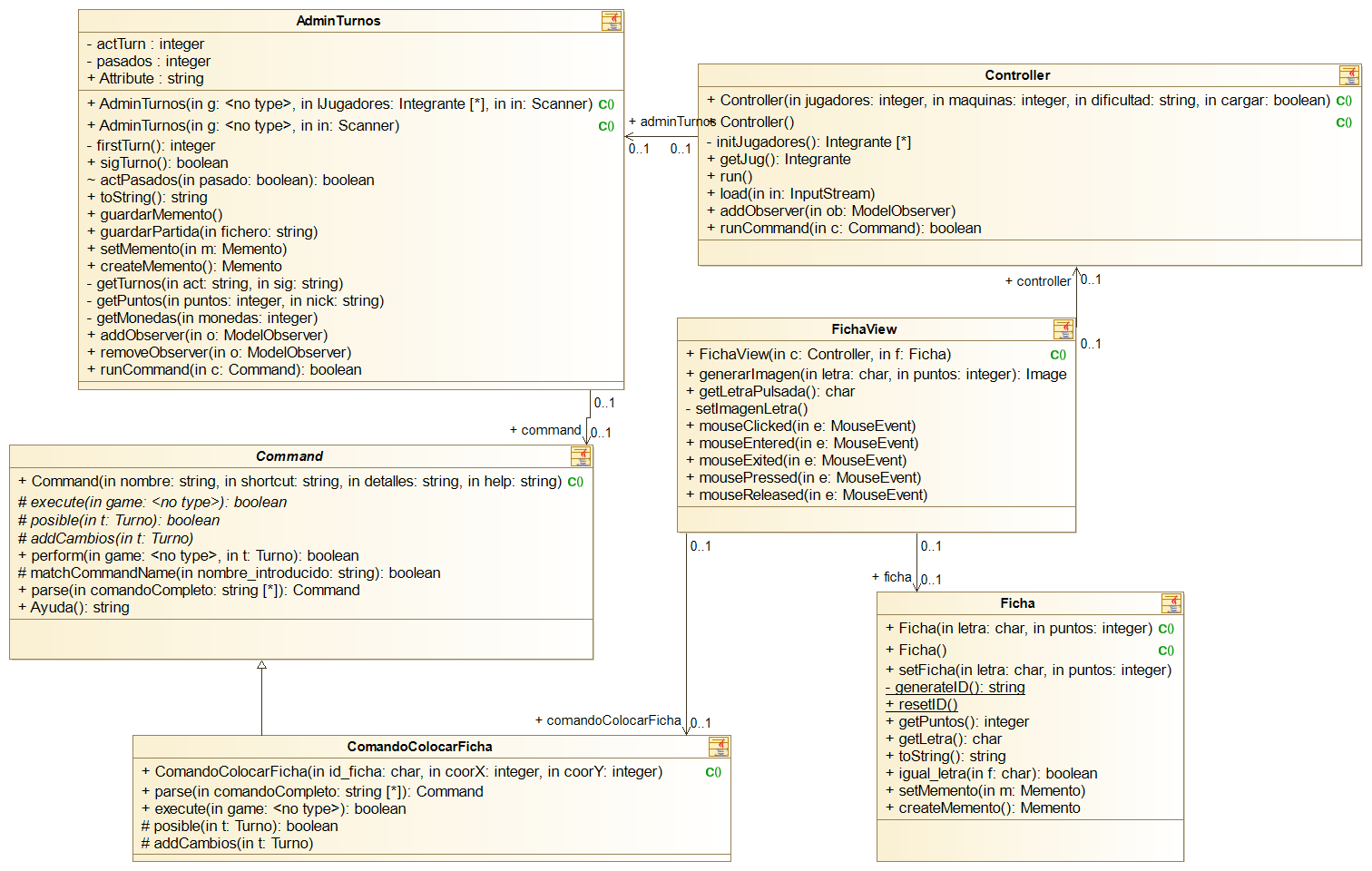
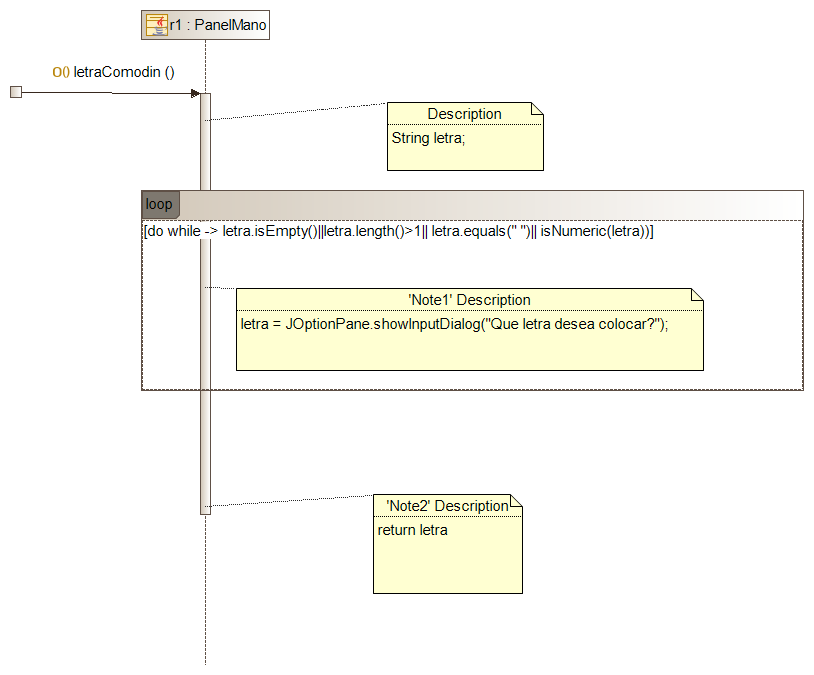
* 1. **Diagrama de clase**
  2. **Explicación**

Para mostrar la mano del jugador, hemos creado dos clases nuevas pertenecientes a la vista -> PanelMano (JPanel) y FichaView (JLabel). El PanelMano tiene de atributos el Controlador y un mapa<Ficha,FichaView>, además implementa la interfaz ModelObserver y por tanto pertenece a la lista de observadores del game (ahora implementa la interfaz observer). Al rellenar la mano del jugador nos quedamos con la ficha y se la pasamos a al PanelMano a través del método propio de los ModelObserver fichaRobada(Ficha f, Integrante j). Este método implementado en el PanelMano, crea una FichaView al que le pasa la Ficha en el constructor y la añade a su lista de Componentes. Crea un par Ficha-FichaView a partir de la Ficha dada y la FichaView construida y lo añade al mapa. Después llama a updateUI() para actualizar la GUI.

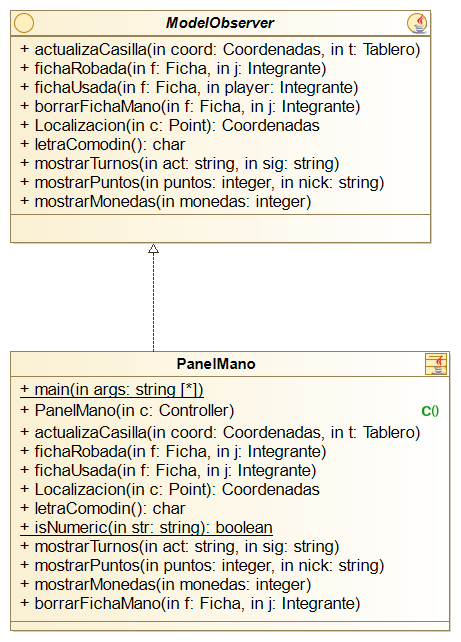
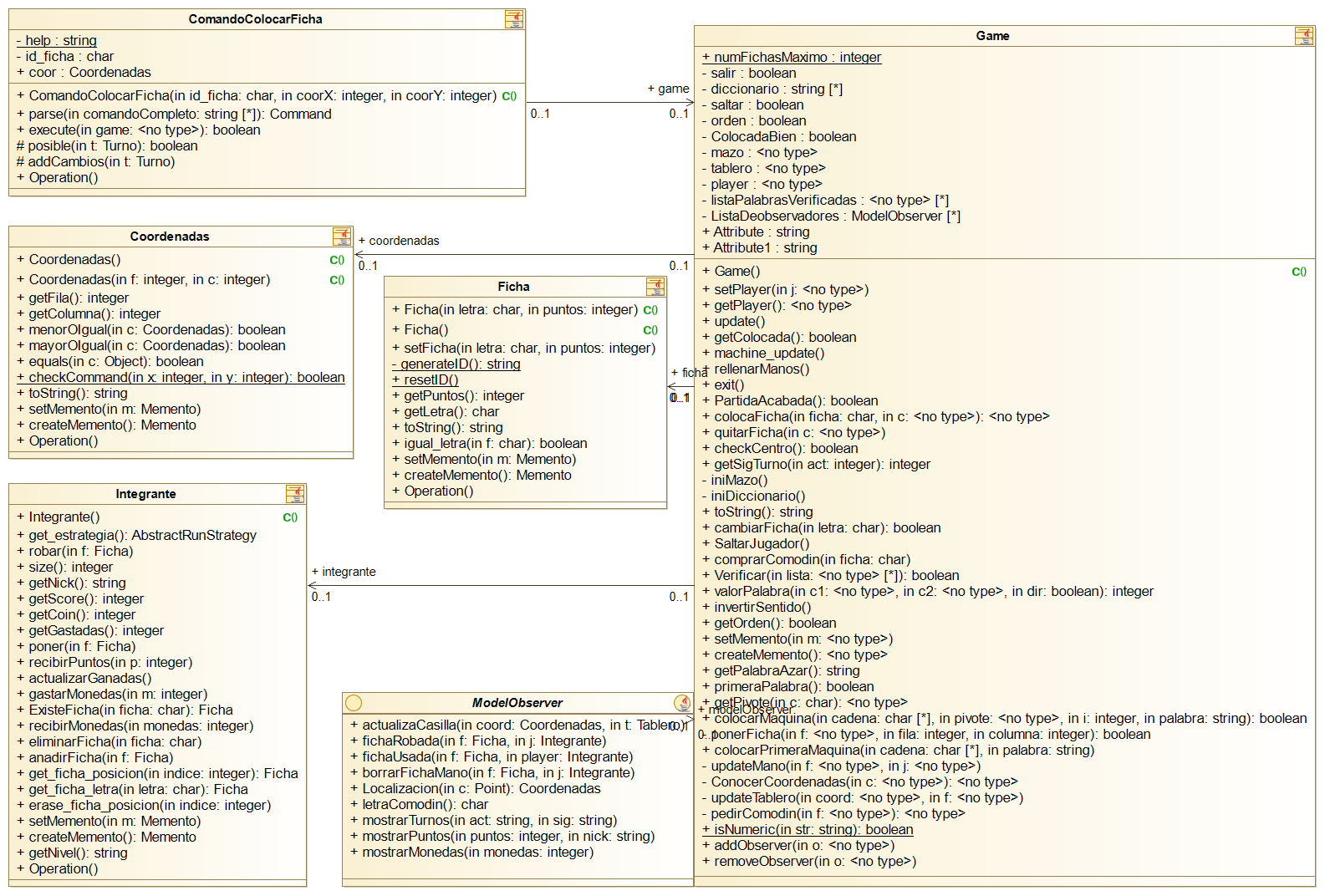
1. **Colocar Ficha en el Tablero en la GUI**
   1. **Diagramas de secuencia**







* 1. Diagrama de clase



* 1. **Explicación**

Para colocar una ficha en el tablero. Al presionar el ratón, la imagen de la FichaView se elimina y el cursor toma la imagen de la FichaView. Al soltar el ratón, se crea el comando colocar ficha al que le paso la letra de la ficha y las coordenadas en la pantalla del cursor, y se ejecuta. Si este no lo hace correctamente, la ficha vuelve a tener la imagen original.

Para ejecutarse el comando colocar ficha, se llama al método del Controller runCommand que recibe el comando como parámetro. Este a su vez llama al runConmmad del administrador de turnos, que recibe el comando como parámetro y llama al método perform del mismo. Este primero comprueba que sea posible ejecutar el comando y si lo es llama al execute del comando. Despues lo hará del addCambios.

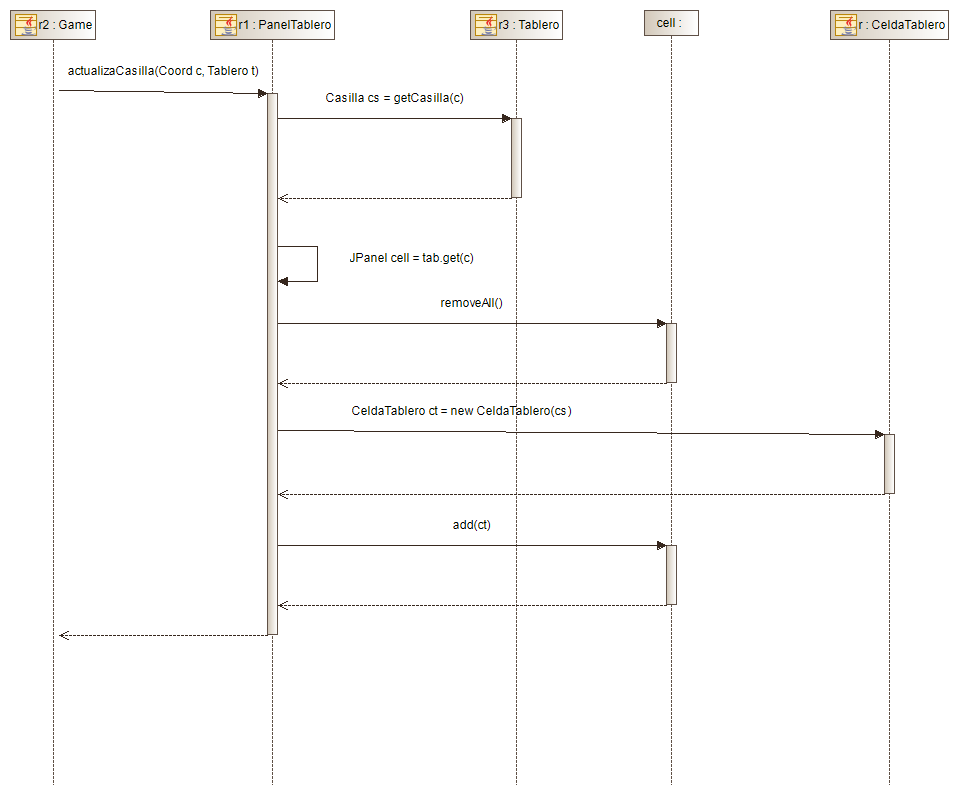
El execute de colocarficha es el que realiza la llamada pasandole las coordenadas y la letra de la ficha.

El colocarFicha del game lo primero que hace es poner a false un bool del game (colocadaBien)que indica si este metodo se realizó correctamente convertir las coordenadas a coordenadas del tablero llamando al Localizacion del ModelObserver. Despues comprueba que sean correctas y si lo son, después de realizar comprobaciones varias de la ficha para saber si existe en la mano, etc, compueba si es un comodín. Para ello llama al pedirComodin de los observadores que muestra un JOptionPane para preguntar al usuario por la letra del comodín. Cuando la introduce la guarda en una variable comprobando antes que sea correcta. Una vez conseguida la letra la devuelve al game que cambia la letra de la ficha a la recibida por el pedirComodin. Cuando el game tiene la ficha correspondiente, la añade al tablero y llama al actualizarCasilla de los modelObserver. Pone a true el colocadaBien.

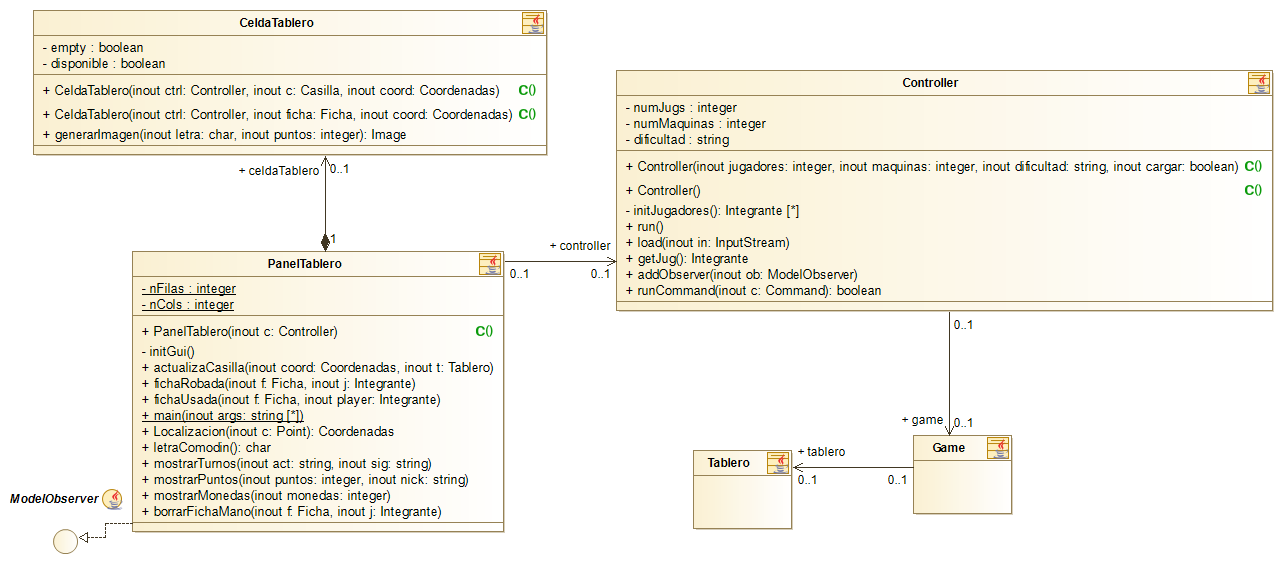
Una vez finalizado el método, devuelve las coordenadas.

**3. Ver el tablero en la GUI:**

**a. Diagrama de secuencia:**

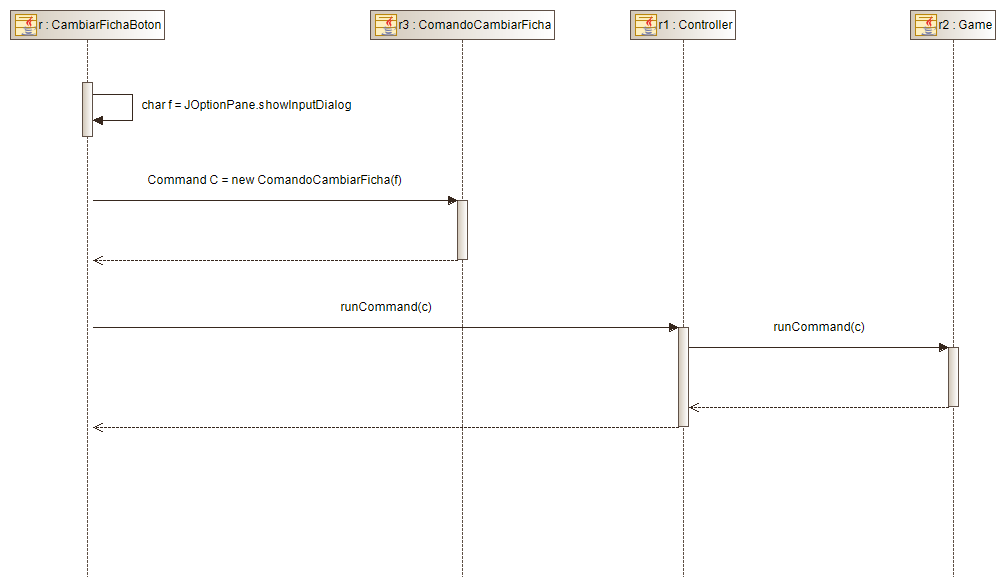


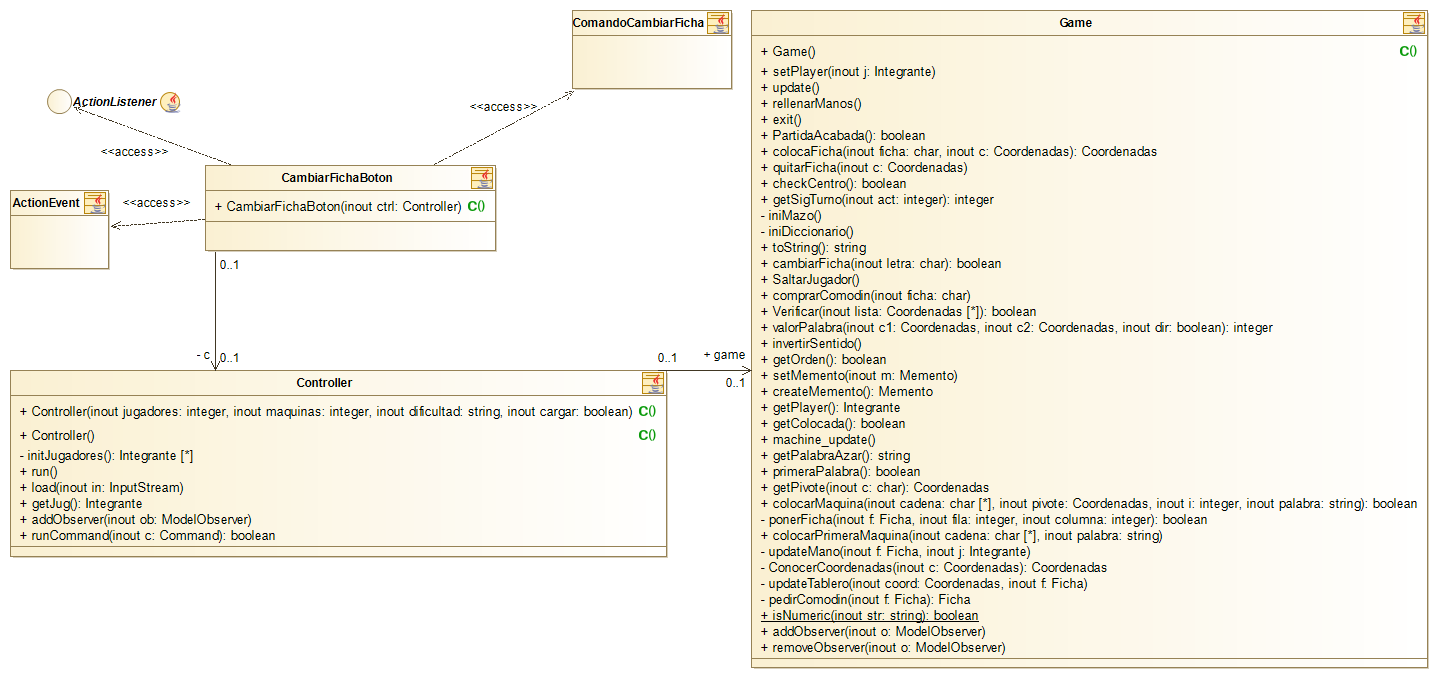
**b. Diagrama de Clases:**

**c. Explicación:**

Cuando en el modelo se produce un cambio en una casilla del tablero, se avisa a la vista por el método de ModelObserver ‘actualizaCasilla’, pasando por parámetro la coordenada de la casilla cambiada y el tablero de juego. A continuación, en el PanelTablero se extrae el Jpanel que contiene la representación de esa casilla , se vacía, y se vuelve a rellenar con una representación acorde a la nueva información del modelo, que se encuentra en la casilla del tablero.

**4. Cambiar Ficha desde la GUI**

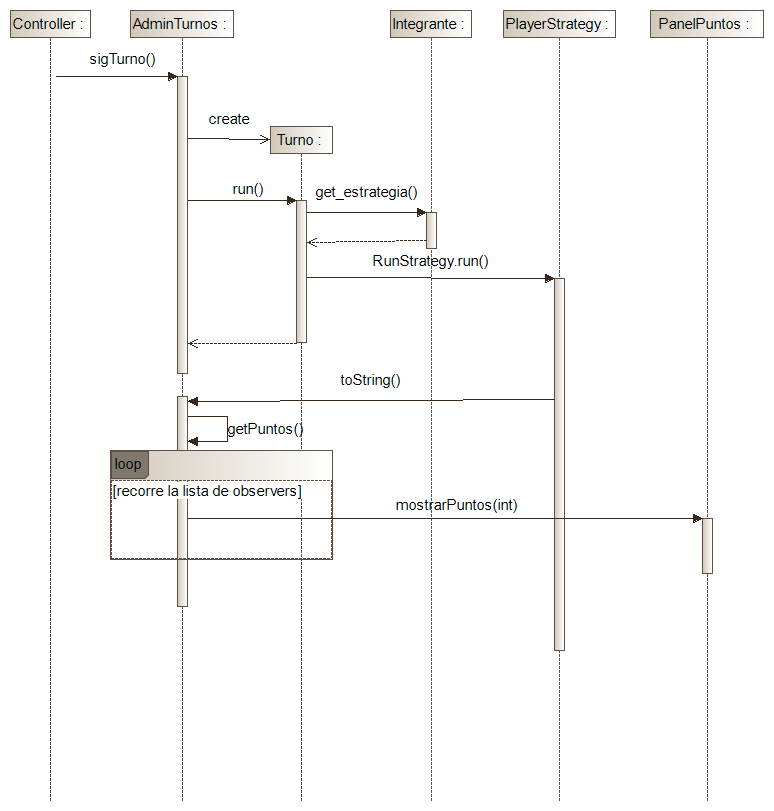
**a. Diagrama de Secuencia**

 **b. Diagrama de Clases**

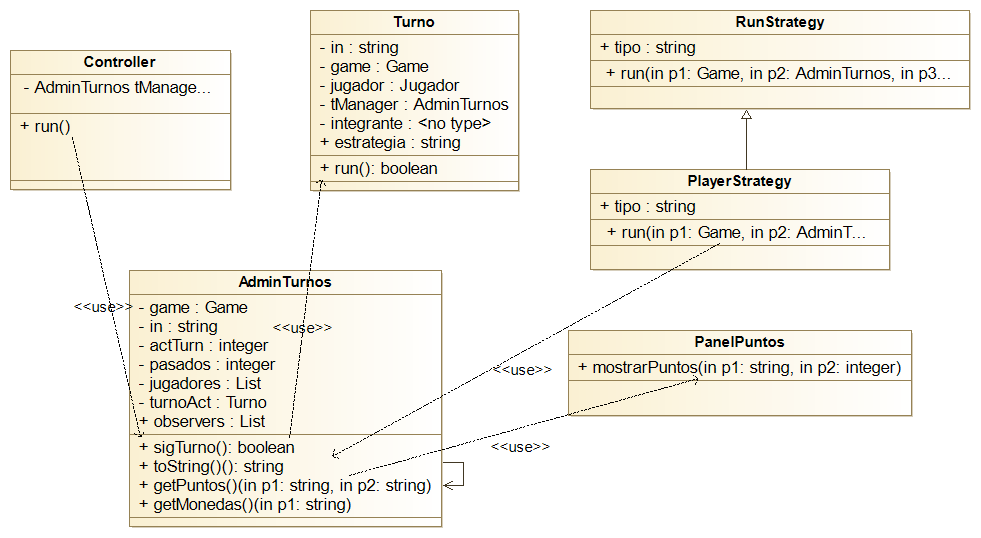
**c. Explicación:**

Al pulsar el botón de cambiar Ficha de la GUI, se muestra un Jdialog que pide la letra que quieres cambiar. Cuando la proporcionas y das a aceptar, se crea un nuevo comando CambiarFicha con la letra que das como parámetro y se pasa al controller, y del controller al AdminTurnos, donde se ejecuta el comando. Durante la ejecución, se avisa a la GUI de los cambios que se van produciendo.

1. **Mostrar la puntuación del jugador en la GUI**
   1. **Diagrama de secuencia**



* 1. **Diagrama de clases**

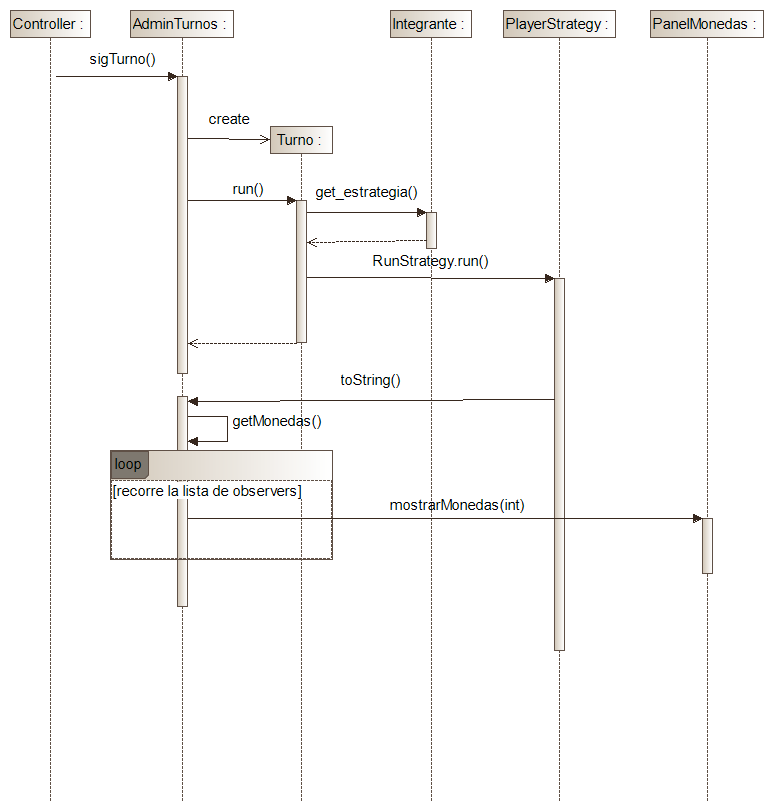
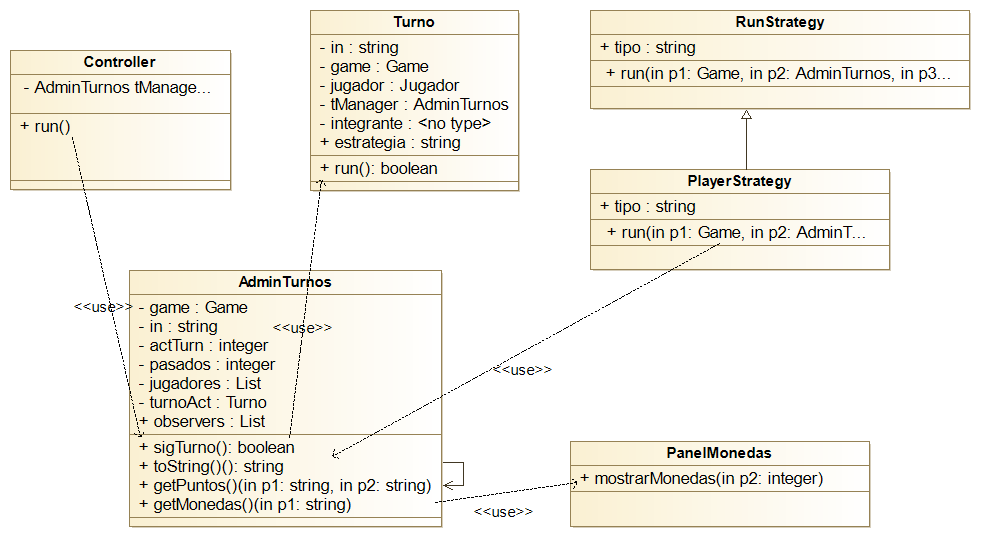
****

* 1. **Explicación**

Cada vez que se cambia de turno el Controller hace una llamada al método sigTurno(). En este método se crea un nuevo turno de la clase Turno para el jugador actual y se llama al método run() con el turno creado. El método run() de la clase Turno llama a su vez al método run() con la estrategia del jugador actual, que puede ser PlayerStrategy si es un jugador o bien EasyStrategy, MediumStrategy o HardStrategy si en el turno actual está jugando la máquina.

Estas clases Strategy extienden a la clase RunStrategy, de esta forma conseguimos diferenciar el método run() para cada caso (jugadores o máquina).

Los puntos del jugador junto a su nick se muestran cuando en el turno actual está jugando un jugador y se llama al método run() de la clase PlayerStrategy. Al inicio de este método se hace un println del AdminTurnos y al final del método toString() de la clase AdminTurnos se hace una llamada a otro método getPuntos() en el cual se recorre la lista de observers llamando al método mostrarPuntos(String nick, int puntos). Como en la lista de observers se encuentra el PanelPuntos, una vez lleguemos al método mostrarPuntos(String nick, int puntos) con dicho observer PanelPuntos se actualizará la información del nick y los puntos para mostrarlos por la GUI.

1. **Mostrar las monedas del jugador en la GUI**
   1. **Diagrama de secuencia**
   2. **Diagrama de clases**
   3. **Explicación**

En este caso, para mostrar las monedas del jugador el procedimiento es esencialmente el mismo que en el caso de mostrar la puntuación cambiando unas llamadas a unos métodos.

Las monedas del se muestran cuando en el turno actual está jugando un jugador y se llama al método run() de la clase PlayerStrategy. Al inicio de este método se hace un println del AdminTurnos y al final del método toString() de la clase AdminTurnos se hace una llamada a otro método getMonedas() en el cual se recorre la lista de observers llamando al método mostrarMonedas(int monedas). Como en la lista de observers se encuentra el PanelMonedas, una vez lleguemos al método mostrarMonedas(int monedas) con dicho observer PanelMonedas se actualizará la información de las monedas para mostrarlas por la GUI.