Práctica 5: Interfaz gráfico de usuario para los juegos de tablero

Fecha de entrega: 22 de abril de 2016 a las 16:00

Objetivo: GUI en Java utilizando Swing. Patrón de diseño Modelo-Vista-Controlador. Diseño Orientado a Objetos.

Antes de comenzar, crea un paquete practica5 y copia en él el fichero Main.java del paquete practica4. Todo el código de esta práctica debe estar definido dentro de practica5. Recuerda que está prohibido modificar ningún fichero del paquete basecode. Además, te recomendamos que no modifiques nada de practica4, excepto si vas a corregir algún error de programación.

1. Descripción general de la práctica

En esta práctica debes desarrollar e integrar varias vistas gráficas de los juegos *Connect-N*, *Tic-Tac-Toe*, *Advanced Tic-Tac-Toe* y *Ataxx* (nos referiremos a ellas como vistas de `ventana`). Debes utilizar el patrón de diseño MVC y además deberías utilizar un diseño que permita desarrollar una sola vista que después puedes adaptar a cada juego utilizando herencia o composición. En las clases de la asignatura veremos más reglas generales de diseño que puedes utilizar.

La vista `ventana` debería permitir dos modos de uso: (1) `ventana única`, en la que todos los jugadores utilizan la misma ventana; y (2) `múltiples ventanas`, en la que cada jugador tiene su propia ventana y cada ventana solo la puede utilizar un jugador. Por defecto se debe utilizar el modo `ventana única`. El modo `múltiples ventanas` debe utilizarse cuando el usuario proporciona la opción -m (o --multiviews) en la línea de órdenes. El interfaz gráfico de la vista `ventana` debe tener la misma información que el que se muestra en la figura 1 y debe proporcionar la funcionalidad que se detalla a continuación.

En esta vista, un jugador puede estar en cualquiera de los siguientes modos de juego: `MANUAL`, `RANDOM` o `INTELLIGENT`. Todos los jugadores comienzan en modo `MANUAL`, sea cual sea el modo que se haya especificado en la línea de órdenes. Los jugadores pueden cambiar su modo de juego durante la partida utilizando los *combo-boxes* del área "*Player*

Figura 1: La vista `ventana`: en la parte superior aparece en modo `ventana única`, y en la inferior en modo `múltiples ventanas`.

Modes" de la ventana. Los jugadores en modo `MANUAL` pueden hacer movimientos, cuando están en su turno, utilizando uno de los siguientes métodos:

1. Pulsando con el ratón sobre el tablero, de una forma que dependerá del juego al que se esté jugando;

2. Utilizando el botón *Random* (en el área "*Automatic Moves*") para hacer un movimiento aleatorio (mediante un jugador aleatorio); o

3. Utilizando el botón *Intelligent* (en el área "*Automatic Moves*" de la ventana) para realizar un movimiento inteligente (mediante un jugador inteligente).

Los jugadores en modo `MANUAL` no deben poder introducir un movimiento cuando no sea su turno. Los jugadores en modo `RANDOM` o `INTELLIGENT` deben jugar *automáticamente* cuando sea su turno –y no deben poder introducir un movimiento manual si están en un modo no manual. Por tanto, si todos los jugadores están en modo no manual, podemos sentarnos a ver cómo juegan automáticamente hasta que termina la partida (o hasta que modificamos el modo de juego de uno de los jugadores y llega su turno). Todos los movimientos deben realizarse en el modelo a través del controlador, mediante la correspondiente llamada a makeMove(...).

Los objetos Player que se deben utilizar para hacer movimientos `RANDOM` o `INTELLIGENT` deben pasarse a la clase de la vista mediante el método GameFactory.createSwingView(...), que a su vez los recibe de Main.java (veremos más detalles más abajo). Cuando uno de los jugadores es null (por ejemplo, cuando estos jugadores no están permitidos por el juego al

que se vaya a jugar), la vista se debe adaptar a esta situación y no debería permitir realizar movimientos con ese modo de juego (más abajo hay más detalles sobre esto).

En los siguientes apartados se explican los componentes de la vista de la figura 1 y se indican los requisitos que la implementación **debe satisfacer.**

⊷ Título de la ventana

El título de la ventana debe mostrar la descripción del juego al que se está jugando y, si está activado el modo `múltiples ventanas`, debe incluir el nombre del jugador (el identificador de la ficha) a la que pertenece la ventana.

⊷ *Status Messages*

Este área de la ventana debe mostrar la información del estado del juego, pedir a cada jugador que realice un movimiento, etc. Debe implementarse utilizando un componente JTextArea de solo lectura con un JScrollPane. A continuación se muestran ejemplos de la información que **debe mostrarse:**

- Cuando cambia el turno se debe mostrar el identificador de la ficha que debe jugar a continuación. Por ejemplo, "*Turn for X*". Si se está en modo `múltiples ventanas` también debe incluir la palabra "*You*" en la ventana del jugador al que le corresponde el turno. Por ejemplo, "*Turn for X (You!)*", o "*Turn for You X!*".
- Cuando sea el turno de jugadores en modo `MANUAL`, se debe mostrar información que les indique cómo realizar un movimiento. Por ejemplo, en el caso de *Connect-N* se puede mostrar un mensaje como el siguiente: "*Click on an empty cell*"; en el caso de *Ataxx* se puede mostrar "*Click on an origin piece*" y, cuando se ha seleccionado la ficha, "*Click on the destination position*".
- Cuando termina el juego se debe mostrar el mensaje "*Game Over!*", el estado de finalización del juego ("*Stopped*", "*Won*" o "*Draw*") y el nombre del ganador (si hay ganador).

Puedes mostrar cualquier otro mensaje que creas necesario. Por ejemplo, cuando se cambia el modo de juego, cuando se inicia o termina un movimiento, etc.

⊷ *Player Information*

Este área de la ventana está formado por una tabla de solo lectura (implementada utilizando la clase JTable con un JScrollPane) que contiene la siguiente información de cada jugador:

1. El identificador de la ficha;
2. El modo de juego —en el modo `múltiples ventanas` solo se debe mostrar el modo de juego del jugador al que pertenece la ventana (no se sabe el modo de los demás jugadores); y
3. El contador de fichas, si board.getPieceCount(piece) devuelve un valor distinto de null (por ejemplo, en *Advanced Tic-Tac-Toe*).

El color de fondo de cada línea de la tabla debe ser el color de la ficha correspondiente (más sobre esto a continuación).

➺ *Piece Colors*

Este área de la ventana permite cambiar el color de todas las fichas. Estos colores se deben utilizar cuando se dibuja el tablero. Se debe permitir cambiar el color de las fichas de los jugadores (por ejemplo, no es obligatorio permitir cambiar el color de los obstáculos de *Ataxx*). Cuando se cambia un color, se debe reflejar inmediatamente en el tablero y en el área "*Players Information*" de la ventana. Observa que en el caso de `ventana única` esto solo afecta a los colores de la ventana local. Por ejemplo, en la parte inferior de la figura 1, cada ventana usa colores diferentes para el mismo juego.. Esta parte debería implementarse usando JColorChooser (más información en clase).

➺ *Player Modes*

Este área de la ventana permite cambiar el modo de juego de los distintos jugadores durante la partida. En el modo `ventana única` solo se puede cambiar el modo de juego del jugador al que pertenece la ventana – en este caso, el primer *combo-box* solo debe contener el identificador del jugador. Estos componentes siempre deben estar activos para permitir el cambio del modo de juego de los jugadores en cualquier momento de la partida. Cuando termina la partida se pueden deshabilitar y habilitarlos de nuevo cuando se reinicia una nueva partida. El cambio del modo de juego de un jugador se debe reflejar inmediatamente en la tabla de "*Player Information*" y el jugador debe jugar en ese modo la próxima vez que le toque el turno.

IMPORTANTE: Si los jugadores de los modos `RANDOM` e `INTELLIGENT` no están disponibles (los objetos que recibe la vista en el método GameFactory.createSwingView(...) son null), este área de la ventana no debe estar disponible en la vista. Si uno de los dos es null, no debe mostrarse la opción correspondiente en el *combo-box*.

➺ *Automatic Moves*

Los jugadores en modo `MANUAL` pueden utilizar los botones de este área de la ventana para realizar un *único* movimiento aleatorio o inteligente.

IMPORTANTE: Como en el caso anterior, si los jugadores de los modos `RANDOM` e `INTELLIGENT` no están disponibles (los objetos que recibe la vista en el método GameFactory.createSwingView(...) son null), este área de la ventana no debe estar disponible en la vista. Si uno de los dos es null, no debe mostrarse el botón correspondiente en la vista.

DEBE ESTAR DESHABILITADO: durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.

➺ *Restart Button*

Este botón debe aparecer solo en el modo `ventana única` y puede utilizarse para reiniciar la partida (llamando al método restart() del controlador).

DEBE ESTAR DESHABILITADO: durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.

⊷ *Quit Button*

Este botón permite salir del juego. Cuando se pulsa sobre él, debe aparecer un *dialog box* que solicite al usuario confirmación. Antes de salir, debe pararse el juego con el método stop() del controlador.

DEBE ESTAR DESHABILITADO: durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.

⊷ Mensajes de error

Los mensajes de error, como los recibidos mediante el método GameObserver.onError(...), deben mostrarse utilizando *Dialog boxes.* En el modo `múltiples ventanas` los errores que ocurren durante la ejecución de un movimiento solo deben mostrarse en la ventana del jugador que hizo el movimiento.

⊷ Otros requisitos

A continuación se indican algunos requisitos que no corresponden a ninguno de los apartados anteriores (puede haber más que se indiquen en clase):

1. En *Ataxx* y *Advanced Tic-Tac-Toe*, después de seleccionar una ficha de origen el usuario puede cancelar la selección utilizando, por ejemplo, el botón derecho del ratón sobre el tablero. Además, si después de seleccionar la ficha de origen se cambia el modo de juego, se debe cancelar la selección.

2. En *Advanced Tic-Tac-Toe* hay dos tipos de movimiento, que cambian durante el juego. Al principio los jugadores deben ir colocando fichas en el tablero, como en *Tic-Tac-Toe*, hasta que se quedan sin fichas. En cuanto un jugador se queda sin fichas, puede continuar jugando, pero en lugar de añadir fichas debe mover las fichas que ya están en el tablero a otras posiciones que estén libres (comprueba cómo funciona en el modo de consola).

3. No está permitido utilizar herramientas de generación automática de interfaces gráficos de usuario (como *NetBeans*), debes escribir todo el código.

2. Modificación de las factorías de juego

La creación de la vista se debe realizar mediante una llamada al método createSwingView de la subclase de GameFactory que corresponda. Por tanto, es necesario modificar las factorías de todos los juegos existentes. Sin embargo, no se debe modificar ningún fichero del paquete basecode. Para resolver este problema, debes hacer lo siguiente para cada juego (lo explicamos para *Connect-N*):

1. crear un paquete nuevo practica5.connectn;

2. crear una clase practica5.connectn.ConnectNFactoryExt que extienda ConnectNFactory;

3. definir el método createSwingView en ConnectNFactoryExt, sobrescribiendo el que está definido en ConnectNFactory;

4. modificar Main.java para que utilice ConnectNFactoryExt en lugar de ConnectNFactory.

Es recomendable hacer esto incluso para el juego *Ataxx* que has creado en la práctica 4 para evitar la modificación del código de practica4 (excepto los errores que hayas corregido en este código).

3. Modificación de Main.java

En Main.java debes modificar el método startGame para que se permita el modo de vista `ventana`. Debes reemplazar la excepción que se lanza en el caso de elegir el modo ventana por el código que corresponda. Ten en cuenta que el *parser* de la línea de órdenes ya procesa las opciones -v y -m. Para crear una vista `ventana` en el caso de `ventana única` puedes hacer lo siguiente:

```
gameFactory.createSwingView(g, c, null,
                            gameFactory.createRandomPlayer(),
                            gameFactory.createAIPlayer(aiPlayerAlg));
```

Para crear una vista `ventana` para el caso `múltiples ventanas` utiliza lo siguiente:

```
for (Piece p : pieces) {
   gameFactory.createSwingView(g, c, p,
                               gameFactory.createRandomPlayer(),
                               gameFactory.createAIPlayer(aiPlayerAlg));
}
```

Fíjate que el atributo aiPlayerAlg corresponde al algoritmo que utilizará el jugador inteligente y actualmente tiene el valor null. Esto lo modificaremos en la práctica 6.

4. Entrega de la práctica

La práctica debe entregarse en un único archivo comprimido utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha y hora indicada en la cabecera de la práctica. El fichero debe tener al menos el siguiente contenido:

- Directorio src con el código fuente del programa.
- Fichero alumnos.txt donde se indicará el nombre de los componentes del grupo.
- Directorio doc con la documentación generada automáticamente sobre el código que has implementado.