UNIVERSIDAD ORT

Obligatorio II

Programación II

Grupo M2B

24/11/2014

Constanza Curzio 178606

Agustín Verdera 182584

Profesores: Gabriel Olivera – Gonzalo Guadalupe

Contenido

Diagrama de clases UML:

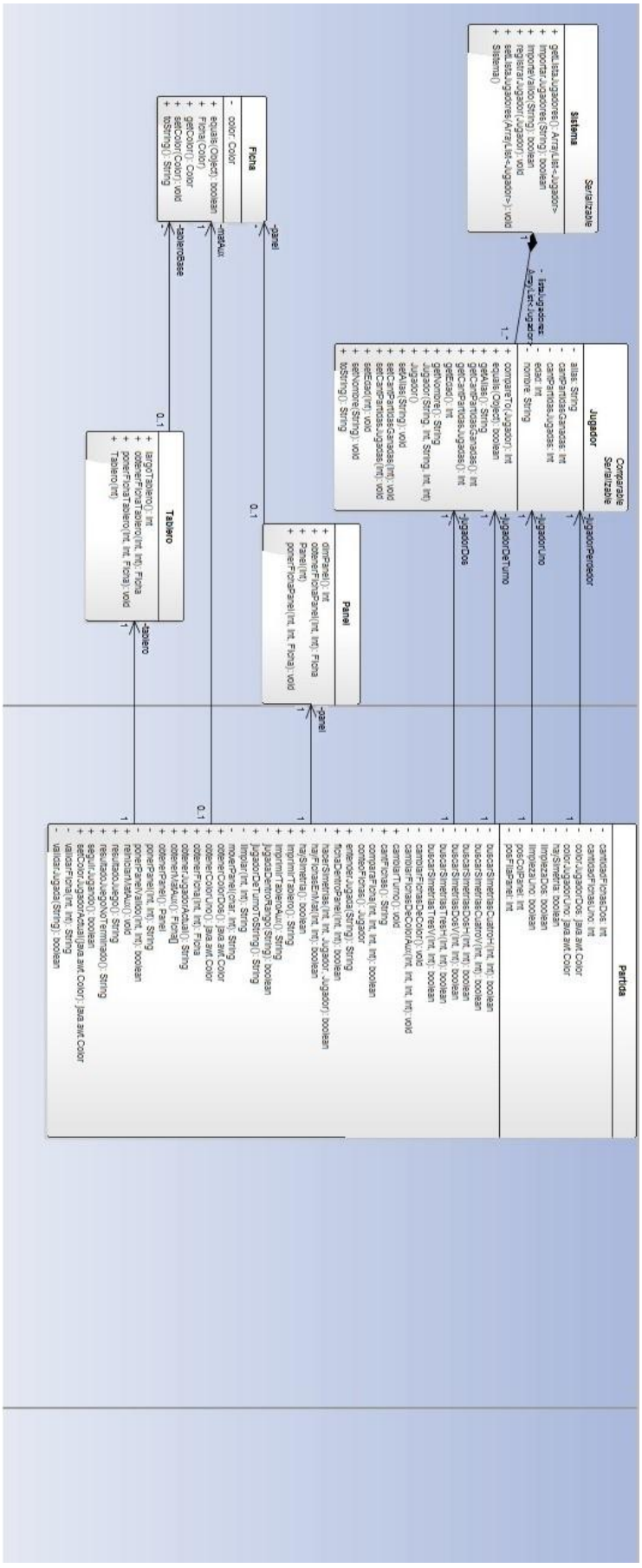


Tabla de datos de prueba:

Pruebas del menú principal

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en el botón de Registrar Jugador.	Ingresa a la ventana Registrar Jugador.	Ingresa a la ventana Registrar Jugador.	OK
Clic en el botón Jugar a Perfección.	Ingresa a la ventana de opciones del juego.	Ingresa a la ventana de opciones del juego.	OK
Clic en el botón Consulta de Jugadores.	Ingresa a la ventana mostrando la lista de jugadores.	Ingresa a la ventana mostrando la lista de jugadores.	OK
Clic en el botón Cargar Jugadores.	Ingresa a la ventana cargar jugadores.	Ingresa a la ventana cargar jugadores.	OK
Clic en el botón ¿Cómo Jugar?	Ingresa a la ventana como jugar.	Ingresa a la ventana como jugar.	OK
Clic en el botón Salir.	Se cierra el programa.	Se cierra el programa.	OK

Registrar Jugador

Nombre del jugador:

Prueba	Resultado esperado	Resultado obtenido	Resumen
“Fernando”	Acepta el nombre.	Acepta el nombre.	OK
“”	Mensaje de error “Ingrese un nombre válido”.	Mensaje de error “Ingrese un nombre válido”.	OK
“Gustavo”	Acepta el nombre.	Acepta el nombre.	OK
“23”	Acepta el nombre.	Acepta el nombre.	OK

Alias del Jugador:

Prueba	Resultado esperado	Resultado obtenido	Resumen
“roro”	Acepta el alias.	Acepta el alias.	OK
“tito”	Acepta el alias.	Acepta el alias.	OK
“”	Ingrese un alias válido.	Ingrese un alias válido.	OK
“17”	Acepta el alias.	Acepta el alias.	OK
“tito”	Mensaje error: “Alias ya existente. Ingrese un nuevo alias”.	Mensaje error: “Alias ya existente. Ingrese un nuevo alias”.	OK

Edad jugador:

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error: “Ingrese una edad válida”.	Mensaje error: “Ingrese una edad válida”.	OK
100	Mensaje error: “Ingrese una edad válida”.	Mensaje error: “Ingrese una edad válida”.	OK
14	Acepta la edad.	Acepta la edad.	OK
59	Acepta la edad.	Acepta la edad.	OK
1	Acepta la edad.	Acepta la edad.	OK
99	Acepta la edad.	Acepta la edad.	OK

Jugar a Perfección

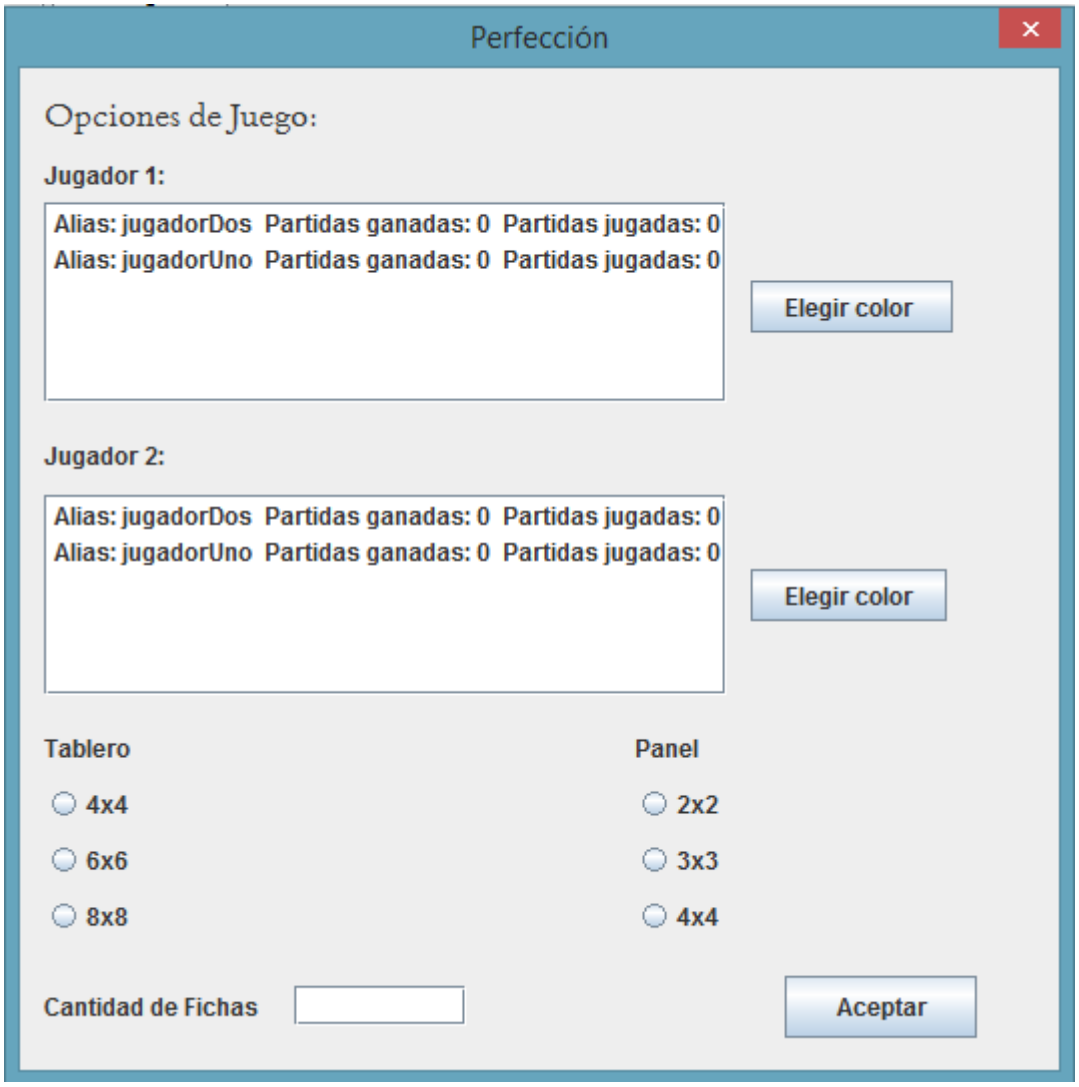
Sin ningún jugador registrado:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en el botón Jugar a Perfección.	Mensaje error: “Deben existir al menos dos jugadores.”	Mensaje error: “Deben existir al menos dos jugadores.”	OK

Con un jugador registrado:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en el botón Jugar a Perfección.	Mensaje error: “Deben existir al menos dos jugadores.”	Mensaje error: “Deben existir al menos dos jugadores.”	OK

Con dos jugadores registrados.



Seleccionar Jugador: [1]

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en Jugador 1	Se actualiza la lista del jugador 2 mostrando solo el jugador dos.	Se actualiza la lista del jugador 2 mostrando solo el jugador dos.	OK
Clic en Jugador 2	Se actualiza la lista del jugador 1 mostrando solo el jugador uno.	Se actualiza la lista del jugador 1 mostrando solo el jugador uno.	OK

[1] De vez en cuando, al hacer clic sobre un jugador, el mismo no queda seleccionado por lo que no lo actualiza en la lista. Esto se debe a un error interno de java.

Elegir color:

(Cada jugador tiene su color por defecto, Jugador 1 tiene color blanco y Jugador 2 tiene color negro).

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en Elegir Color.	Ingresa a la ventana color mostrando la paleta de colores.	Ingresa a la ventana color mostrando la paleta de colores.	OK

Jugador 1 elige un color:

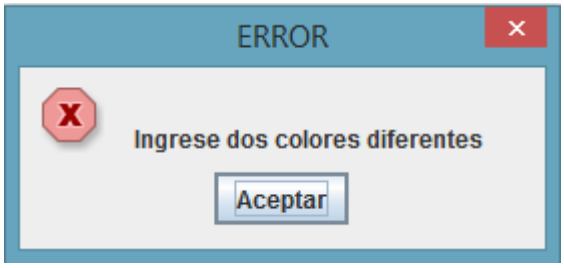
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en uno de los colores y luego clic en aceptar.	El jugador 1 tiene un color de la paleta de colores.	El jugador 1 tiene un color de la paleta de colores.	OK
Clic en Cancelar sin elegir un color.	Vuelve al color por defecto.	Vuelve al color por defecto.	OK
Clic en Aceptar sin elegir un color.	Mensaje error: “Seleccione un color diferente”.	Mensaje error: “Seleccione un color diferente”.	OK

El jugador 2 elige un color:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en uno de los colores y luego clic en aceptar.	El jugador 2 tiene un color de la paleta de colores.	El jugador 2 tiene un color de la paleta de colores.	OK
Clic en Cancelar sin elegir un color.	Vuelve al color por defecto.	Vuelve al color por defecto.	OK
Clic en Aceptar sin elegir un color.	Mensaje error: “Seleccione un color diferente”.	Mensaje error: “Seleccione un color diferente”.	OK

Si ambos jugadores eligen el mismo color:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en la paleta de colores eligiendo el mismo color.	Mensaje error: “Ingrese dos colores diferentes”.	Mensaje error: “Ingrese dos colores diferentes”.	OK



Tablero:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en Tablero 4X4.	Selecciona tablero 4x4 y panel de 2x2.	Selecciona tablero 4x4 y panel de 2x2.	OK
Clic en Tablero 6x6.	Selecciona tablero 6x6. Y deja las opciones habilitadas para panel de 2x2 y 3x3.	Selecciona tablero 6x6. Y deja las opciones habilitadas para panel de 2x2 y 3x3.	OK
Clic en Tablero 8x8.	Selecciona tablero 8x8. Y deja las opciones habilitadas para panel de 2x2, 3x3 y 4x4.	Selecciona tablero 8x8. Y deja las opciones habilitadas para panel de 2x2, 3x3 y 4x4.	OK

Panel:

Si selecciona tablero de 4x4, se ingresa panel de 2x2.

Tablero seleccionado en 6x6:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en panel 2x2	Selecciona panel 2x2.	Selecciona panel 2x2.	OK
Clic en panel 3x3	Selecciona panel 3x3.	Selecciona panel 3x3.	OK

Tablero seleccionado en 8x8:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en panel 2x2	Selecciona panel 2x2.	Selecciona panel 2x2.	OK

Clic en panel 3x3	Selecciona panel 3x3.	Selecciona panel 3x3.	OK
Clic en panel 4x4	Selecciona panel 4x4.	Selecciona panel 4x4.	OK

Cantidad de fichas:

Tablero 4x4 y Panel 2x2.

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:" Ingrese una cantidad de fichas de 1 a 10".	Mensaje error: "Ingrese una cantidad de fichas de 1 a 10".	OK
11	Mensaje error:" Ingrese una cantidad de fichas de 1 a 10".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 10".	OK
100	Mensaje error:" Ingrese una cantidad de fichas válido".	Mensaje error:" Ingrese una cantidad de fichas válido".	OK
10	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

Tablero 6x6 y Panel 2x2.

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:" Ingrese una cantidad de fichas de 1 a 20".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 20".	OK
21	Mensaje error:" Ingrese una cantidad de fichas de 1 a 20".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 20".	OK
100	Mensaje error:" Ingrese una cantidad de fichas válido".	Mensaje error:" Ingrese una cantidad de fichas válido".	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
20	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

Tablero 6x6 y Panel 3x3.

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:" Ingrese una cantidad de fichas de 1 a 22".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 22".	OK
23	Mensaje error:" Ingrese una cantidad de fichas de 1 a 22".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 22".	OK
100	Mensaje error:" Ingrese una cantidad de fichas válido".	Mensaje error:" Ingrese una cantidad de fichas válido".	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
22	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

Tablero 8x8 y Panel 2x2.

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:" Ingrese una cantidad de fichas de 1 a 34".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 34".	OK
35	Mensaje error:" Ingrese una cantidad de fichas de 1 a 34".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 34".	OK
100	Mensaje error:" Ingrese una cantidad de fichas válido".	Mensaje error:" Ingrese una cantidad de fichas válido".	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
34	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

Tablero 8x8 y Panel 3x3.

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:" Ingrese una cantidad de fichas de 1 a 37".	Mensaje error:" Ingrese una cantidad de fichas de 1 a 37".	OK

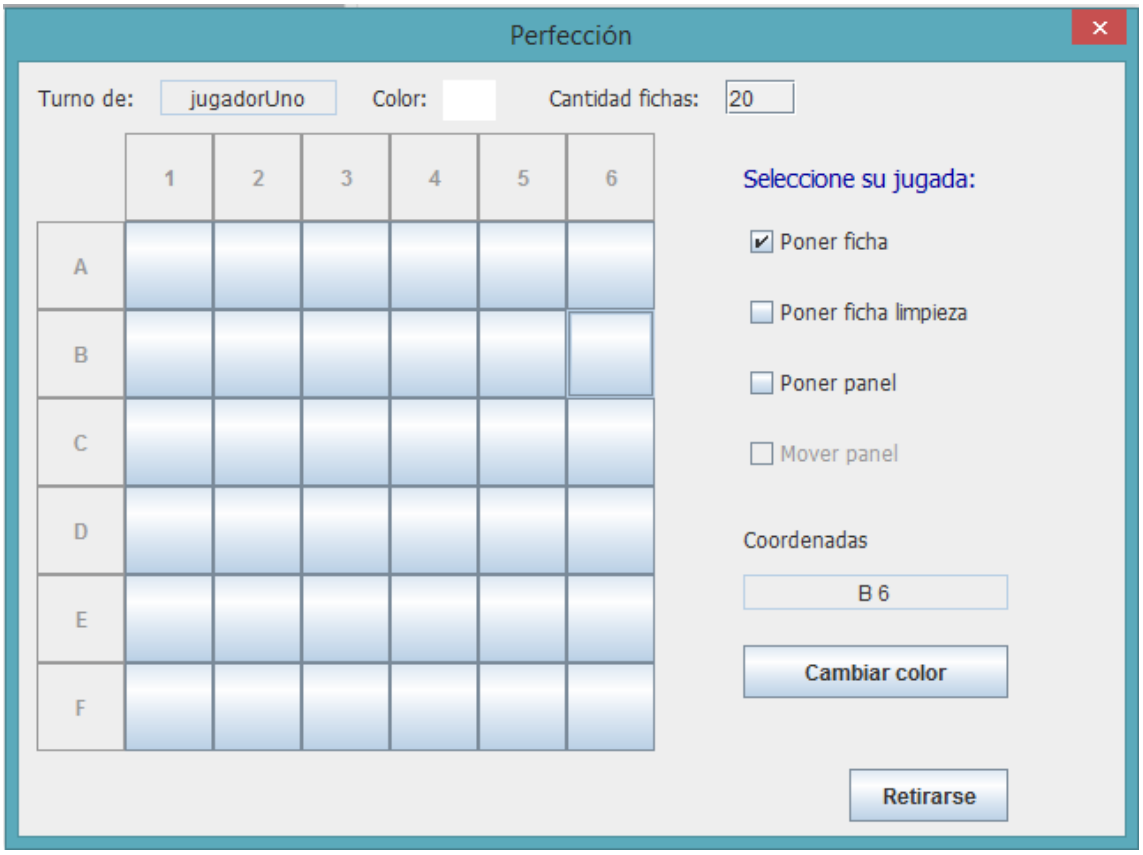
38	Mensaje error:” Ingrese una cantidad de fichas de 1 a 37”.	Mensaje error:” Ingrese una cantidad de fichas de 1 a 37”.	OK
100	Mensaje error:” Ingrese una cantidad de fichas válido”.	Mensaje error:” Ingrese una cantidad de fichas válido”.	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
37	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

Tablero 8x8 y Panel 4x4

Prueba	Resultado esperado	Resultado obtenido	Resumen
0	Mensaje error:” Ingrese una cantidad de fichas de 1 a 40”.	Mensaje error:” Ingrese una cantidad de fichas de 1 a 40”.	OK
41	Mensaje error:” Ingrese una cantidad de fichas de 1 a 40”.	Mensaje error:” Ingrese una cantidad de fichas de 1 a 40”.	OK
100	Mensaje error:” Ingrese una cantidad de fichas válido”.	Mensaje error:” Ingrese una cantidad de fichas válido”.	OK
1	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK
40	Acepta la cantidad de fichas.	Acepta la cantidad de fichas.	OK

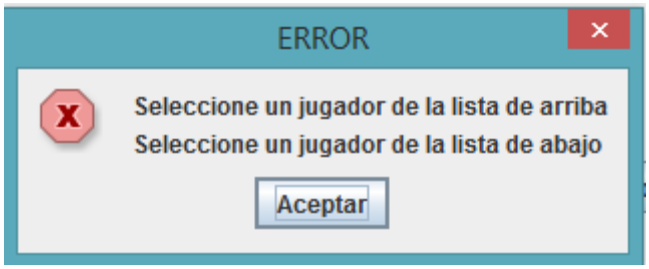
Aceptar:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en Aceptar	Ingrese a la ventana Juego.	Ingrese a la ventana Juego.	OK



Si no se hizo clic en los jugadores:

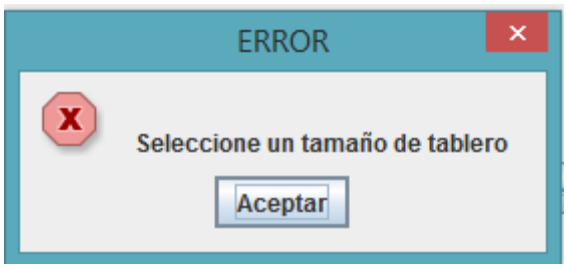
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en Aceptar	Mensaje error: “Seleccione un jugador de la lista de arriba” “Seleccione un jugador de la lista de abajo”.	Mensaje error: “Seleccione un jugador de la lista de arriba” “Seleccione un jugador de la lista de abajo”.	OK



Si no se hizo clic en tablero:

Prueba	Resultado esperado	Resultado obtenido	Resumen

Clic en Aceptar	Mensaje error: “Seleccione un tamaño de tablero”.	Mensaje error: “Seleccione un tamaño de tablero”.	OK

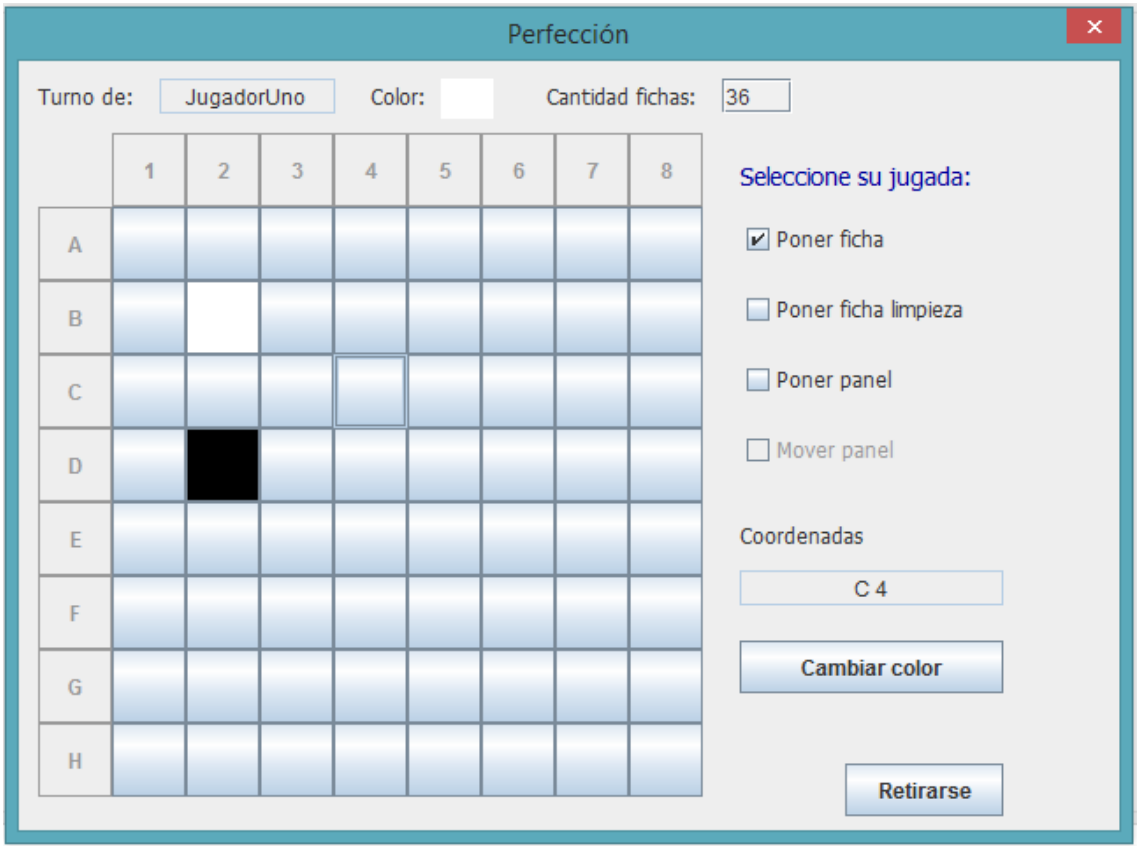


En la ventana del juego:

Las pruebas son con JugadorUno de color blanco, JugadorDos de color negro en tablero de 8x8, panel de 3x3 y 37 fichas para cada jugador.

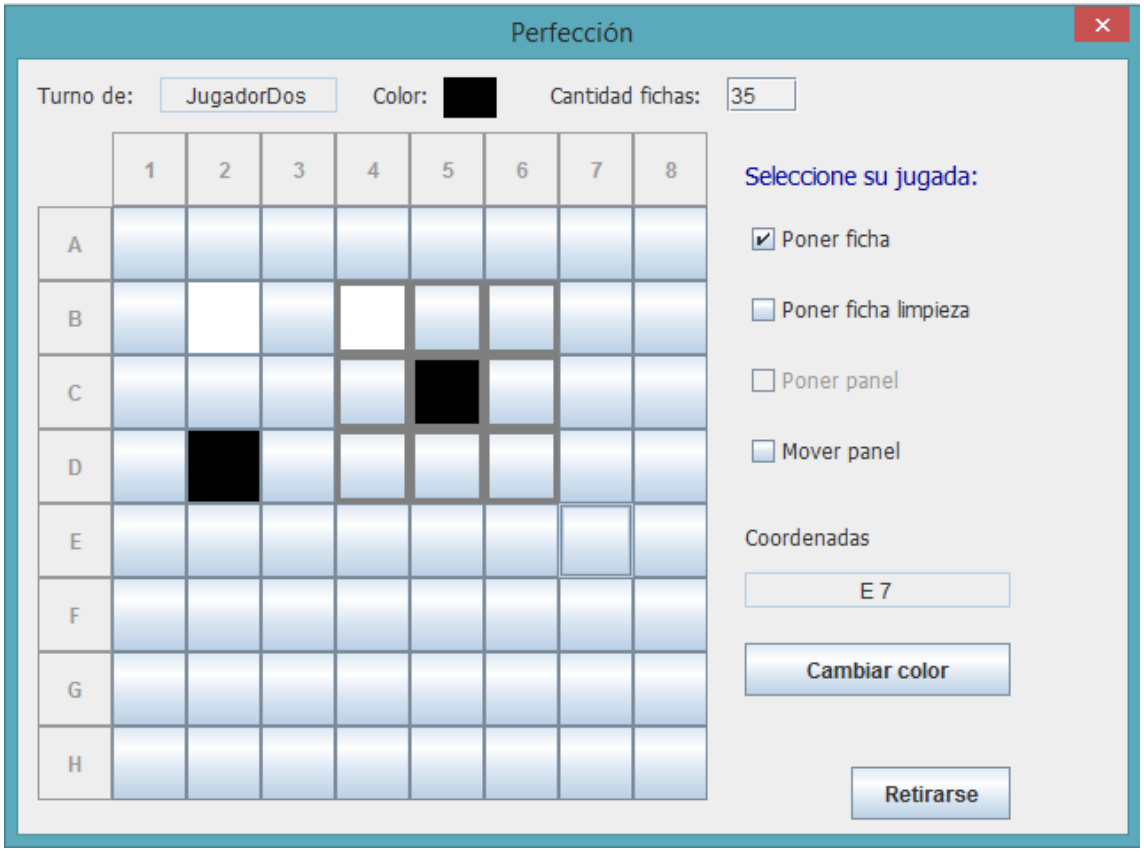
Poner ficha.

Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno JugadorUno color blanco. Clic en B2.	Ingresa una ficha de color blanco en B2.	Ingresa una ficha de color blanco en B2.	OK
Turno JugadorDos color negro. Clic en D2.	Ingresa una ficha de color negro en D2.	Ingresa una ficha de color negro en D2.	OK



Poner ficha con panel ya ingresado en B4.

Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno JugadorUno color blanco. Clic en B4.	Ingresa una ficha de color blanco en el panel posición B4.	Ingresa una ficha de color blanco en el panel posición B4.	OK
Turno JugadorDos color negro. Clic en C5.	Ingresa una ficha de color negro en C5.	Ingresa una ficha de color negro en C5.	OK

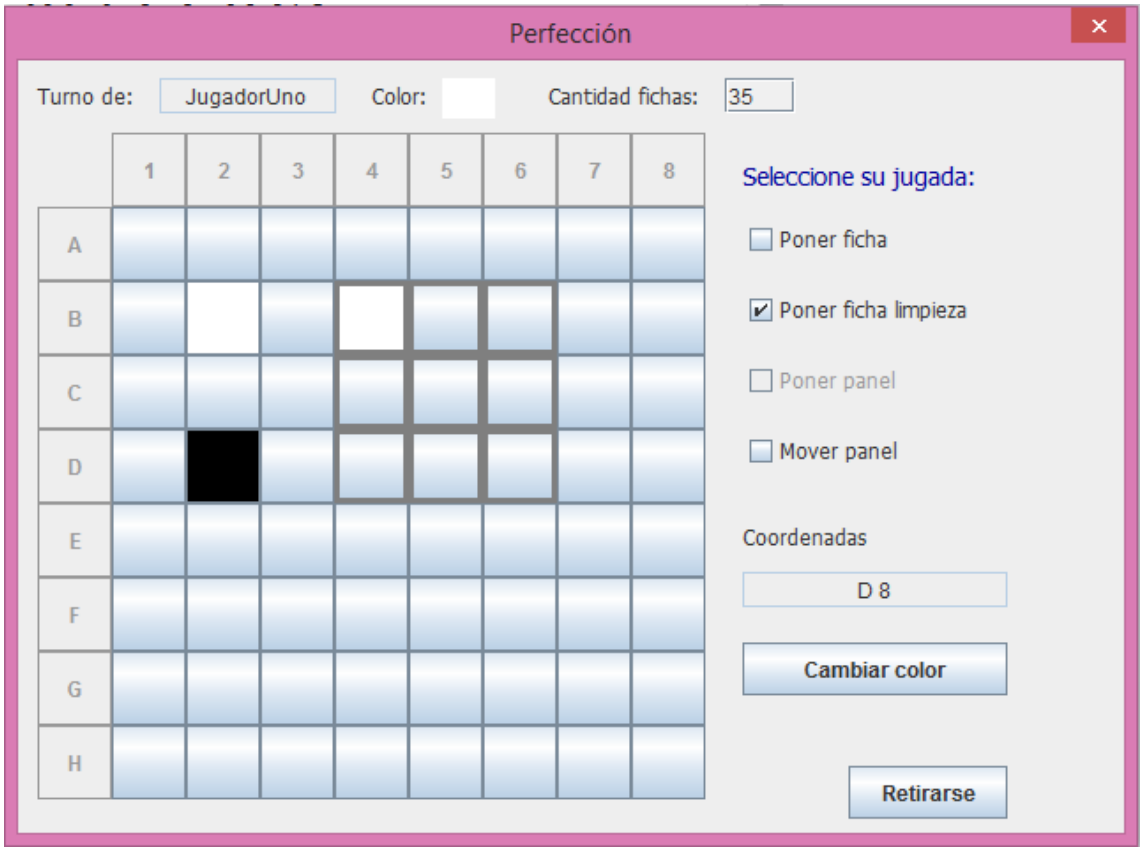


Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno JugadorUno color blanco. Clic en D2.	Mensaje error: “Esa coordenada no es válida”.	Mensaje error: “Esa coordenada no es válida”.	OK

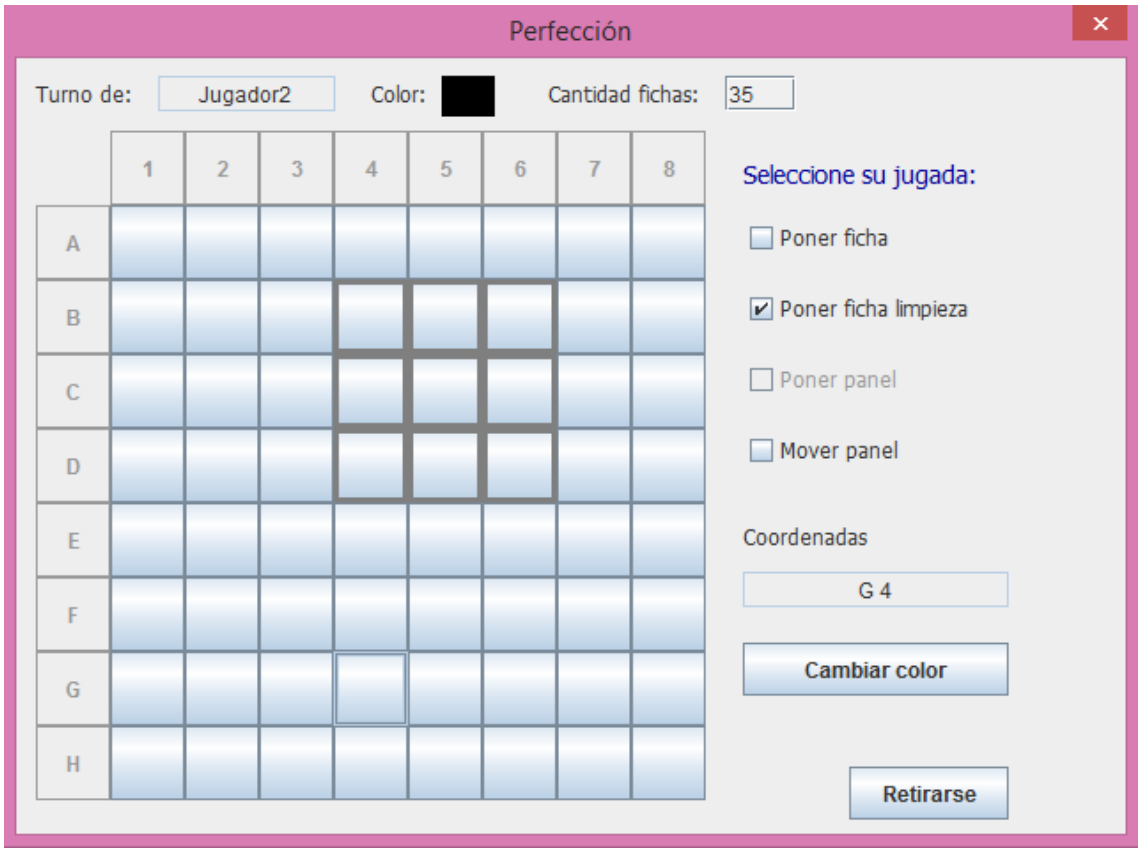
Ficha limpieza:

Prueba	Resultado esperado	Resultado obtenido	Resumen

Turno JugadorDos color negro. Clic en C5.	Borra las fichas que están sobre la fila C columna 5, inclusive la ficha dentro del panel.	Borra las fichas que están sobre la fila C columna 5, inclusive la ficha dentro del panel.	OK



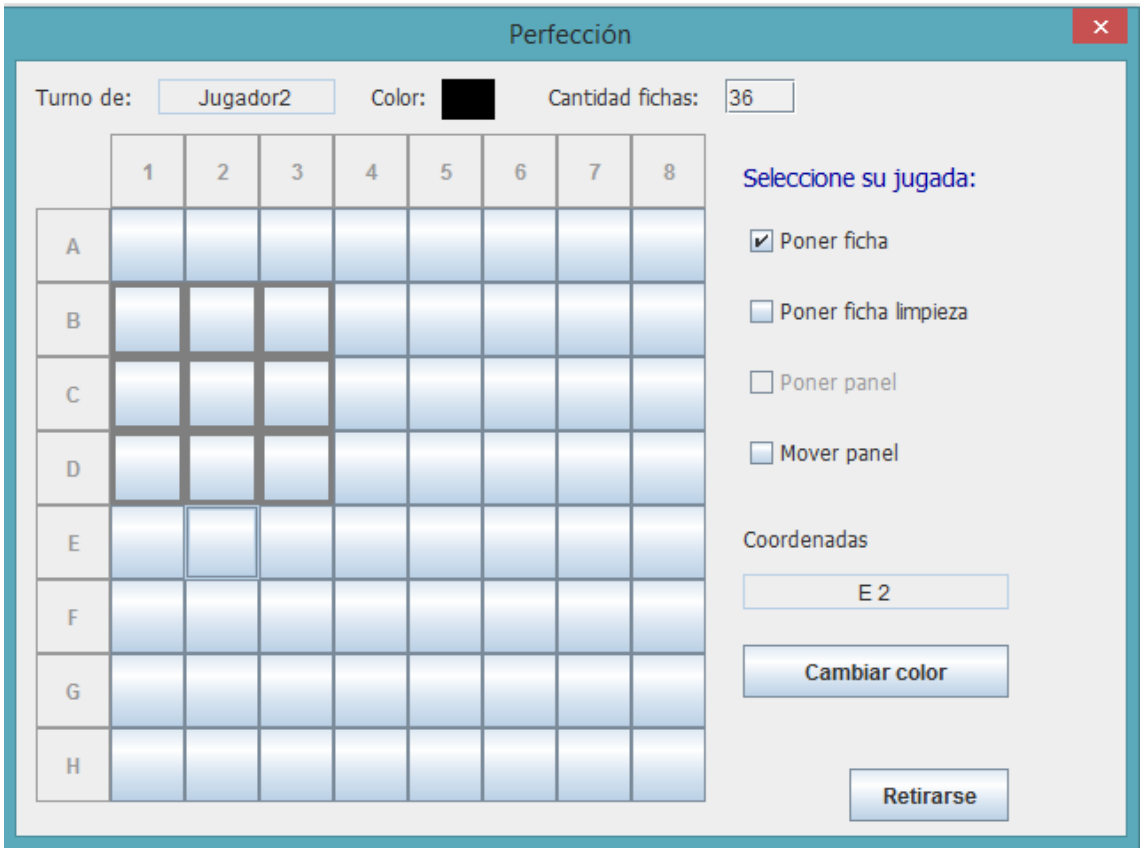
Prueba	Resultado esperado	Resultado obtenido	Resumen
Truno JugadorUno color blanco. Clic en B2.	Borra las fichas que están sobre la fila B columna 2 ,inclusive la ficha del tablero.	Borra las fichas que están sobre la fila B columna 2 ,inclusive la ficha del tablero.	OK



Poner Panel:

Las pruebas son con JugadorUno de color blanco, JugadorDos de color negro en tablero de 8x8, panel de 3x3 y 37 fichas para cada jugador. Ficha blanca ingresada B2 y ficha negra ingresada en D2.

Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno JugadorUno clic en B1.	Ingresa panel en B1 ocultando las fichas.	Ingresa panel en B1 ocultando las fichas.	OK



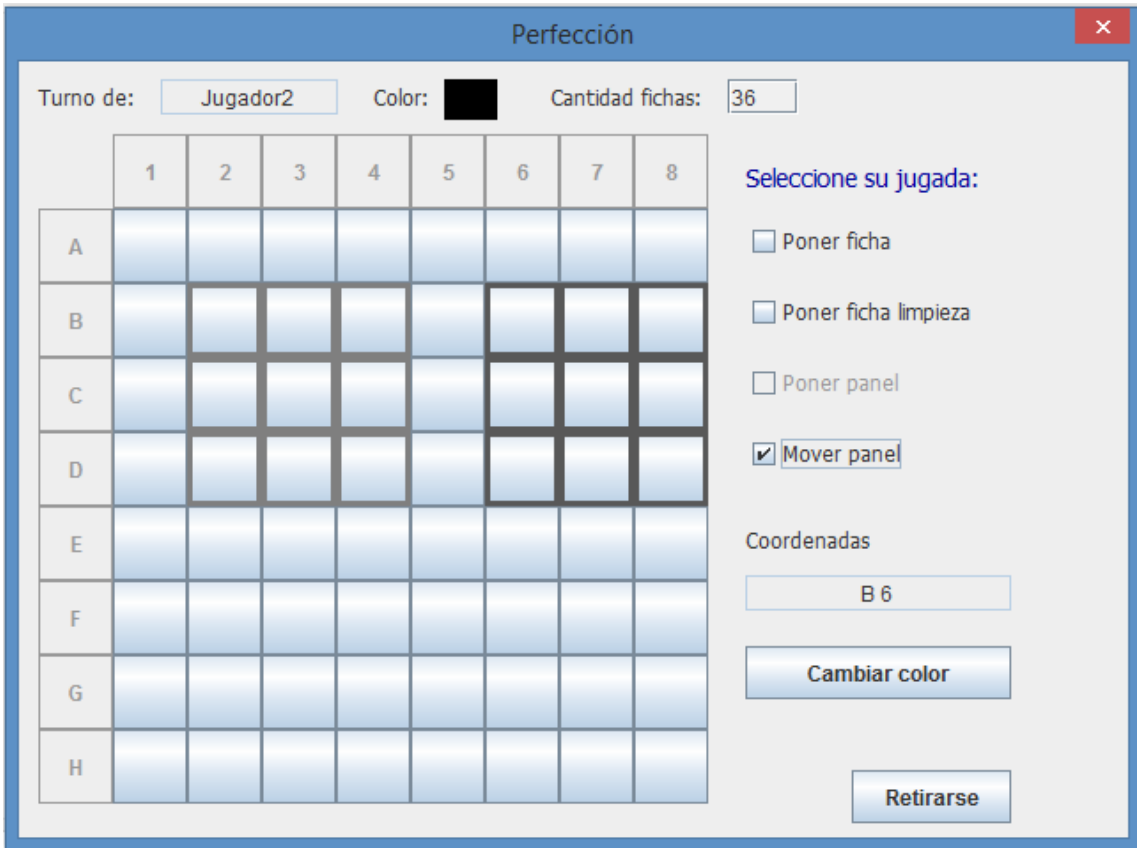
Una vez ingresado el panel se deshabilita la opción poner panel.

Antes de ingresar el panel se dibuja el mismo sobre el tablero indicando las posiciones válidas para colocarlo.

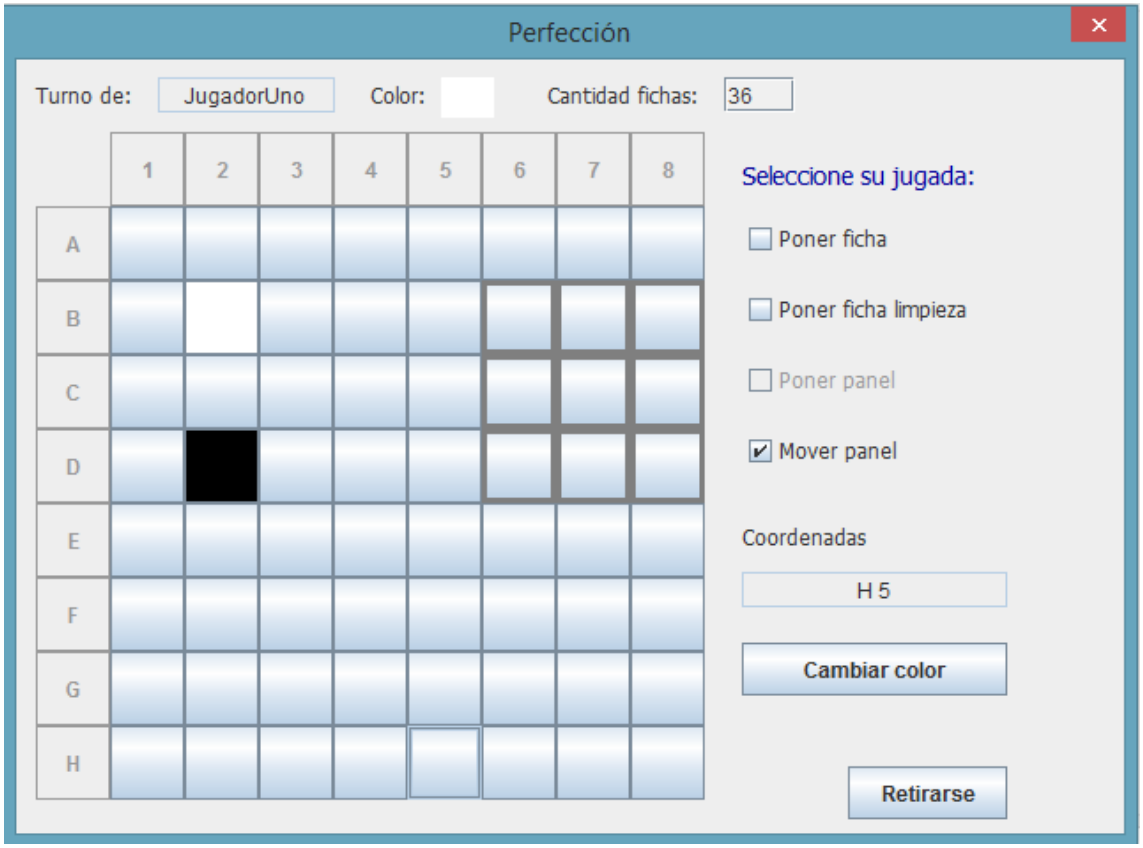
Mover Panel:

Las pruebas son con JugadorUno de color blanco, JugadorDos de color negro en tablero de 8x8, panel de 3x3 y 37 fichas para cada jugador. Ficha blanca ingresada B2, ficha negra ingresada en D2 y panel en B1 ocultando las fichas.

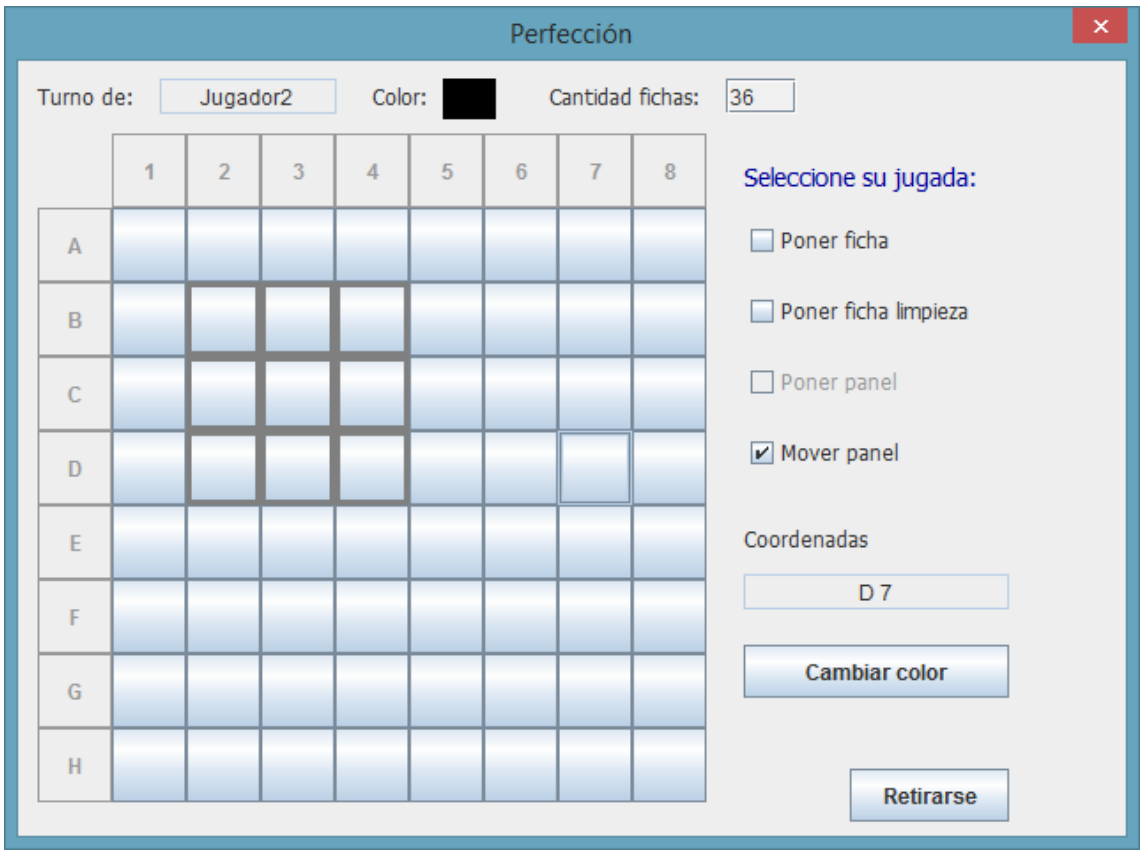
Se muestra sobre el tablero los posibles movimientos del panel, dibujando el panel sobre el tablero como muestra la imágen.


Perfección
Turno de:
Jugador2
Color:
Cantidad fichas:
36
1
2
3
4
5
6
7
8
A
B
C
D
E
F
G
H
Seleccione su jugada:
Poner ficha
Poner ficha limpieza
Poner panel
Mover panel
Coordenadas
B 6
Cambiar color
Retirarse

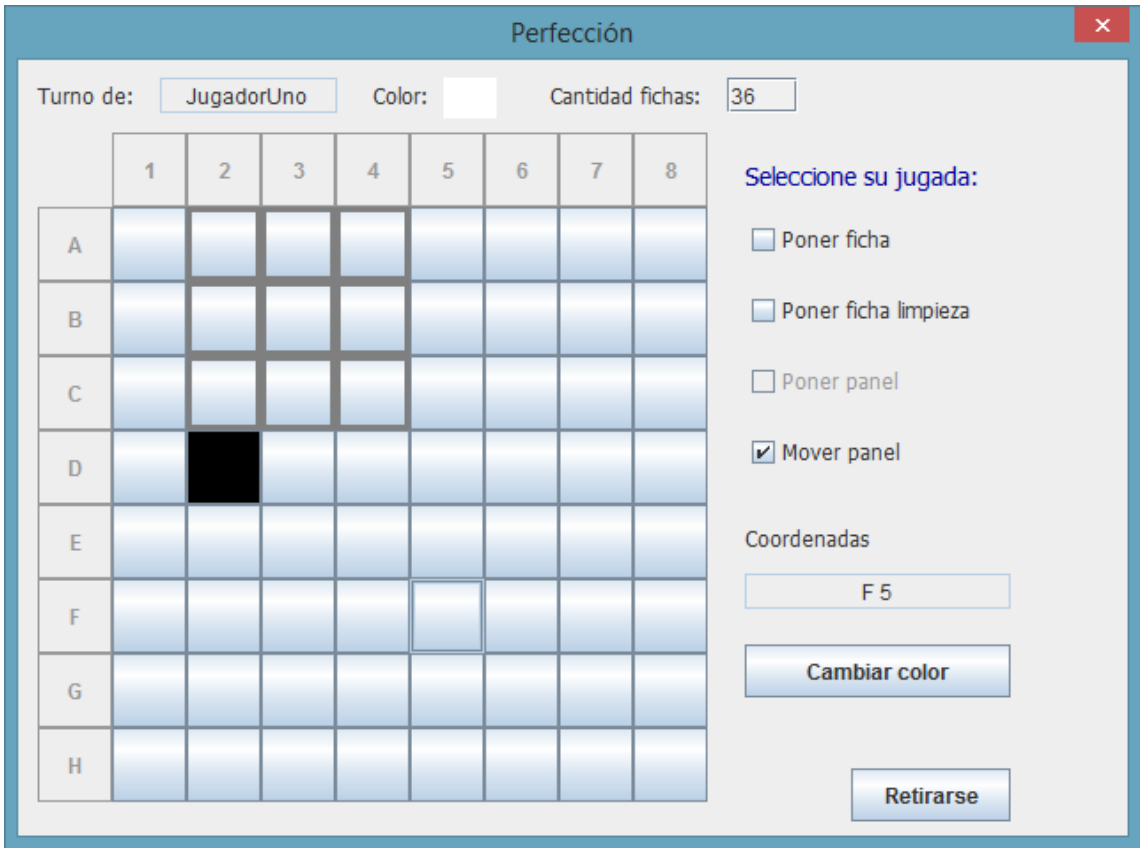
Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno jugadorDos clic en B6.	Mueve el panel hacia la derecha.	Mueve el panel hacia la derecha.	OK



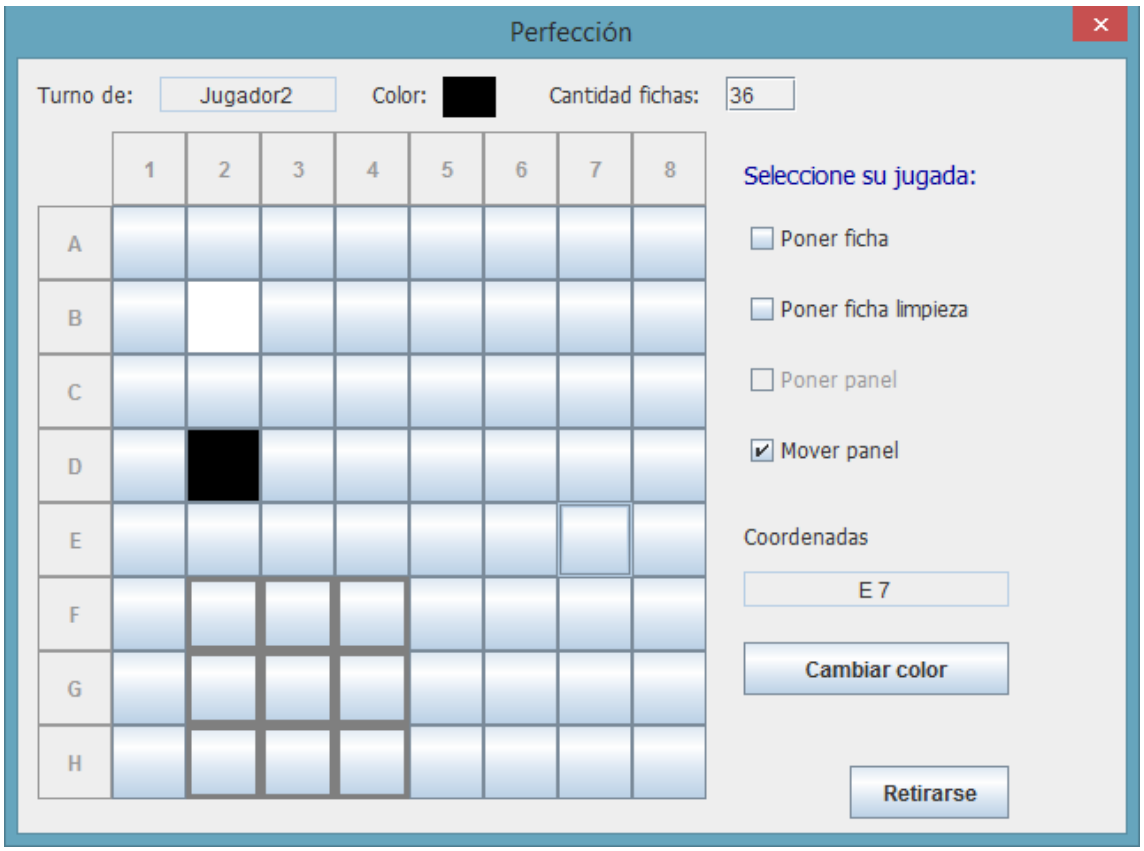
Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno jugadorUno clic en B2.	Mueve el panel hacia la izquierda.	Mueve el panel hacia la izquierda.	OK



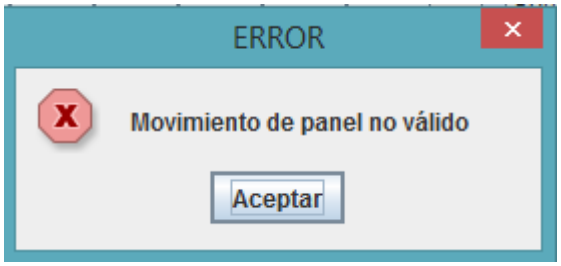
Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno jugadorDos clic en A2.	Mueve el panel hacia arriba.	Mueve el panel hacia arriba.	OK



Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno jugadorUno clic en F2.	Mueve el panel hacia la abajo.	Mueve el panel hacia la abajo.	OK

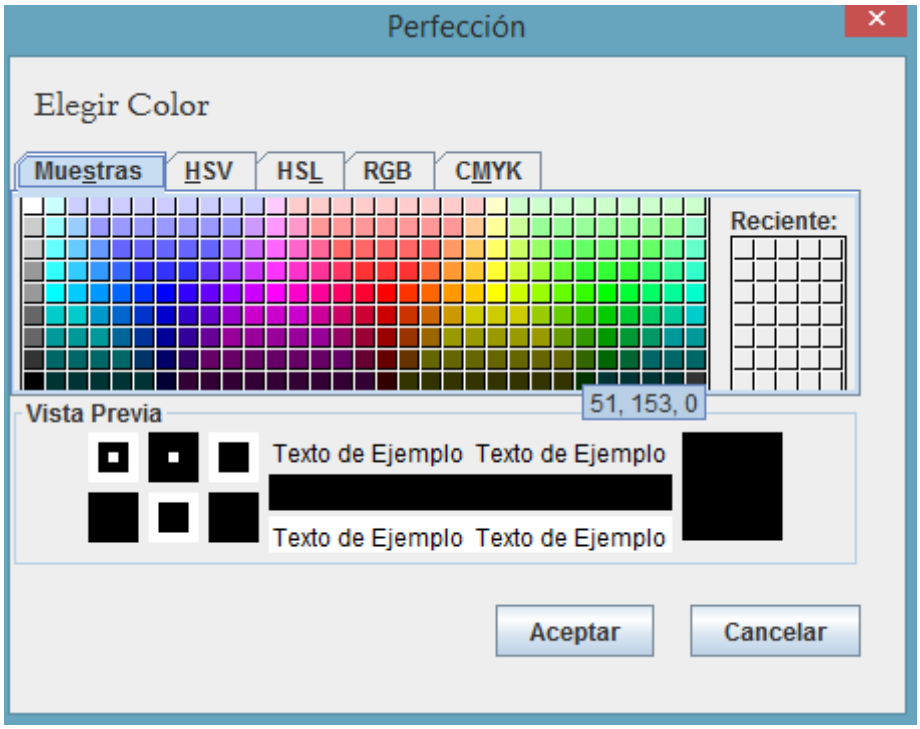


Prueba	Resultado esperado	Resultado obtenido	Resumen
Turno jugadorDos clic en D5.	Mensaje error: ”Movimiento de panel no válido”.	Mensaje error: ”Movimiento de panel no válido”.	OK



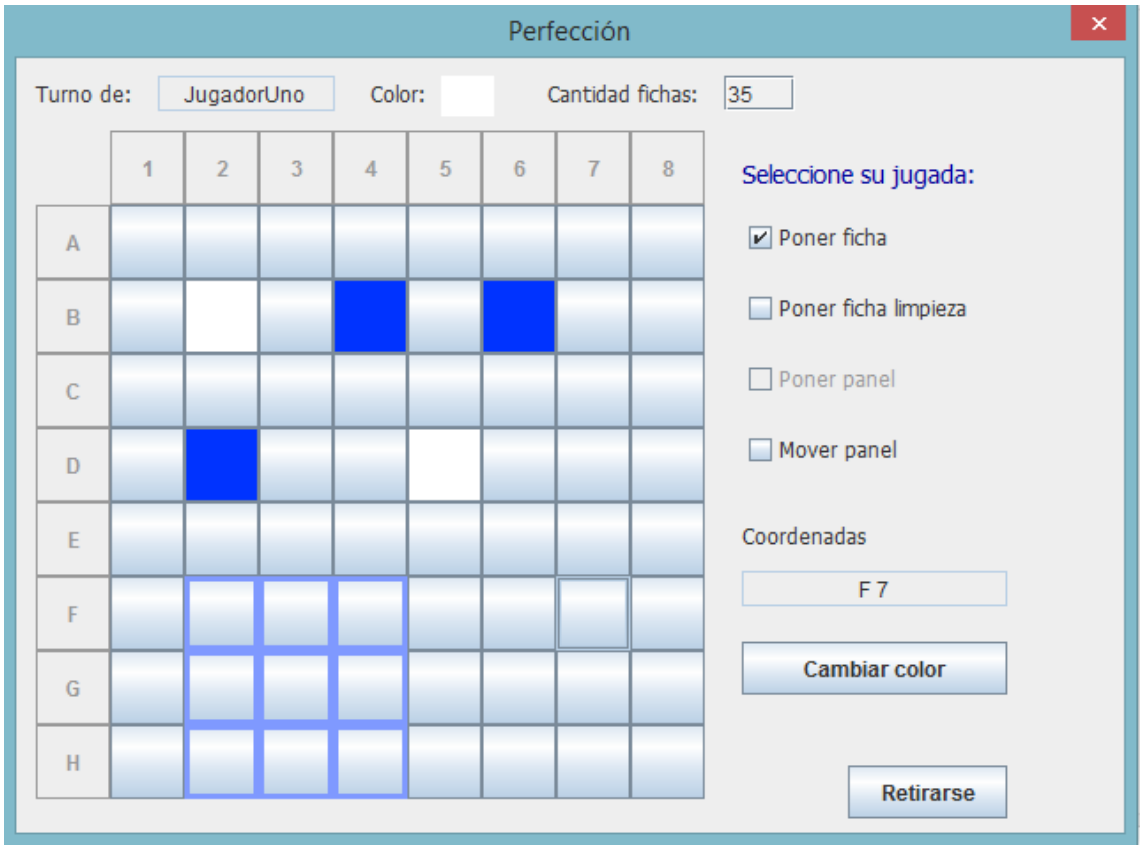
Cambiar Color:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón cambiar color.	Ingresa a la ventana color mostrando la paleta de colores.	Ingresa a la ventana color mostrando la paleta de colores.	OK



Jugador 2:

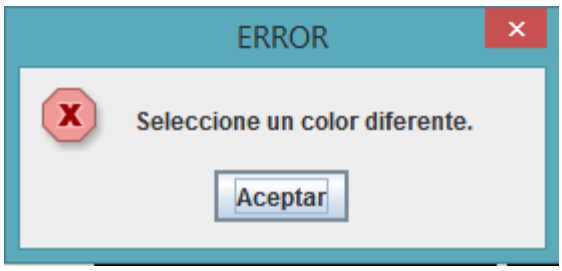
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en color azul luego clic en aceptar.	El jugador 2 tiene un color de la paleta de colores. Las fichas del jugador dos cambian a color azul.	El jugador 2 tiene un color de la paleta de colores. Las fichas del jugador dos cambian a color azul.	OK
Clic en Cancelar sin elegir un color.	Vuelve al color anterior y retorna a la partida.	Vuelve al color anterior y retorna a la partida.	OK
Clic en Aceptar sin elegir un color.	Mensaje error: “Seleccione un color diferente”.	Mensaje error: “Seleccione un color diferente”.	OK

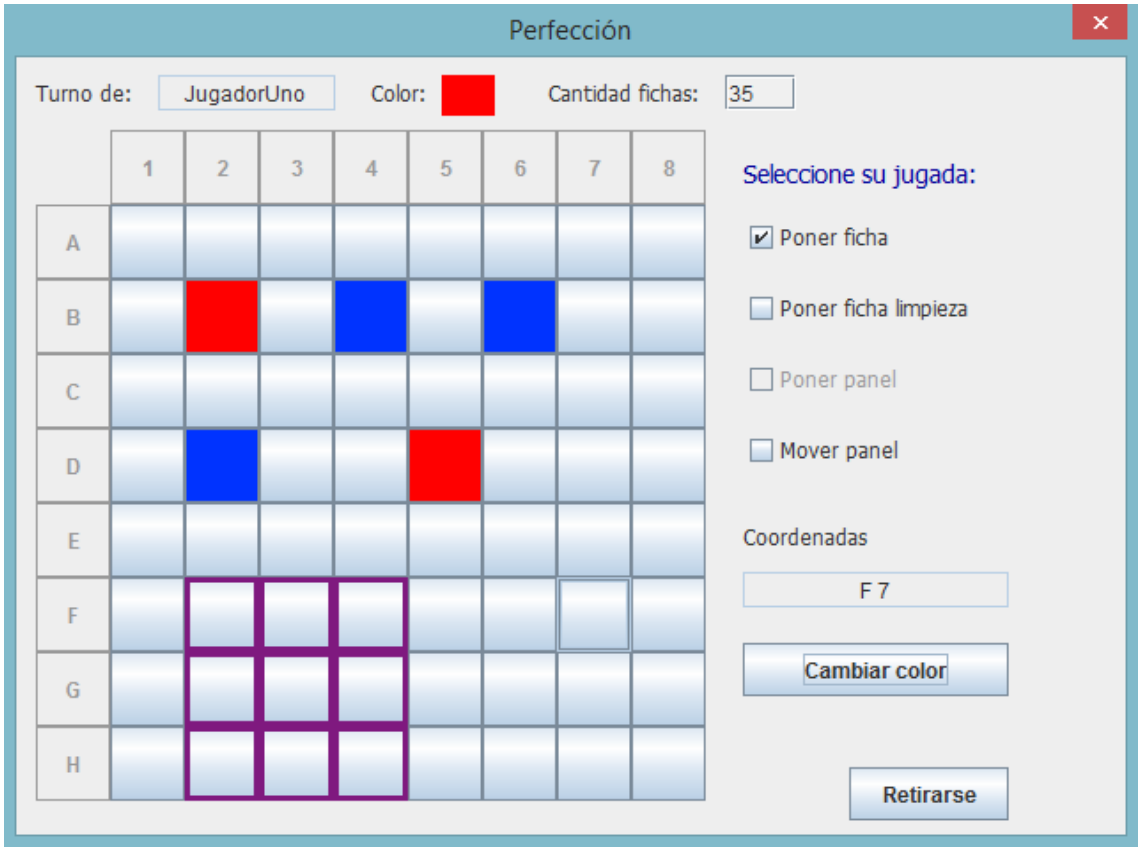


Jugador 1:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en color rojo luego clic en aceptar.	El jugador 1 tiene un color de la paleta de colores. Las fichas del jugador uno cambian a color rojo.	El jugador 1 tiene un color de la paleta de colores. Las fichas del jugador uno cambian a color rojo.	OK
Clic en Cancelar sin elegir un color.	Vuelve al color anterior y retorna a la partida.	Vuelve al color anterior y retorna a la partida.	OK

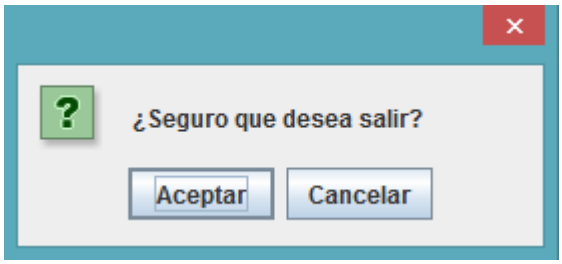
Clic en Aceptar sin elegir un color.	Mensaje error: “Seleccione un color diferente”.	Mensaje error: “Seleccione un color diferente”.	OK





Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en la paleta de colores eligiendo el mismo color.	Mensaje error: “Ambos jugadores no pueden tener el mismo color”.	Mensaje error: “Ambos jugadores no pueden tener el mismo color”.	OK

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en el botón cruz.	Mensaje: “¿Seguro que desea salir?”.	Mensaje: “¿Seguro que desea salir?”.	OK
Clic en el botón cancelar.	Vuelve a la ventana juego.	Vuelve a la ventana juego.	OK
Clic en el botón aceptar.	Vuelve a la ventana menú principal y muestra un mensaje, si la partida termino en empate o quien fue el jugador ganador indicando el alias.	Vuelve a la ventana menú principal y muestra un mensaje, si la partida termino en empate o quien fue el jugador ganador indicando el alias.	OK



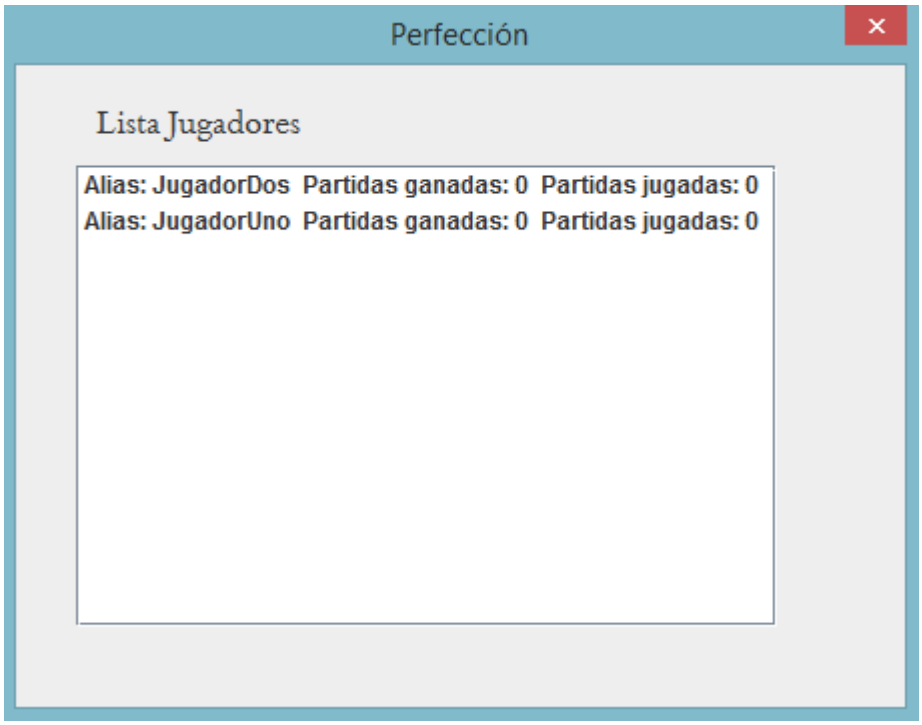
Retirarse:

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón retirarse.	Vuelve a la ventana	Vuelve a la ventana	OK

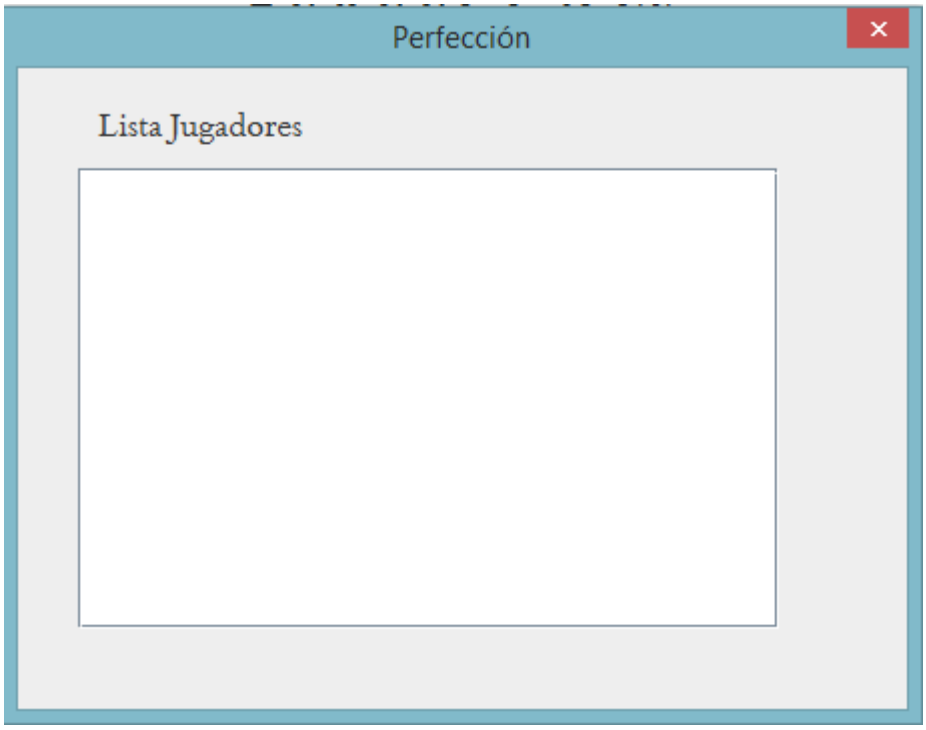
	menú principal y muestra un mensaje, si la partida termino en empate o quien fue el jugador ganador indicando el alias.	menú principal y muestra un mensaje, si la partida termino en empate o quien fue el jugador ganador indicando el alias.	

Consulta Jugadores

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón Consulta Jugadores.	Muestra la lista ordenada alfabéticamente por alias mostrando la cantidad de partidas ganadas y jugadas.	Muestra la lista ordenada alfabéticamente por alias mostrando la cantidad de partidas ganadas y jugadas.	OK



Si no hay jugadores registrados muestra la lista vacía.



Cargar Jugadores

Para esta prueba se crea un archivo llamado

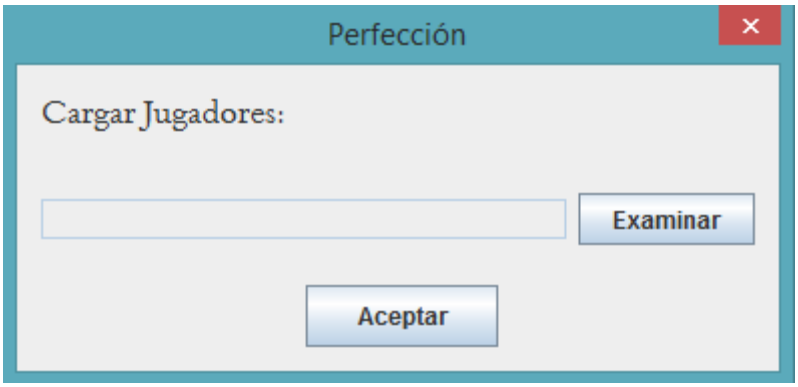
archivo.txt el cual contiene los siguientes datos:

Ignacio#23#Nacho

Lucía#15#Lu

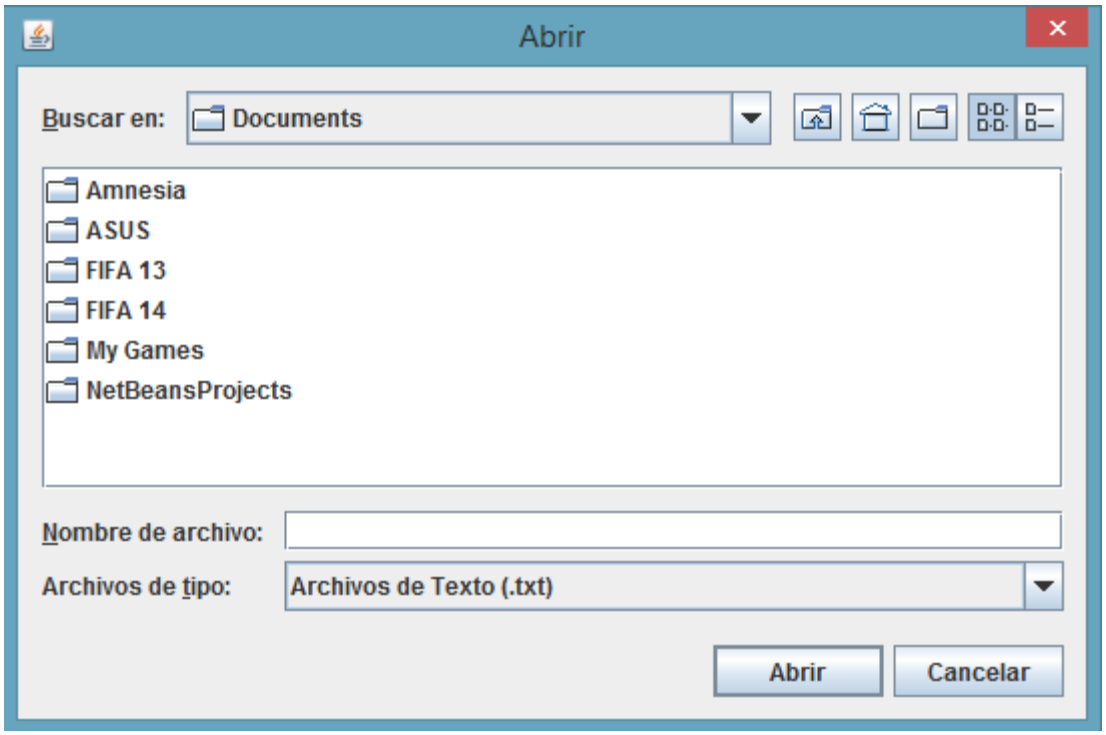
Mathias#40#Mathi

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón Cargar Jugadores.	Ingresa a la ventana cargar jugadores.	Ingresa a la ventana cargar jugadores.	OK

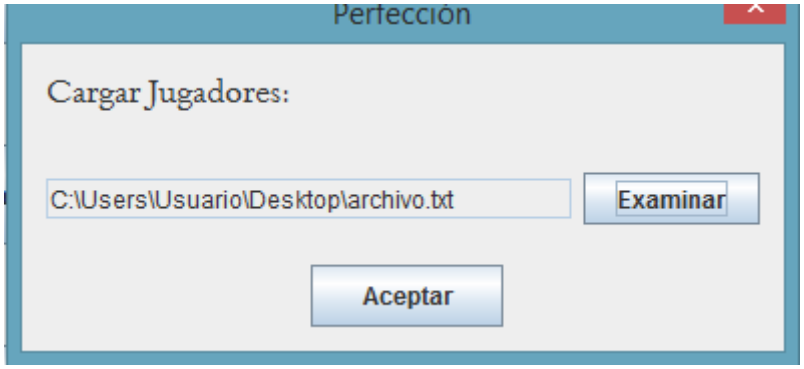


Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón aceptar.	Mensaje error: “Seleccione un archivo a cargar”.	Mensaje error: “Seleccione un archivo a cargar”.	OK

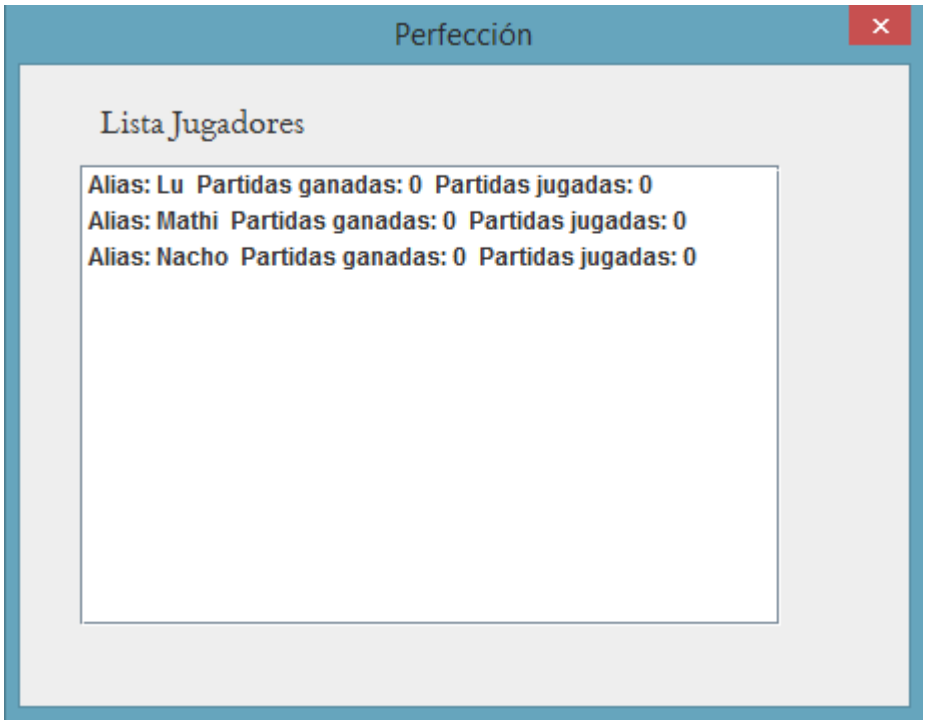
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón examinar.	Ingresa a la ventana para seleccionar la ruta del archivo.	Ingresa a la ventana para seleccionar la ruta del archivo.	OK



Selecciono el archivo.txt



Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en aceptar.	Carga el archivo correctamente.	Carga el archivo correctamente.	OK



Se crea un archivo llamado

archivo.txt el cual está vacío:

Selecciono el archivo.txt

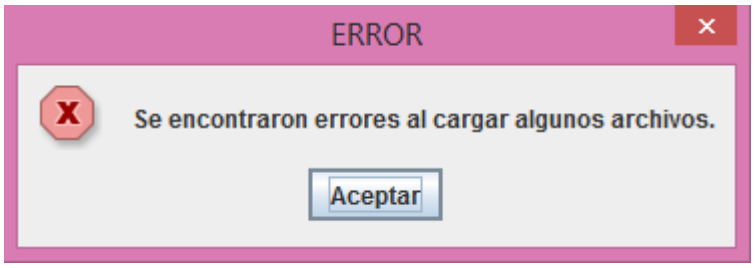
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en aceptar.	Ingresa al menú principal sin cargarme jugadores.	Ingresa al menú principal sin cargarme jugadores.	OK

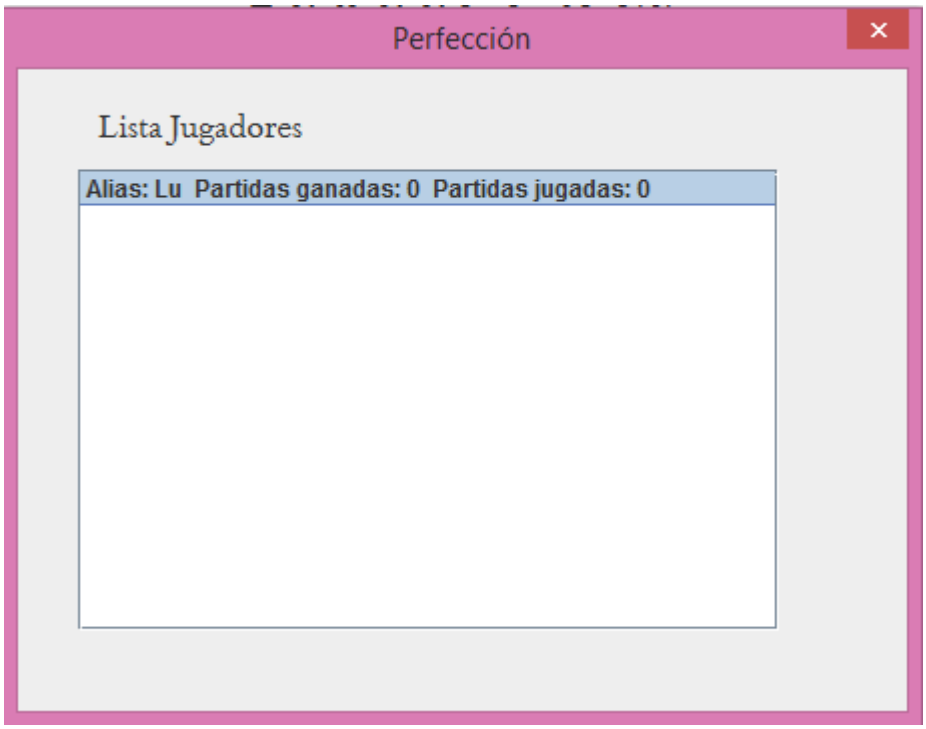
Se crea un archivo llamado
archivo.txt el cual contiene los siguientes datos:

Ignacio#23#Nacho#56
Lucía#15#Lu
Mathias#40#Mathi#Esteban
Carlos#343

Selecciono el archivo.txt

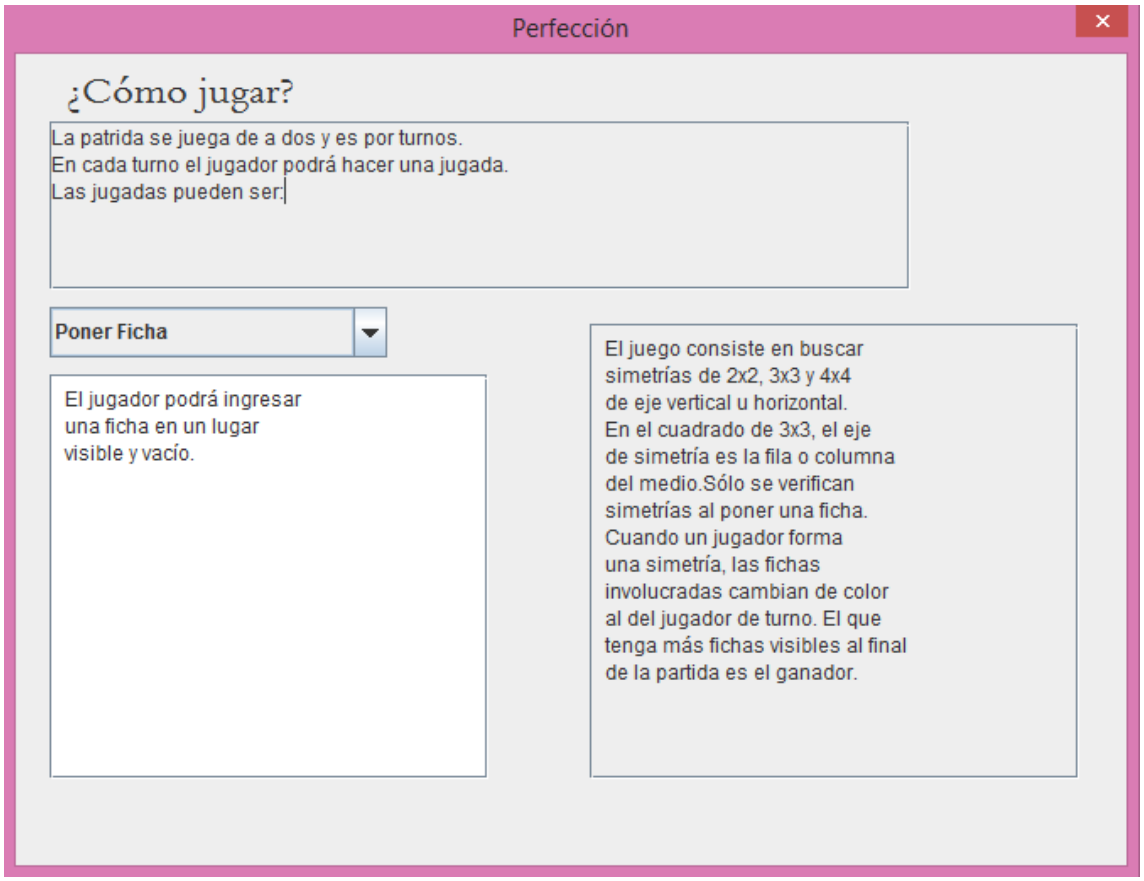
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en aceptar	Mensaje error: “Se encontraron errores al cargar algunos archivos”. Carga solo los jugadores que siguen el siguiente formato: nombre#alias#edad	Mensaje error: “Se encontraron errores al cargar algunos archivos”. Carga solo los jugadores que siguen el siguiente formato: nombre#alias#edad	OK



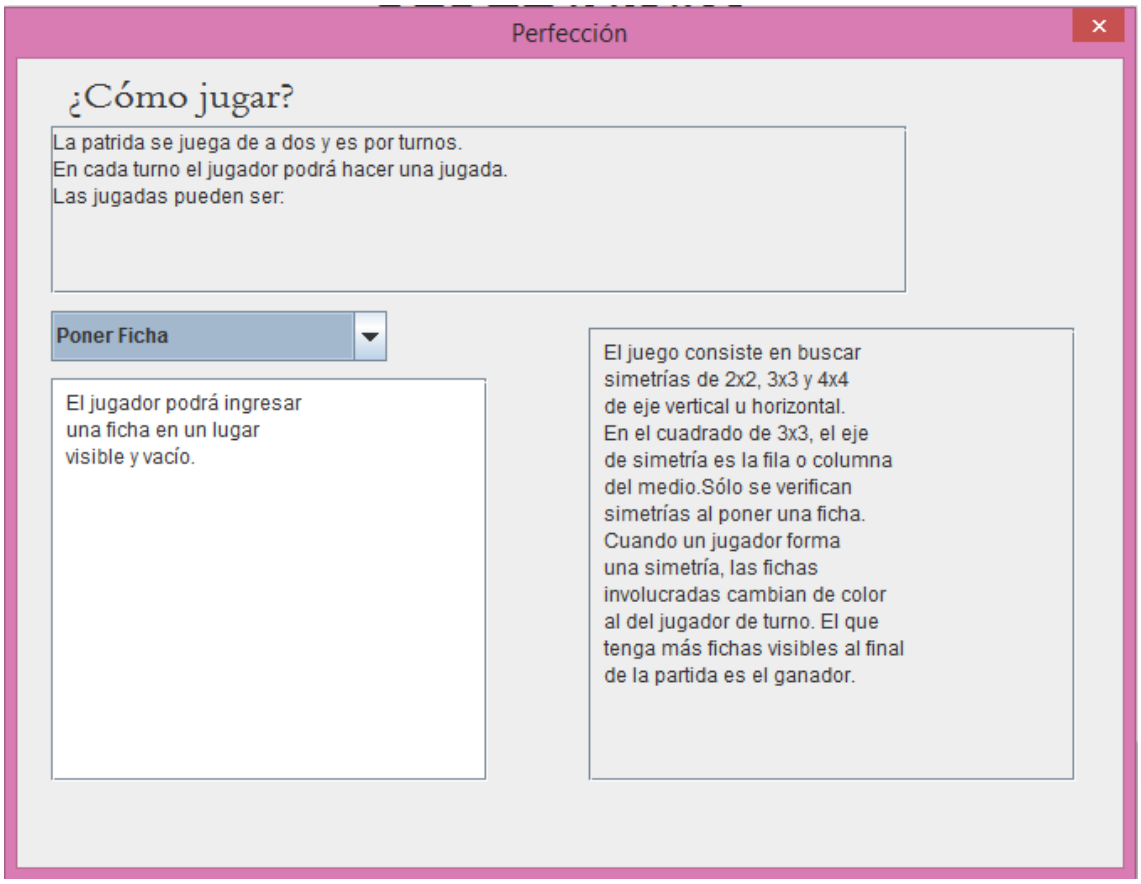


¿Cómo jugar?

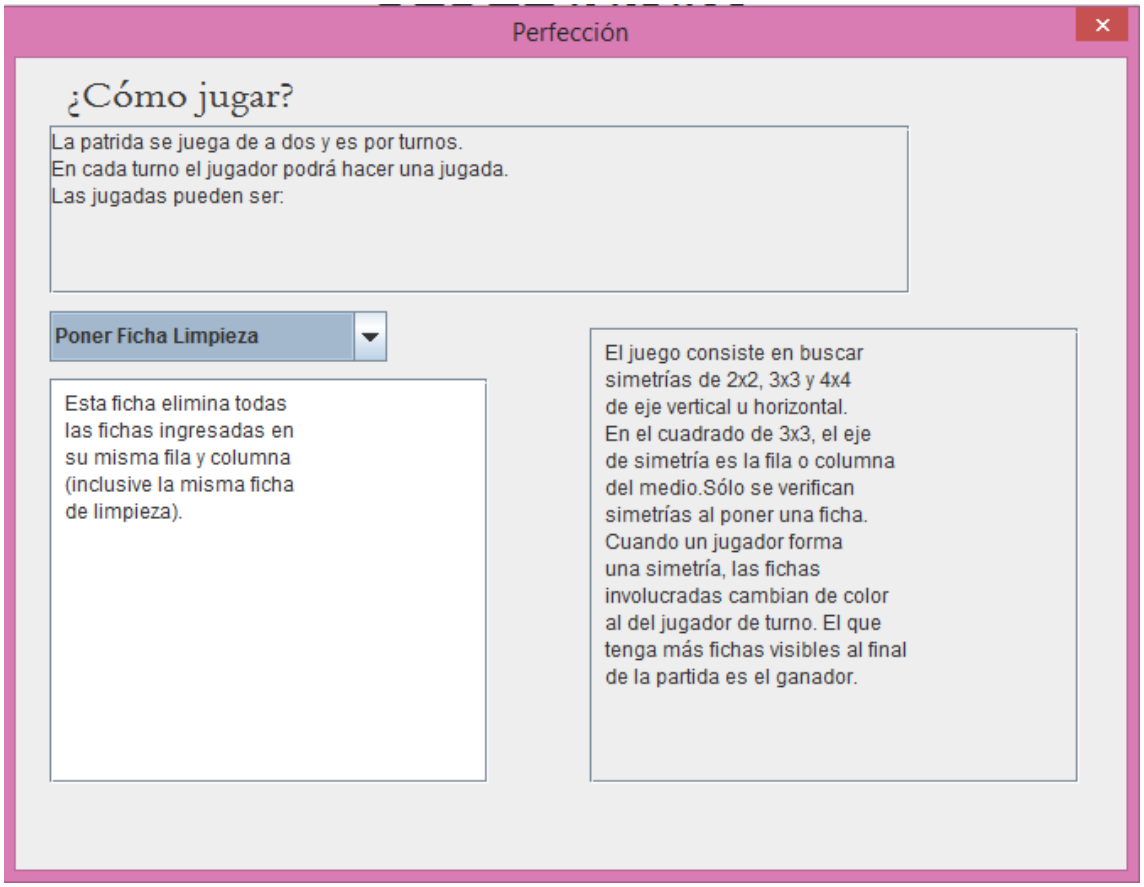
Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en el botón ¿Cómo Jugar?	Ingresa a la ventana Cómo Jugar.	Ingresa a la ventana como jugar.	OK



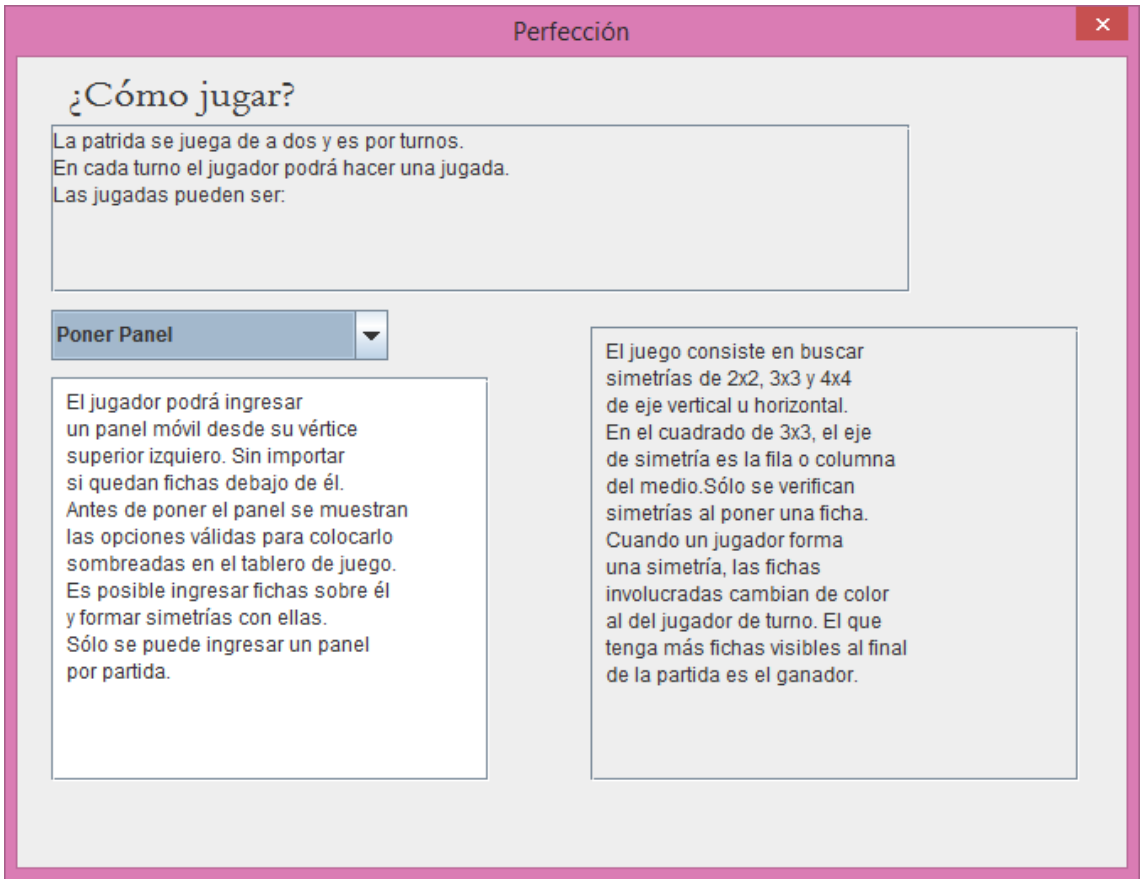
Prueba	Resultado esperado	Resultado obtenido	Resumen
Selecciona “Poner Ficha”.	Actualiza un área de texto mostrando las instrucciones.	Actualiza un área de texto mostrando las instrucciones.	OK



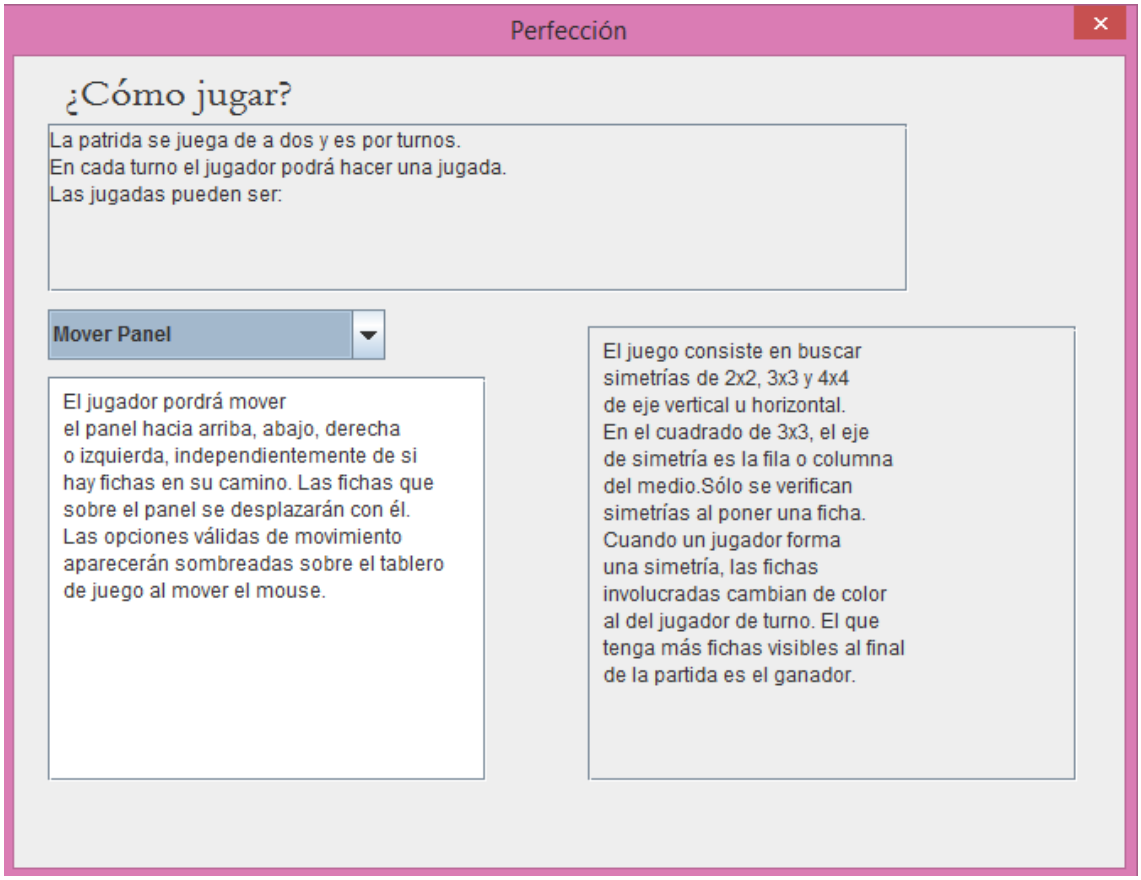
Prueba	Resultado esperado	Resultado obtenido	Resumen
Selecciona “Poner Ficha Limpieza”.	Actualiza un área de texto mostrando las instrucciones.	Actualiza un área de texto mostrando las instrucciones.	OK



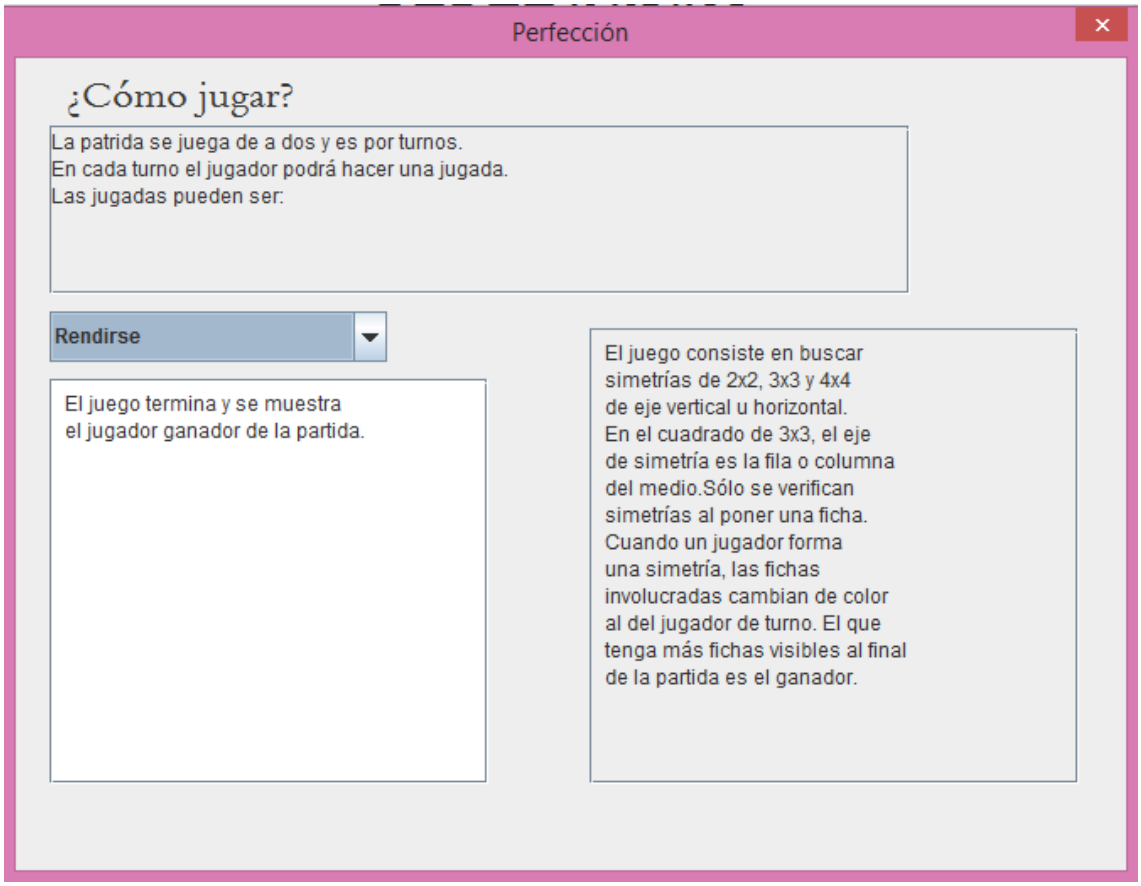
Prueba	Resultado esperado	Resultado obtenido	Resumen
Selecciona “Poner Panel”.	Actualiza un área de texto mostrando las instrucciones.	Actualiza un área de texto mostrando las instrucciones.	OK



Prueba	Resultado esperado	Resultado obtenido	Resumen
Selecciona “Mover Panel”.	Actualiza un área de texto mostrando las instrucciones.	Actualiza un área de texto mostrando las instrucciones.	OK



Prueba	Resultado esperado	Resultado obtenido	Resumen
Selecciona “Rendirse”.	Actualiza un área de texto mostrando las instrucciones.	Actualiza un área de texto mostrando las instrucciones.	OK



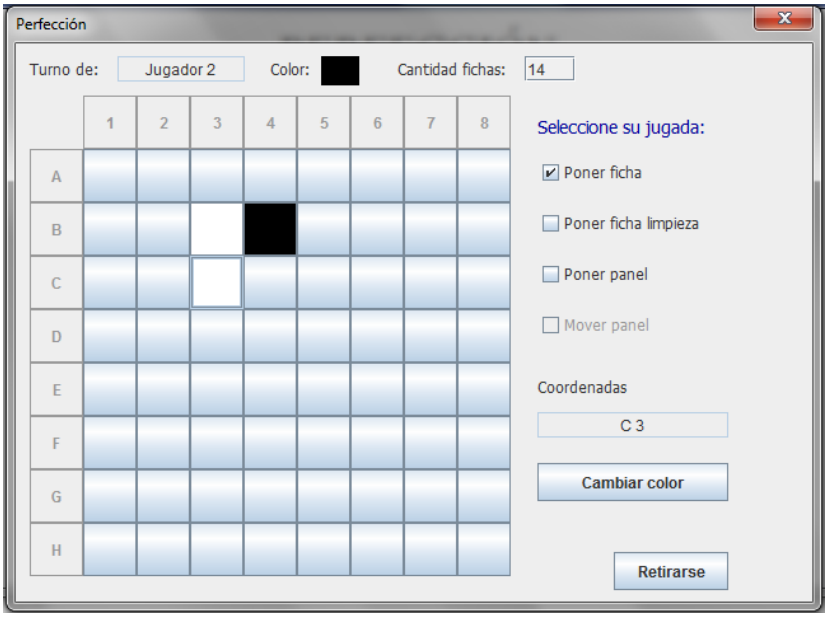
Salir

Prueba	Resultado esperado	Resultado obtenido	Resumen
Clic en botón salir.	Cierra la ventana menú principal. Y se cierra el programa.	Cierra la ventana menú principal. Y se cierra el programa.	OK

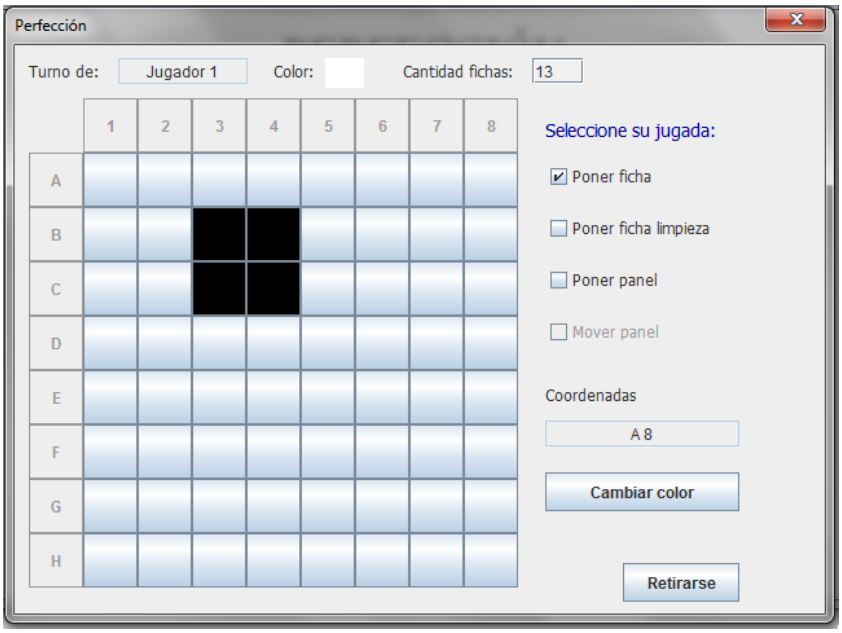
Pruebas de simetrías

Consideremos los siguientes casos de prueba de simetrías para el tablero de 8x8 en el cual el panel no ha sido colocado.

Datos ingresados: Jugador 2 en la posición C4

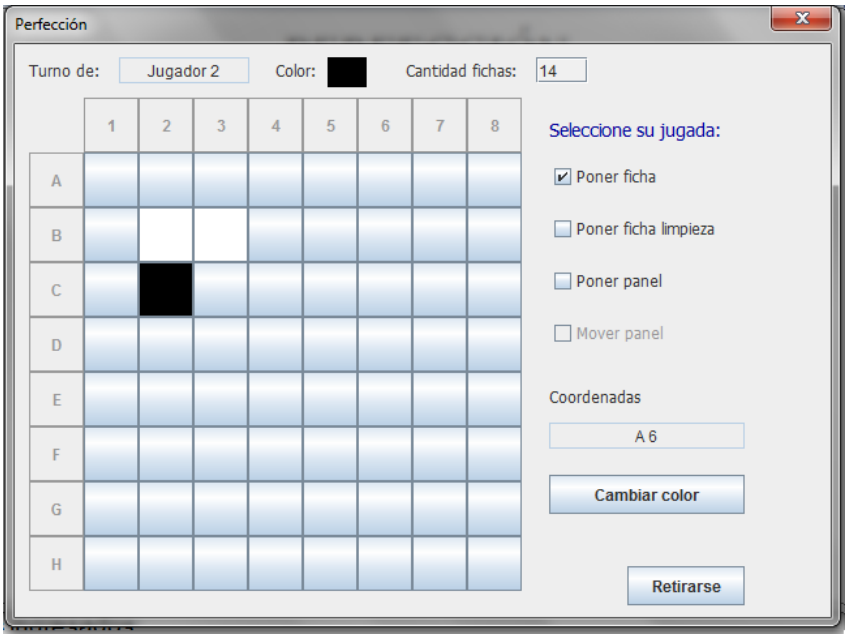


Resultado esperado:

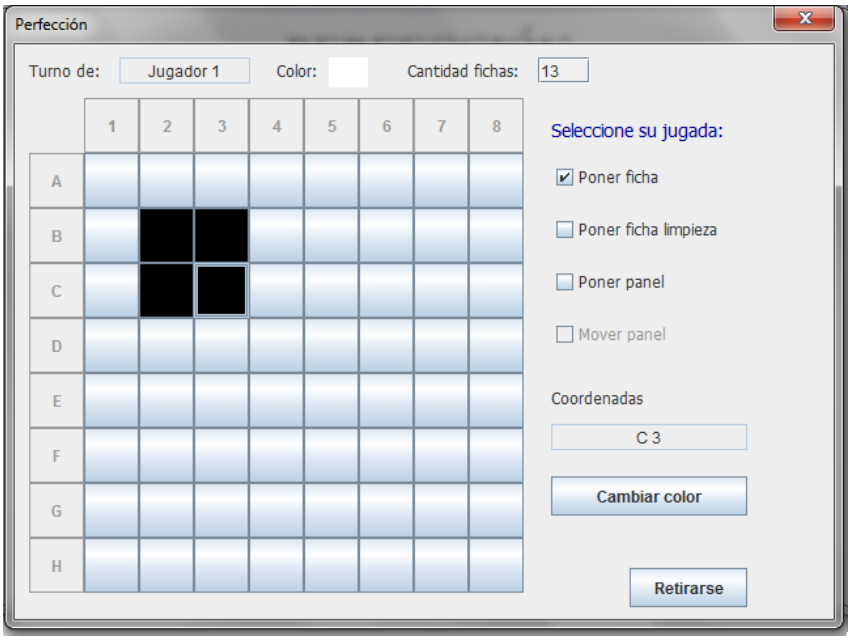


Resumen: Ok

Datos ingresados: Jugador 2 en la posición C3

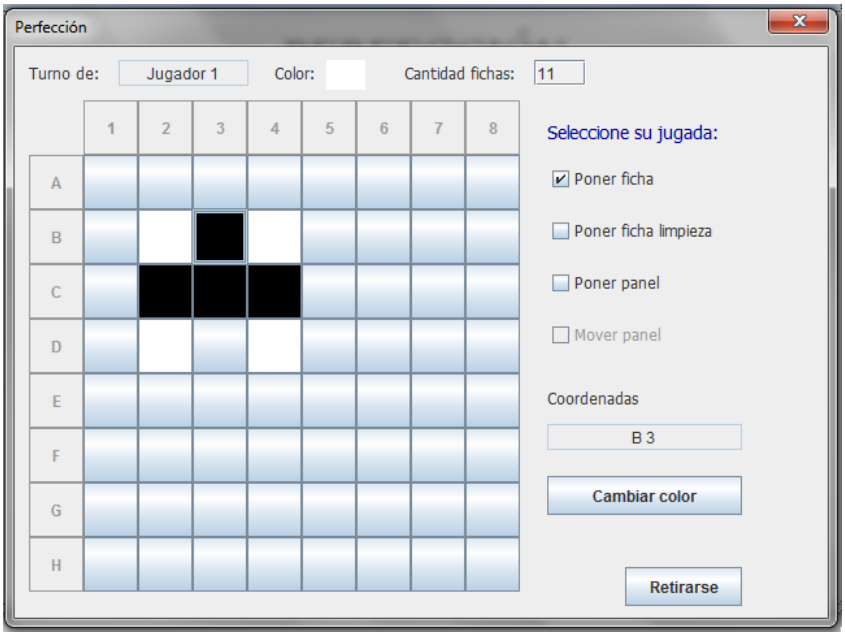


Resultado esperado:

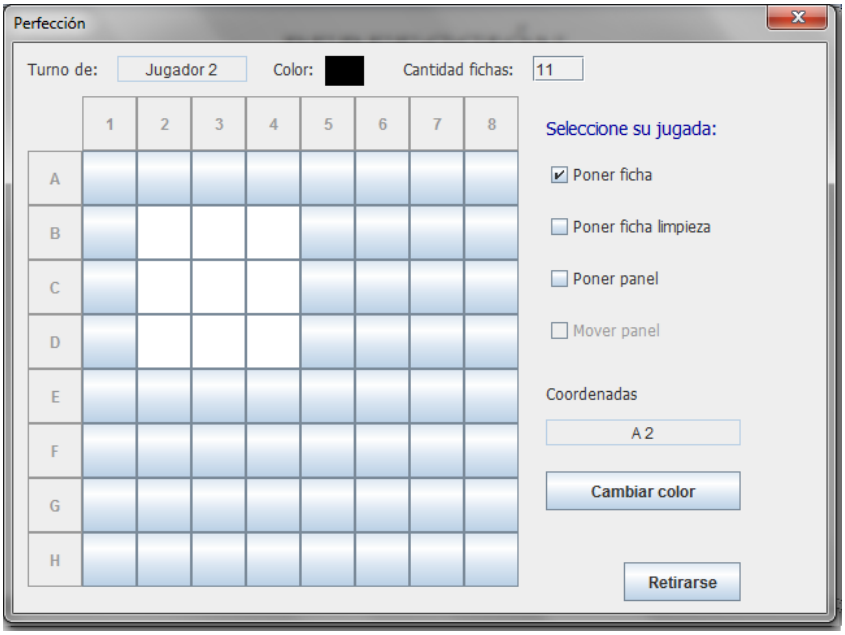


Resumen: Ok

Datos ingresados: Jugador 1 en la posición D3

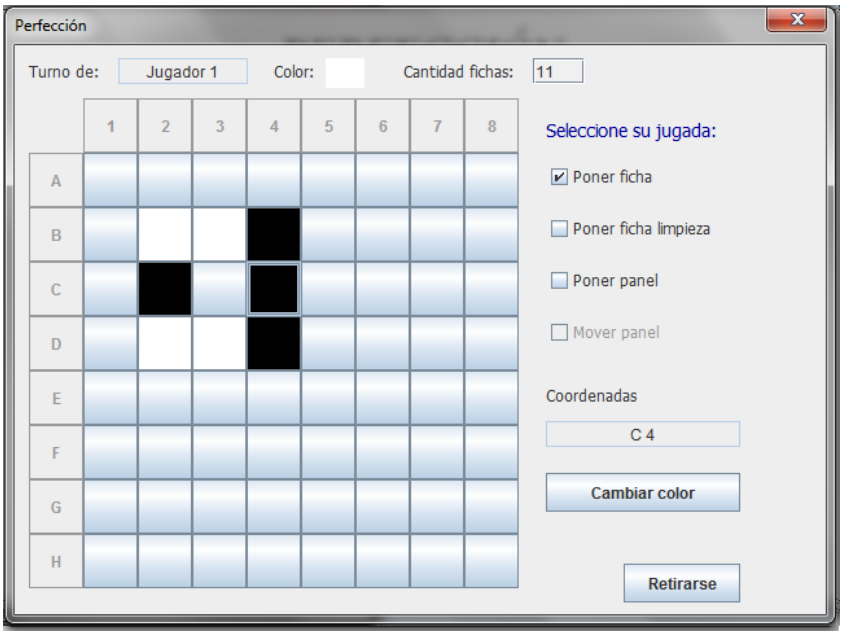


Resultado esperado:

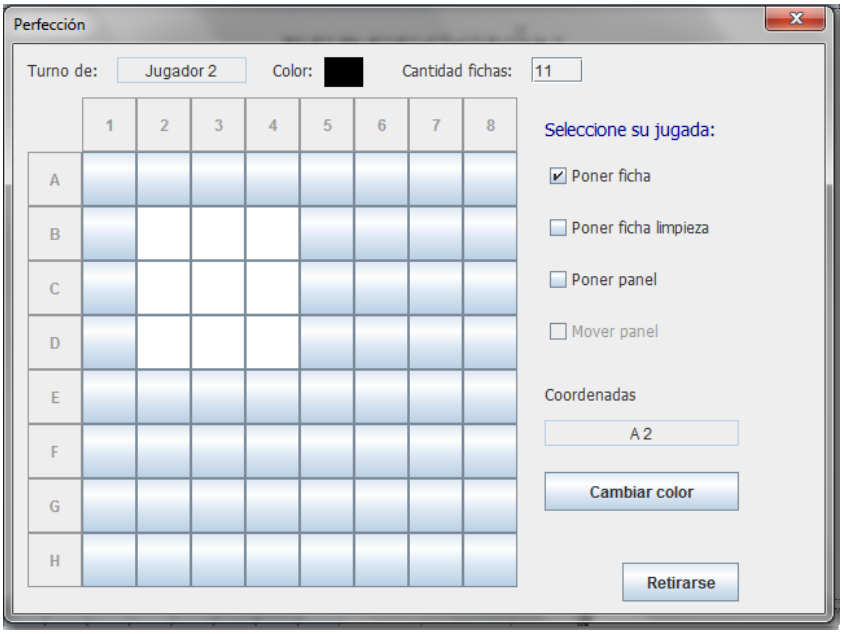


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C3

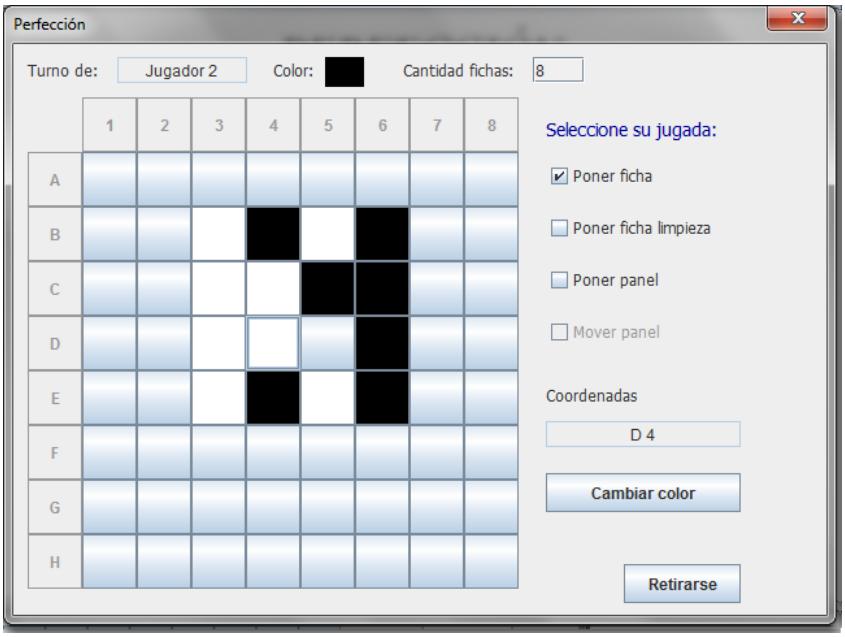


Resultado esperado:

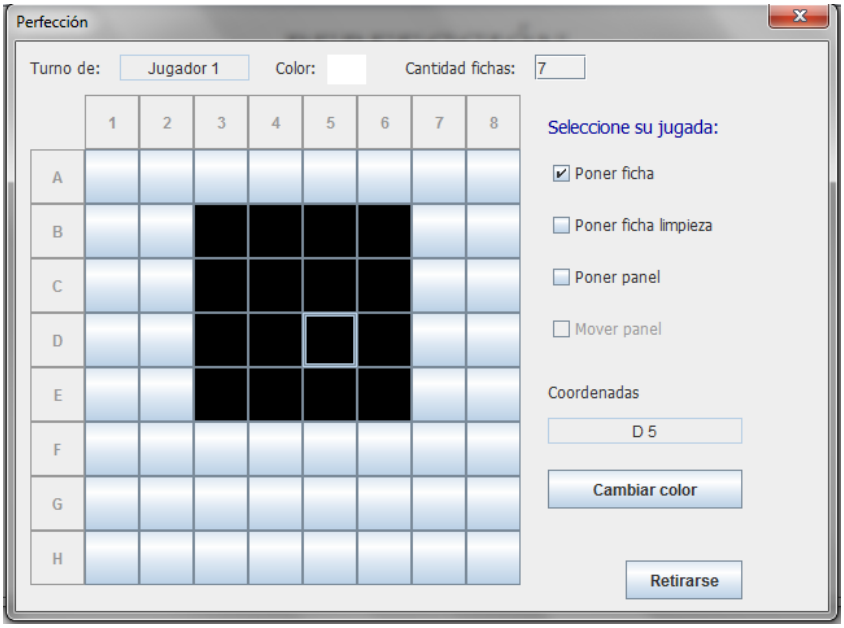


Resumen: Ok

Datos ingresados: Jugador 2 en la posición D5

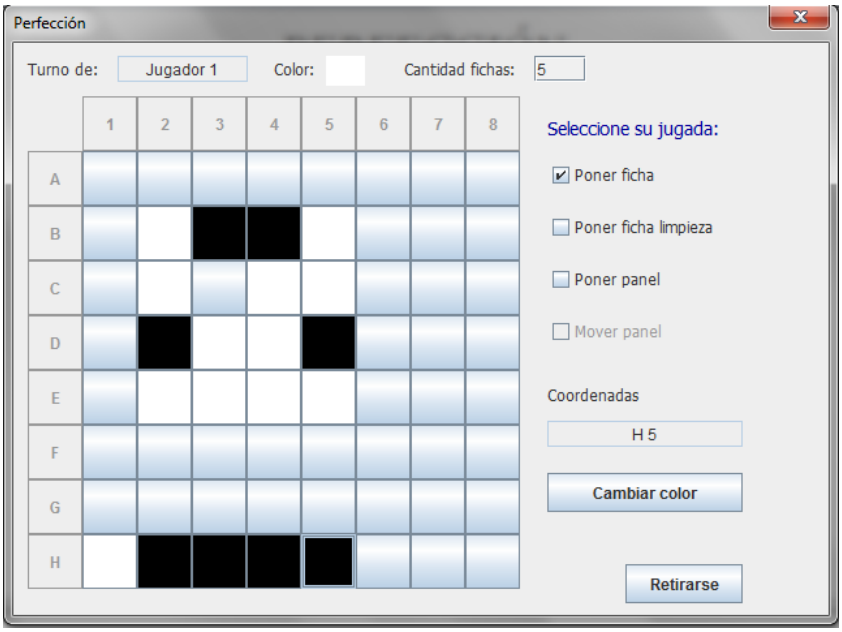


Resultado esperado:

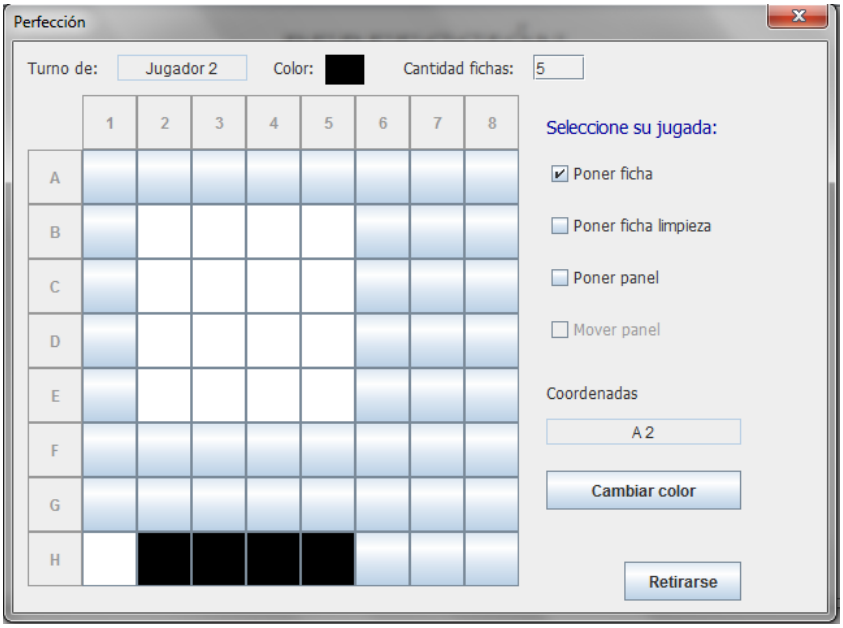


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C3



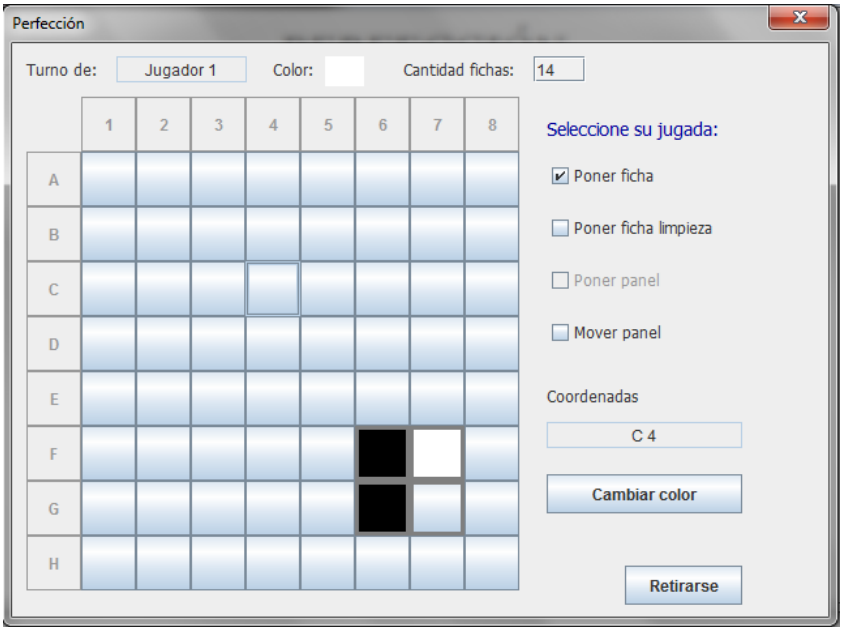
Resultado esperado:



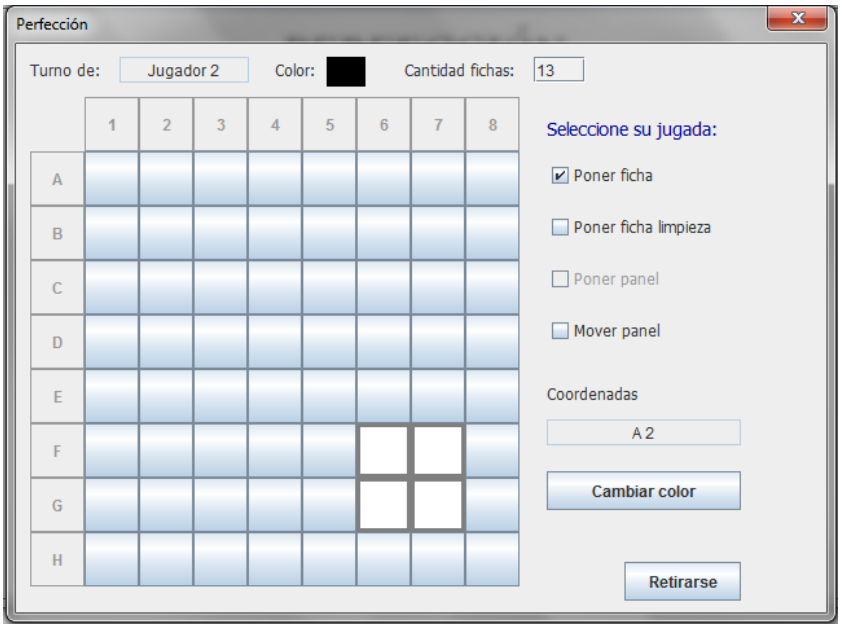
Resumen: Ok

Asimismo, consideremos también las siguientes pruebas para el tablero de 8x8 en el cual el panel de 2x2 ha sido colocado.

Datos ingresados: Jugador 1 en posición G7

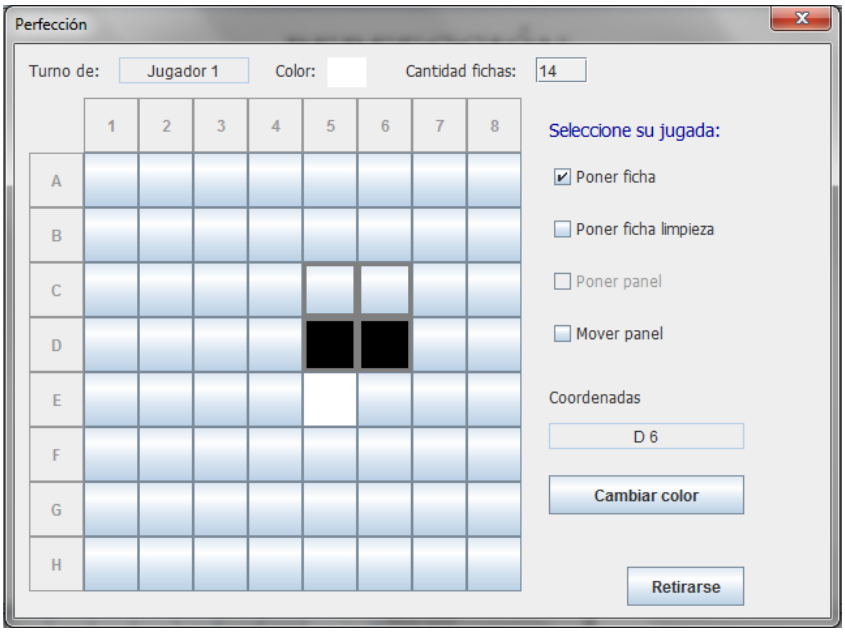


Resultado esperado:

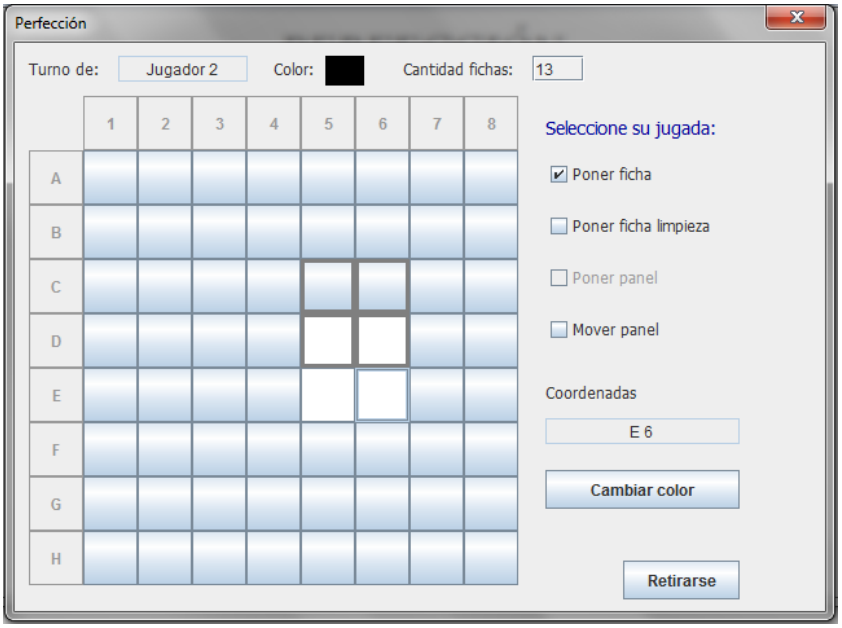


Resumen: Ok

Datos ingresados: Jugador 1 en la posición E6

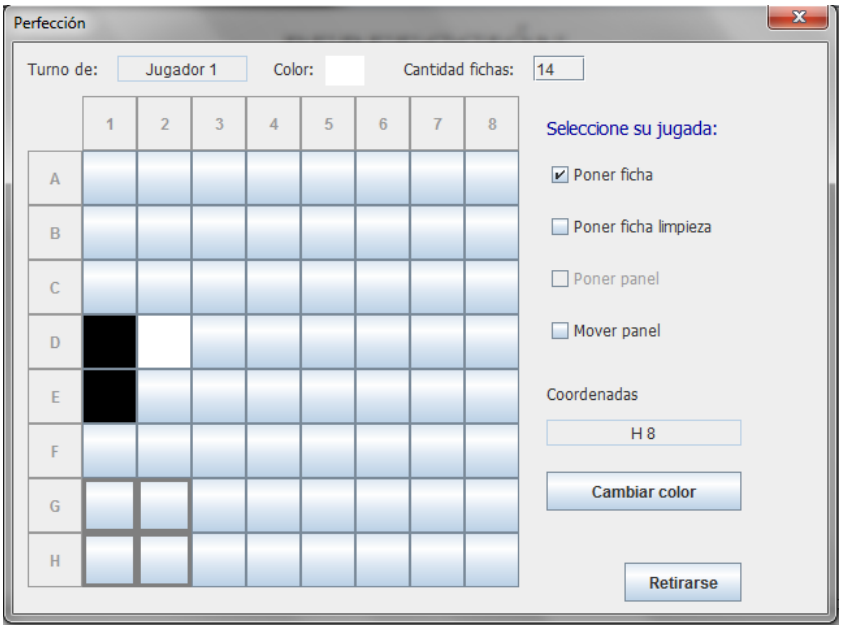


Resultado esperado:

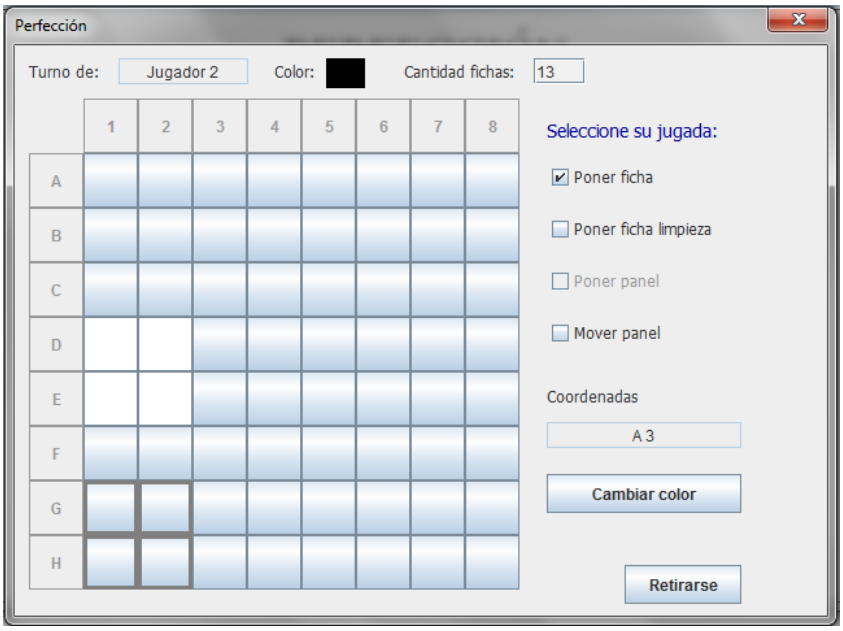


Resumen: Ok

Datos ingresados: Jugador 1 en la posición E2

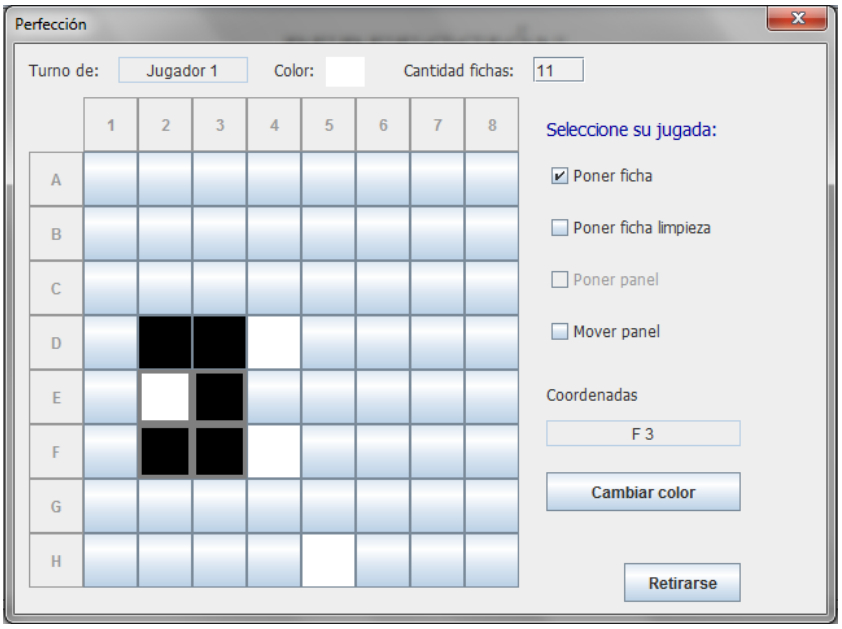


Resultado esperado:

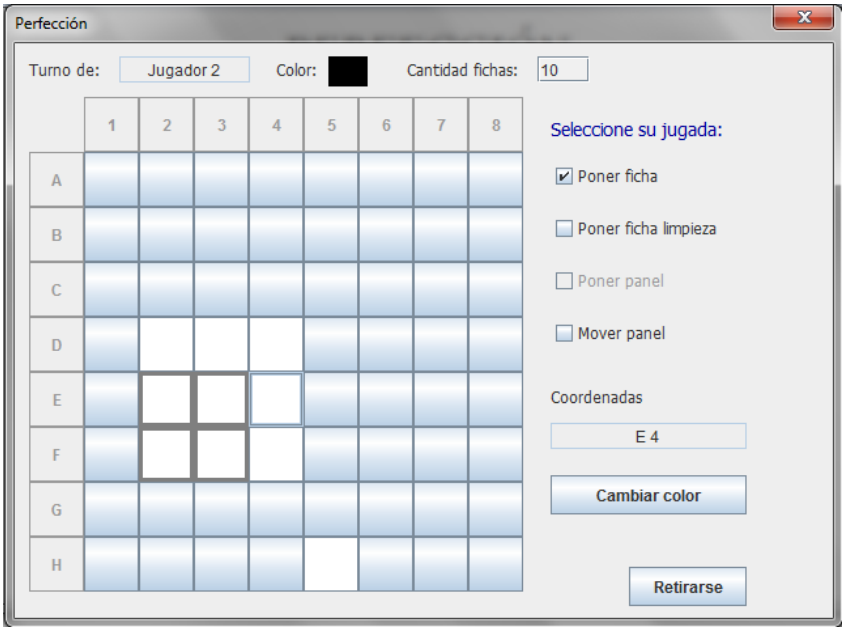


Resumen: Ok

Datos ingresados: Jugador 1 en la posición E4

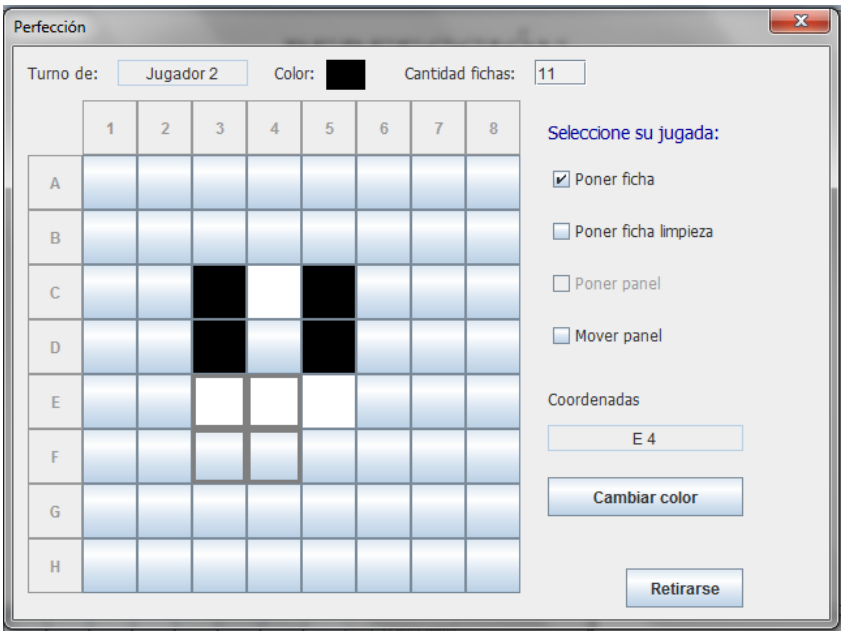


Resultado esperado:

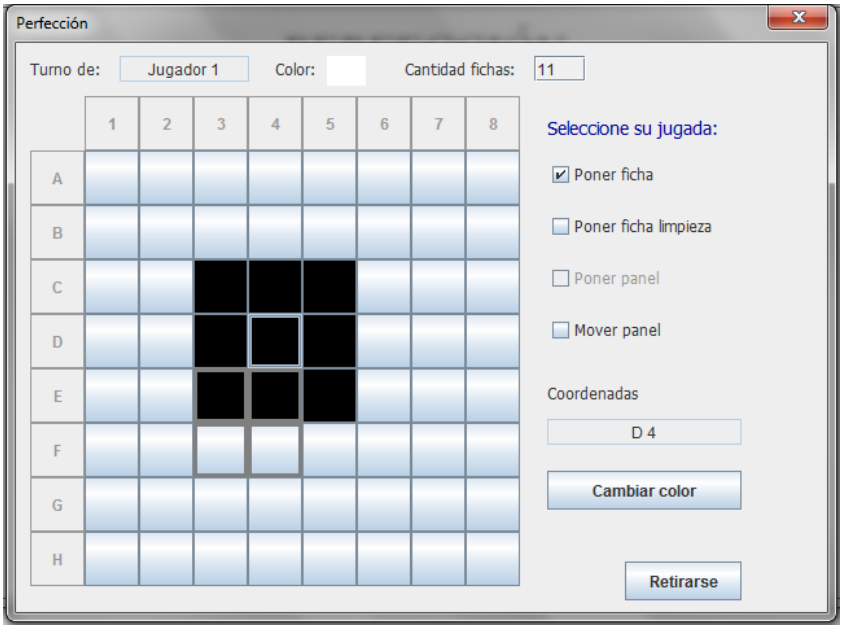


Resumen: Ok

Datos ingresados: Jugador 2 en la posición D4

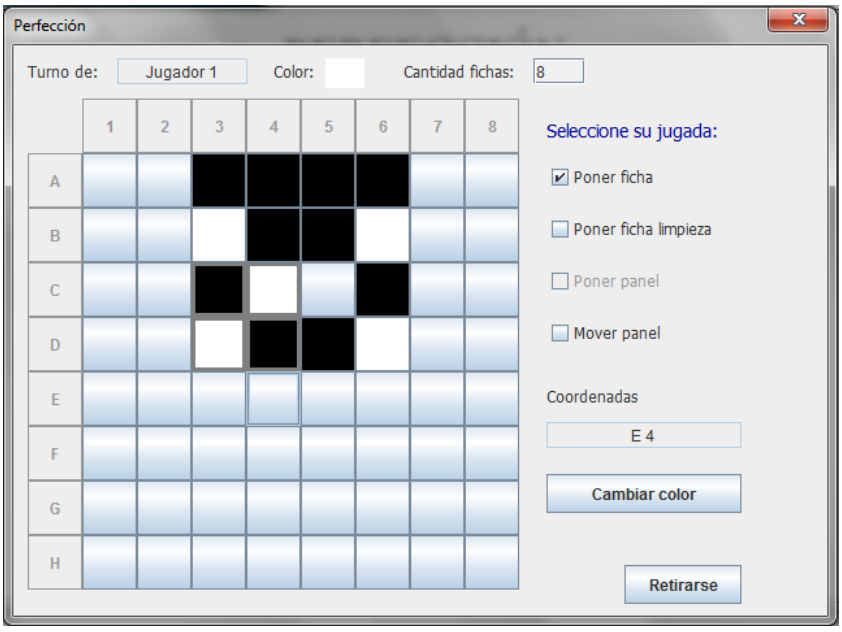


Resultado esperado:

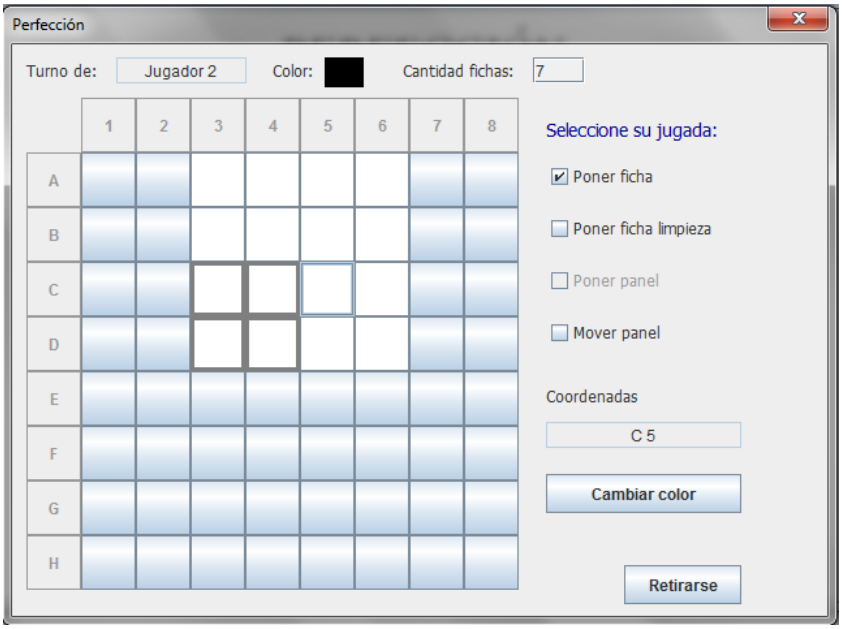


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C5

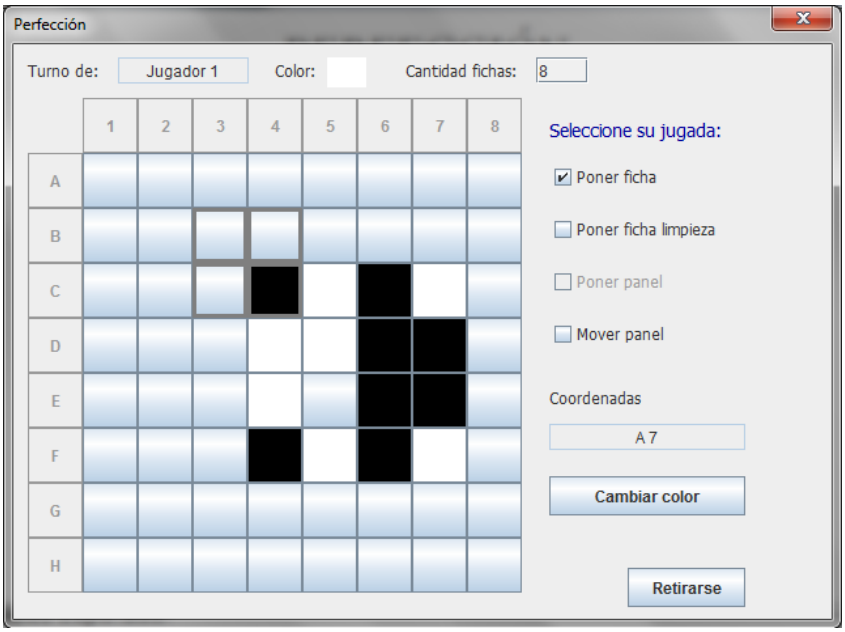


Resultado esperado:

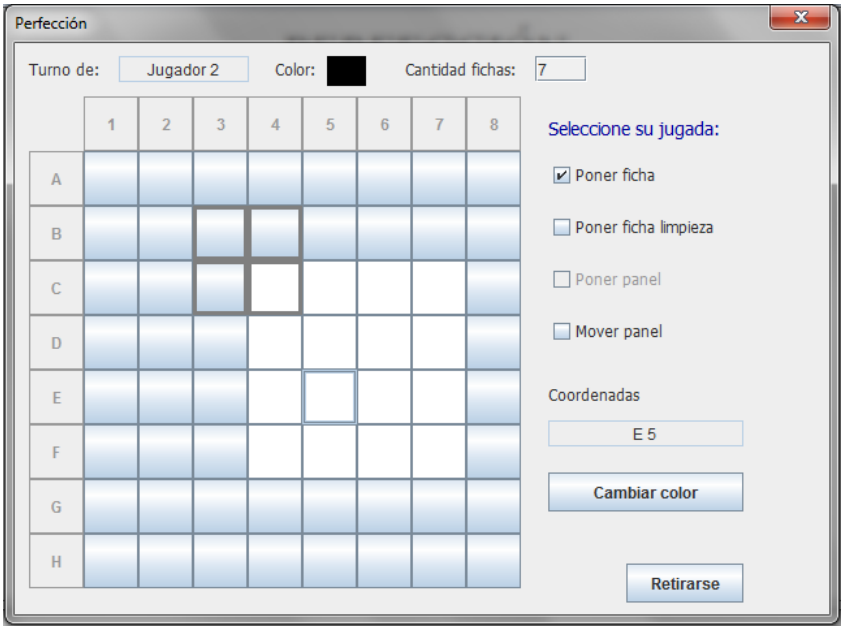


Resumen: Ok

Datos ingresados: Jugador 1 en la posición E5



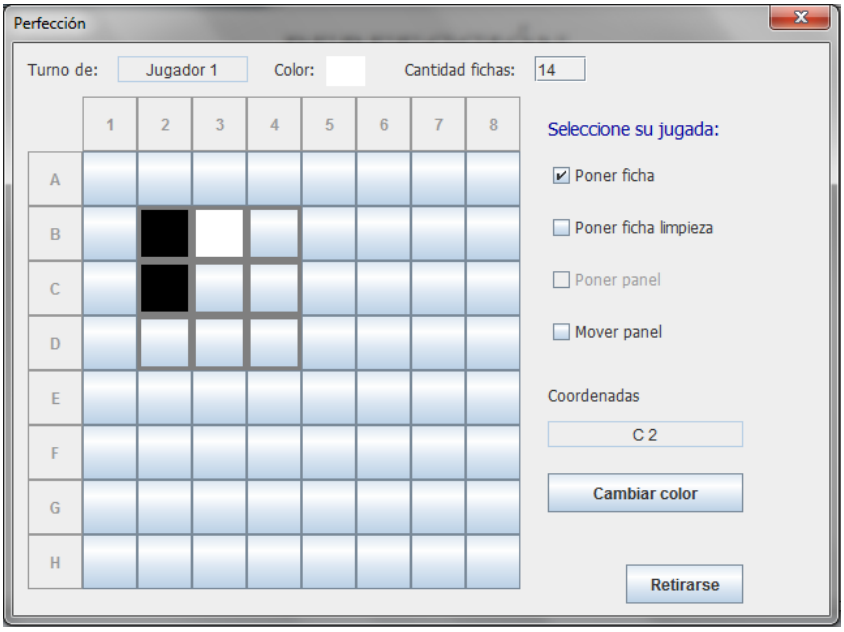
Resultado esperado:



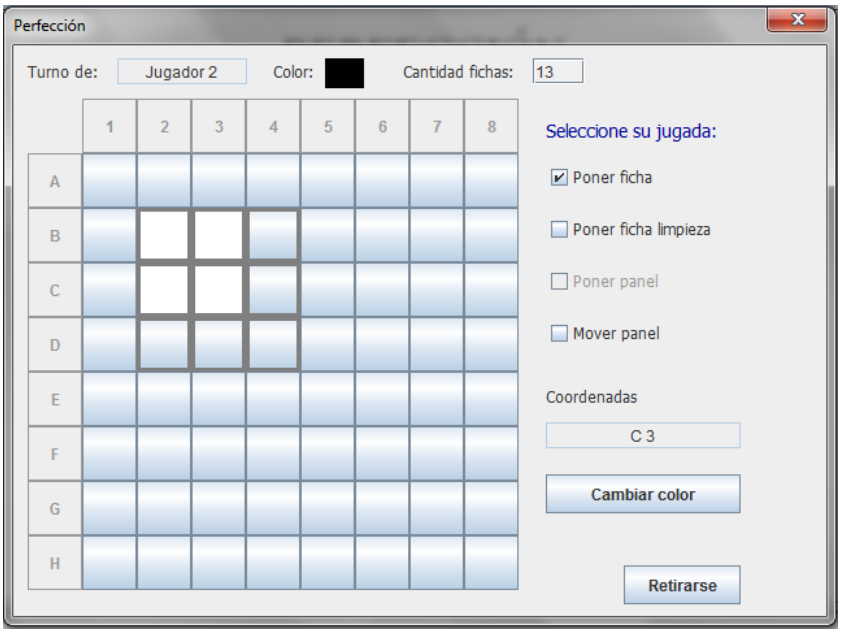
Resumen: Ok

Consideremos ahora las siguientes pruebas para el tablero de 8x8 en el cual el panel de 3x3 ha sido colocado.

Datos ingresados: Jugador 1 en la posición C3

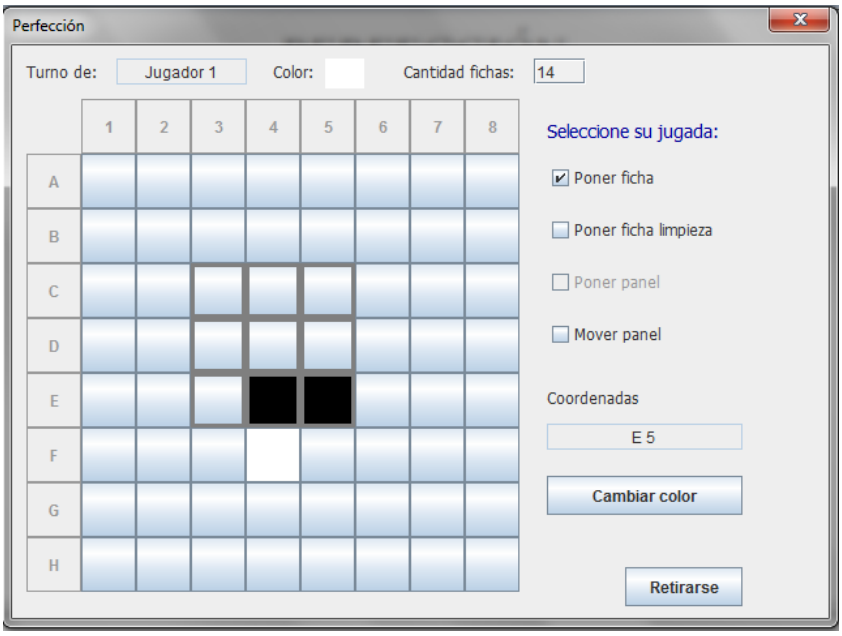


Resultado esperado:

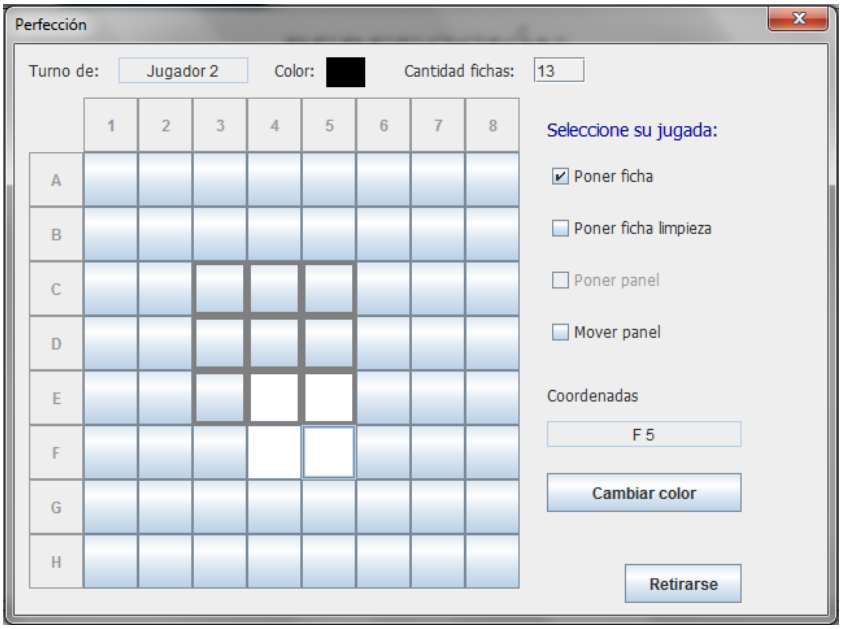


Resumen: Ok

Datos ingresados: Jugador 1 en la posición F5

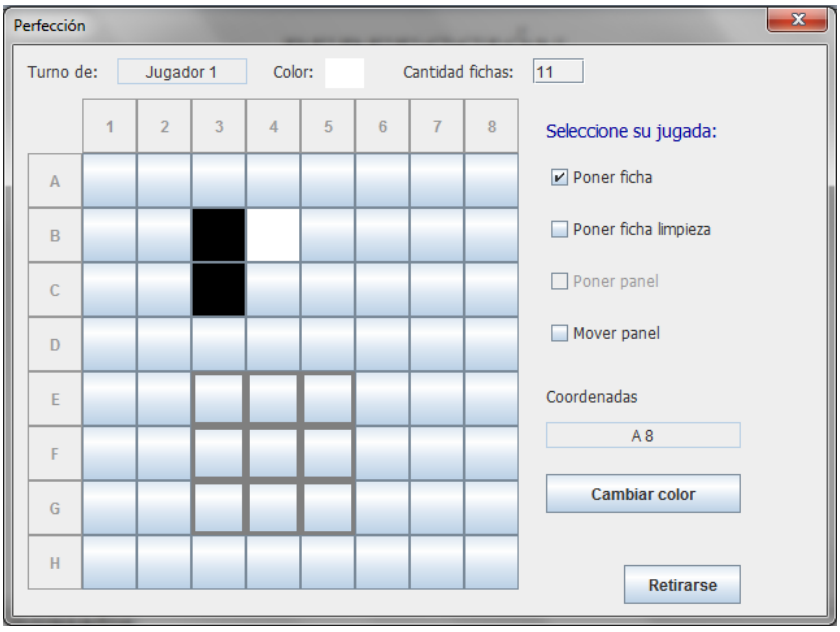


Resultado obtenido:

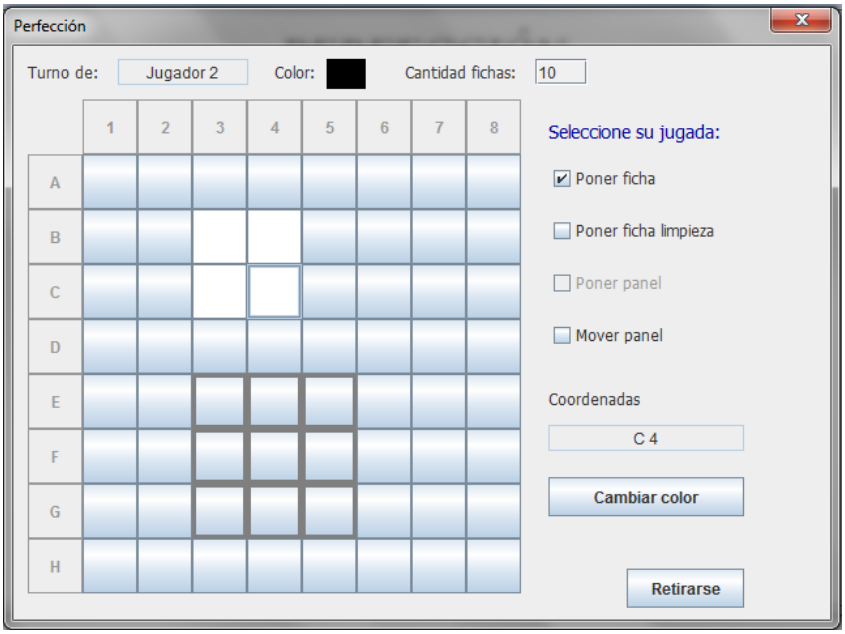


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C4

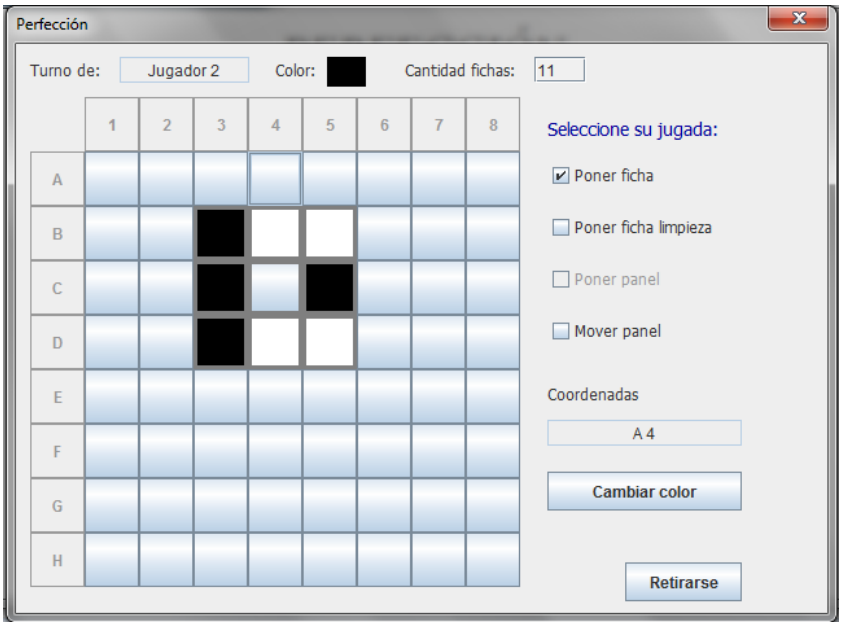


Resultado esperado:

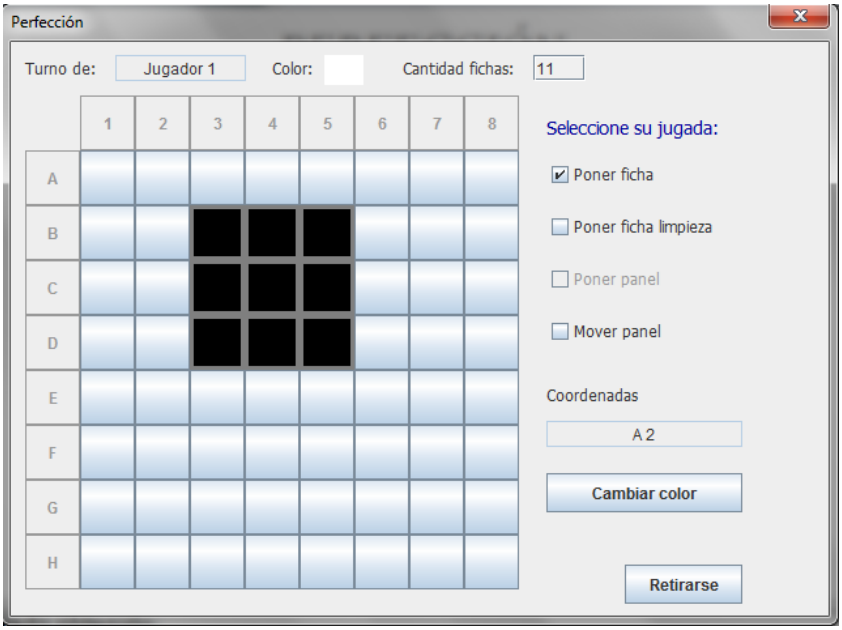


Resumen: Ok

Datos ingresados: Jugador 2 en la posición C4

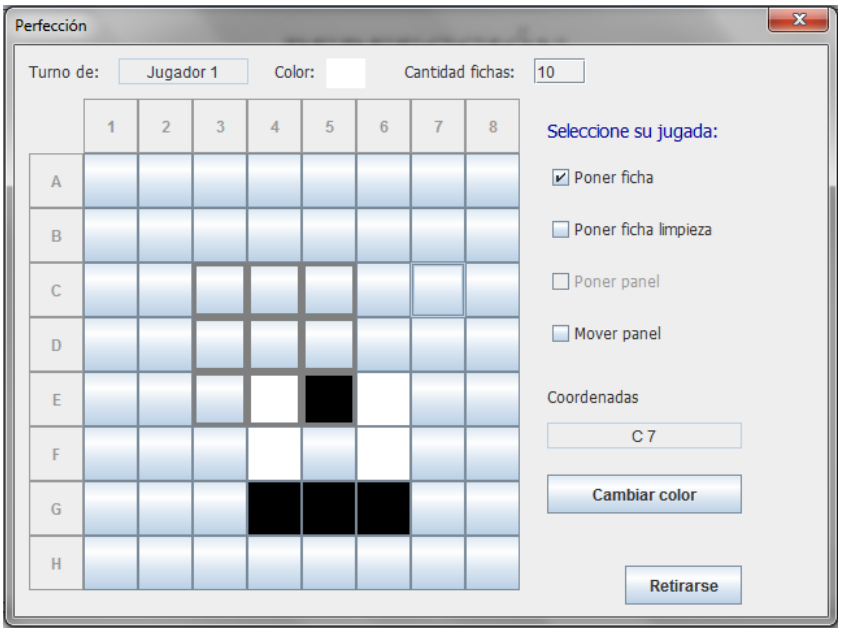


Resultado obtenido:

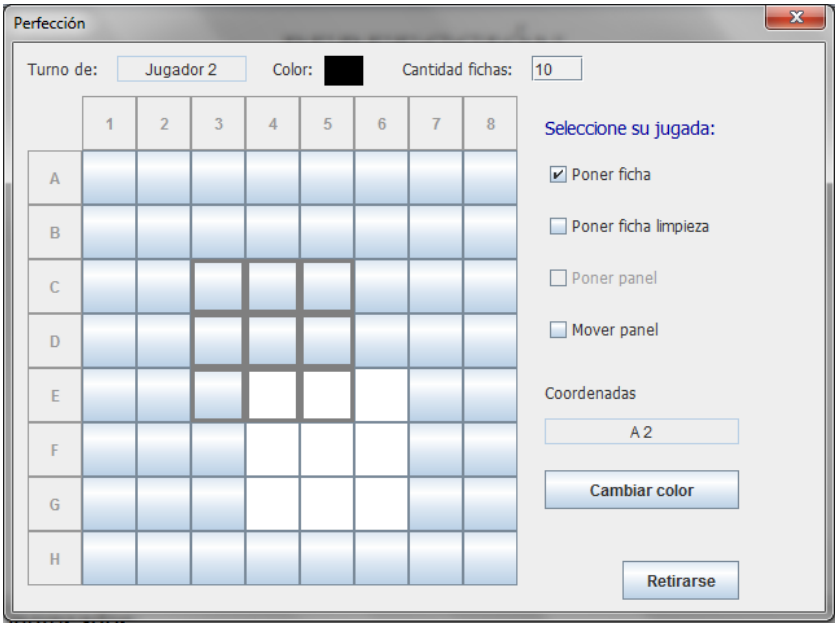


Resumen: Ok

Datos ingresados: Jugador 1 en la posición F5

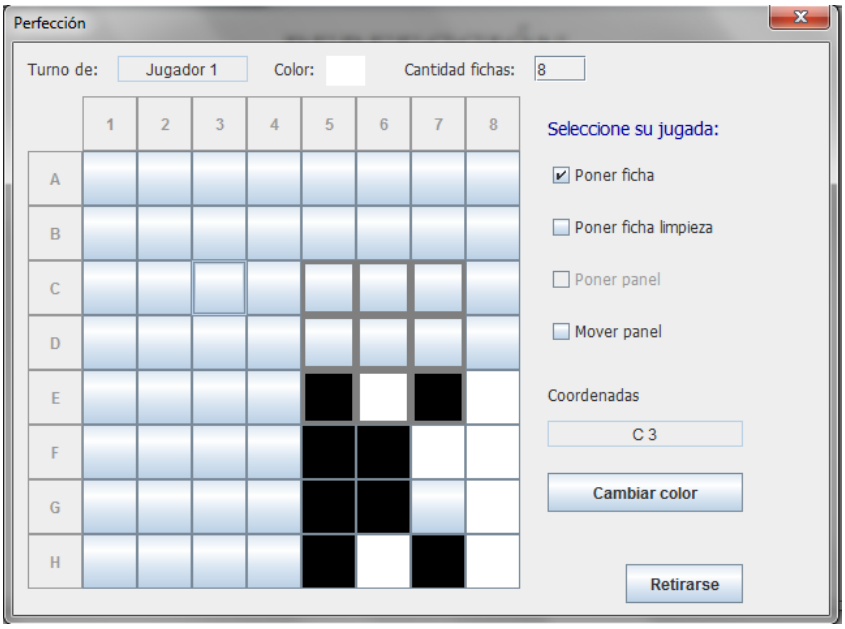


Resultado obtenido:

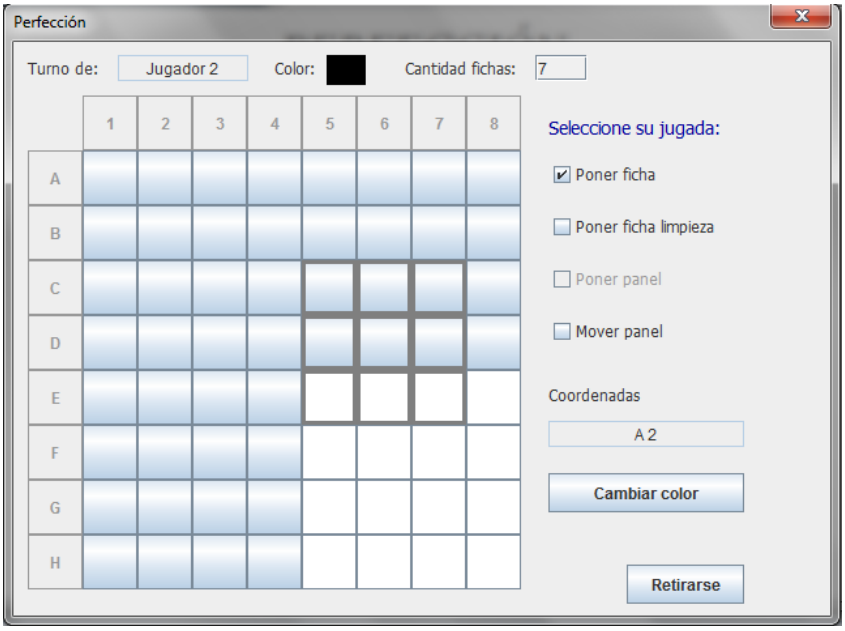


Resumen: Ok

Datos ingresados: Jugador 1 en la posición G7

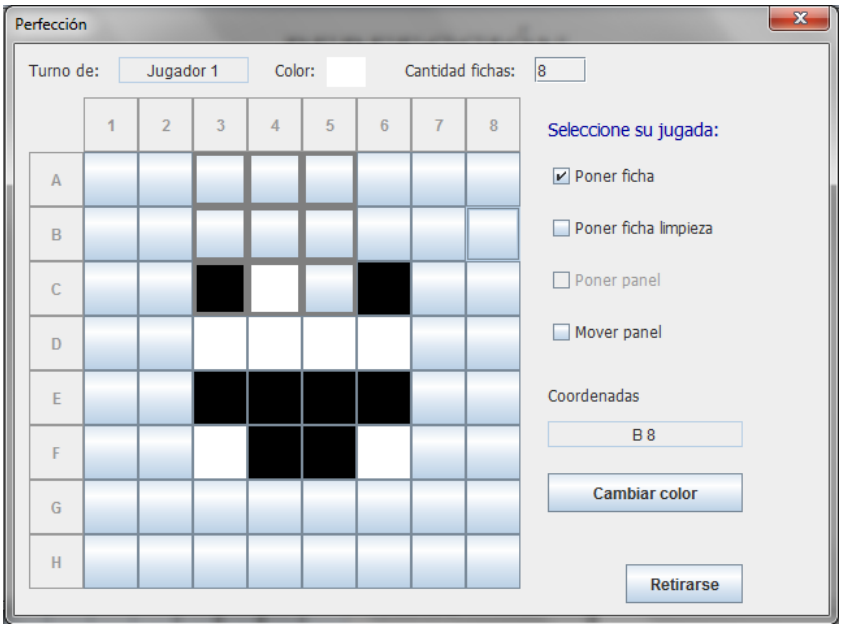


Resultado obtenido:

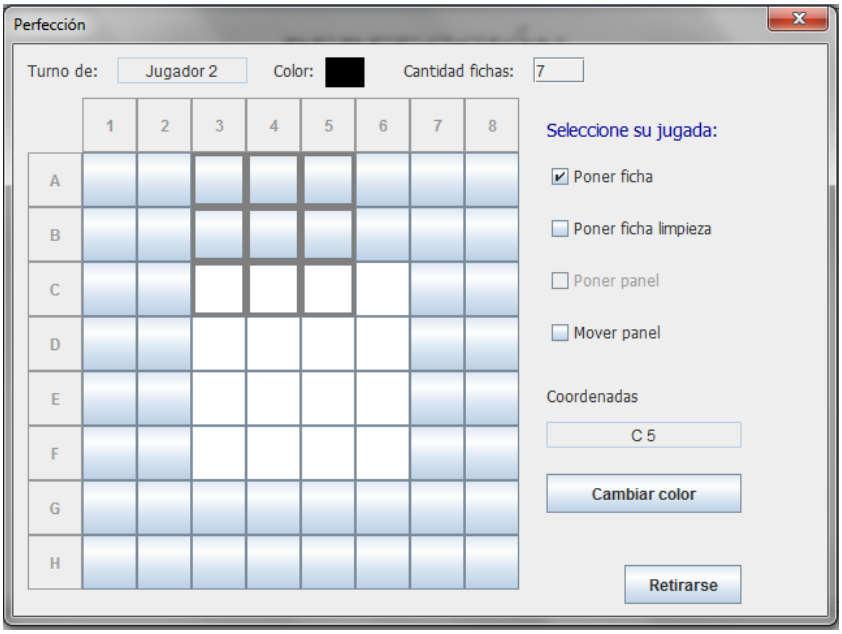


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C5



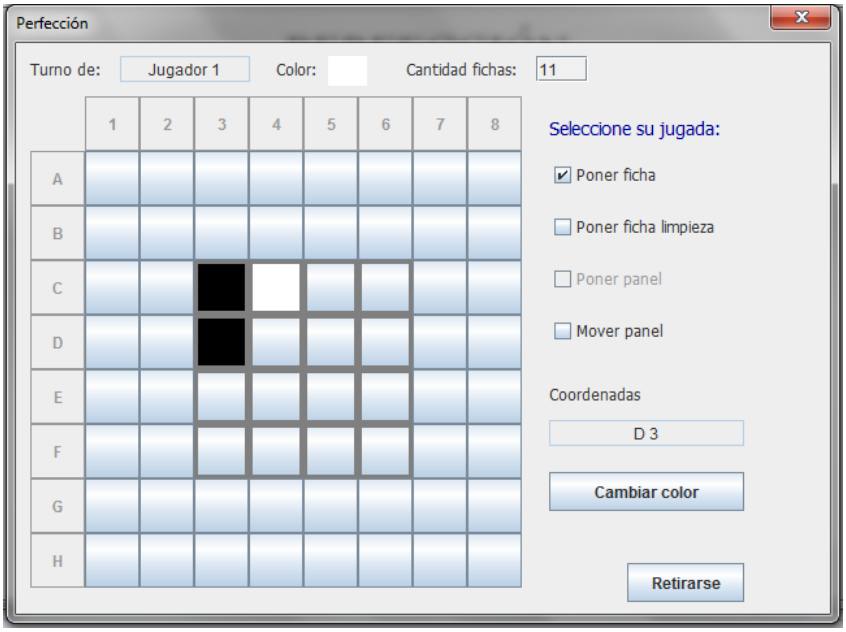
Resultado obtenido:



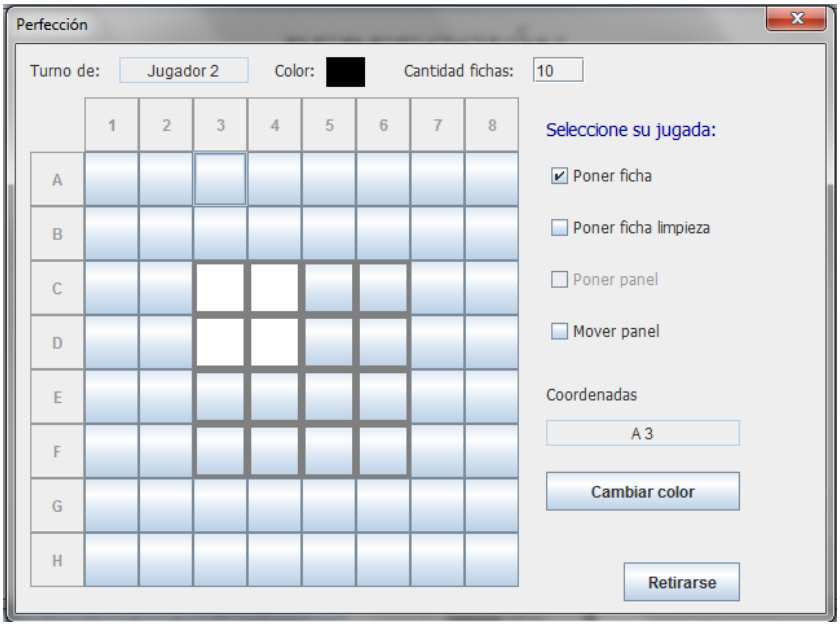
Resumen: Ok

Finalmente, las siguientes pruebas son para el tablero de 8x8 en el cual el panel de 4x4 ha sido colocado.

Datos ingresados: Jugador 1 en la posición D4

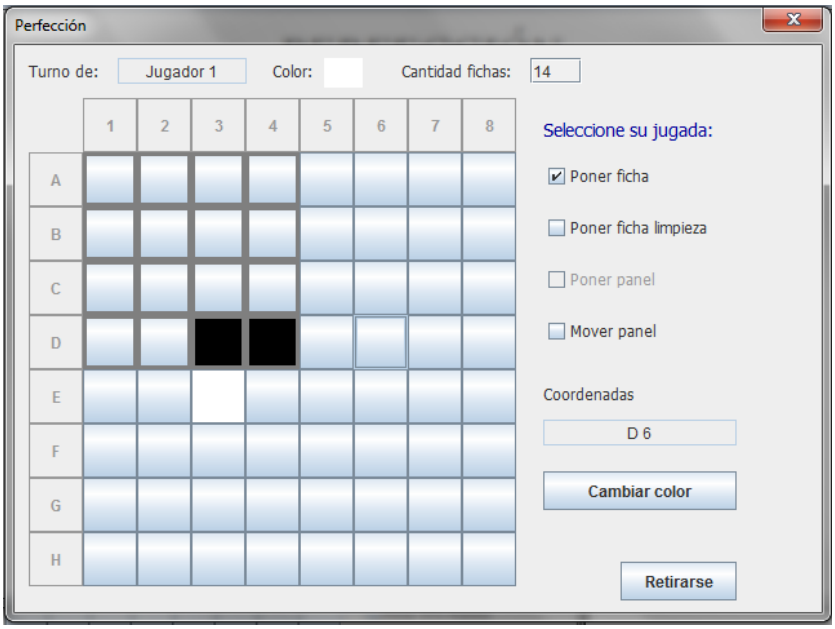


Resultado obtenido:

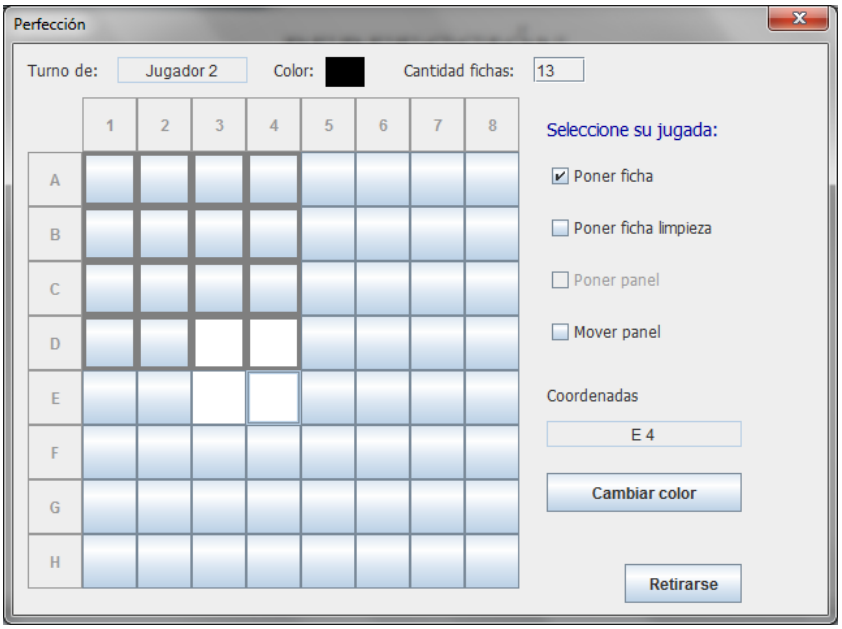


Resumen: Ok

Datos ingresados: Jugador 1 en la posición E4

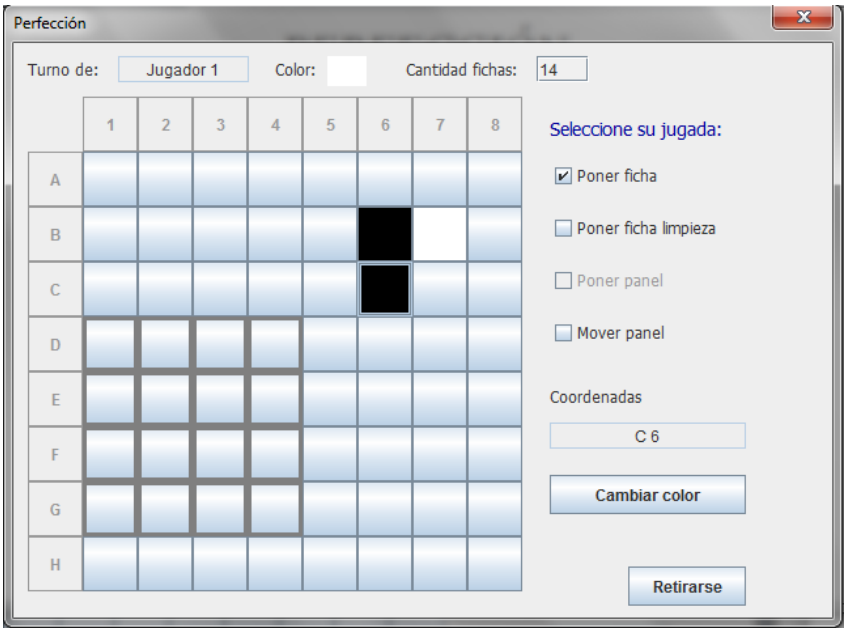


Resultado obtenido:

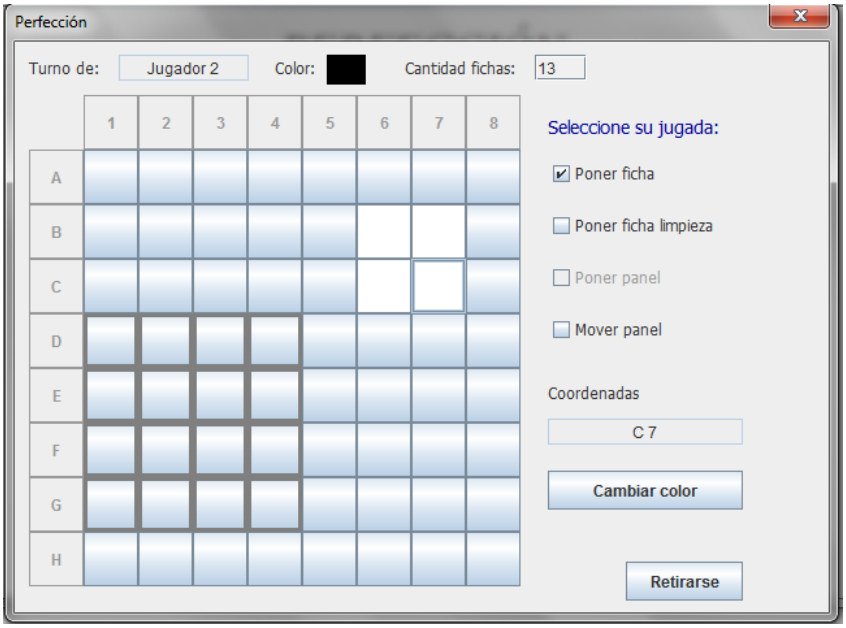


Resumen: Ok

Datos ingresados: Jugador 1 en la posición C7

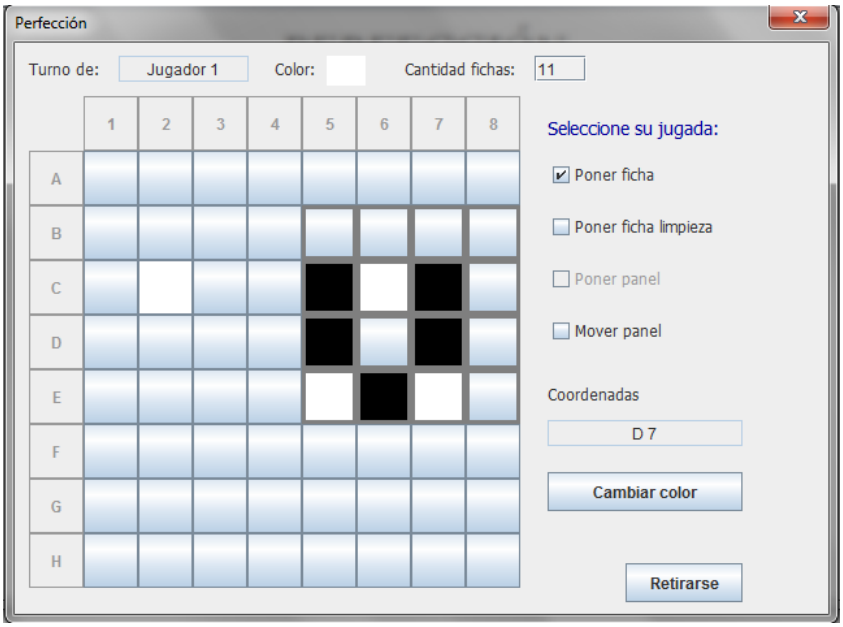


Resultado obtenido:

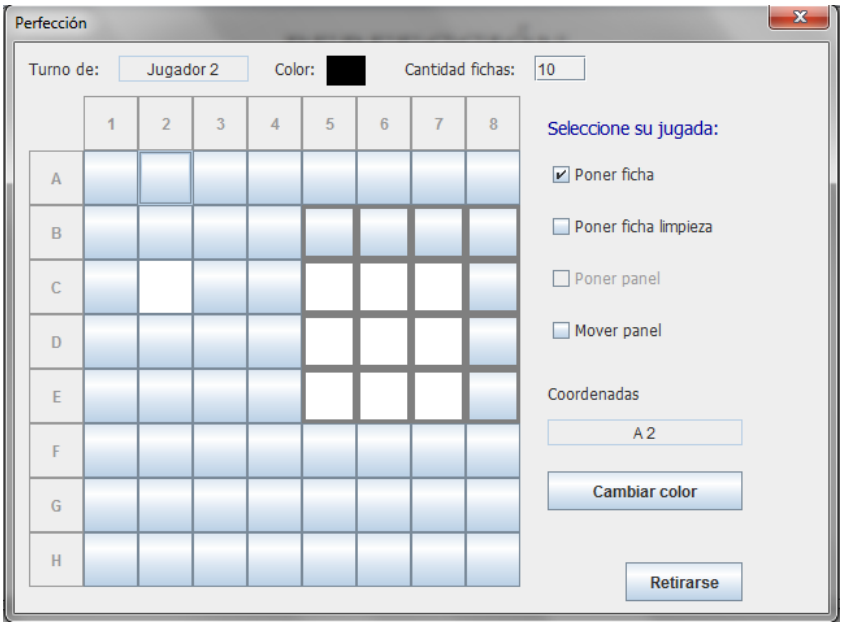


Resumen: Ok

Datos ingresados: Jugador 1 en la posición D6

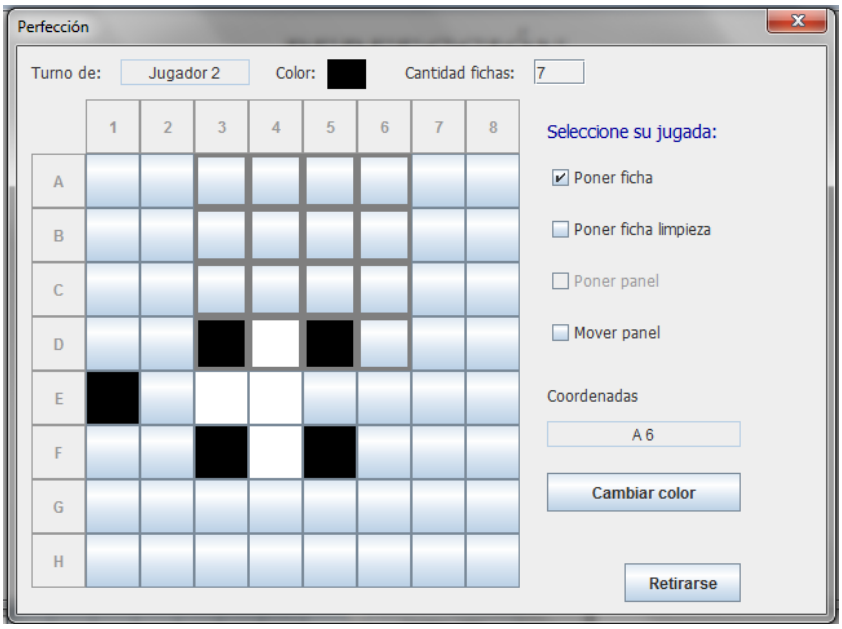


Resultado obtenido:

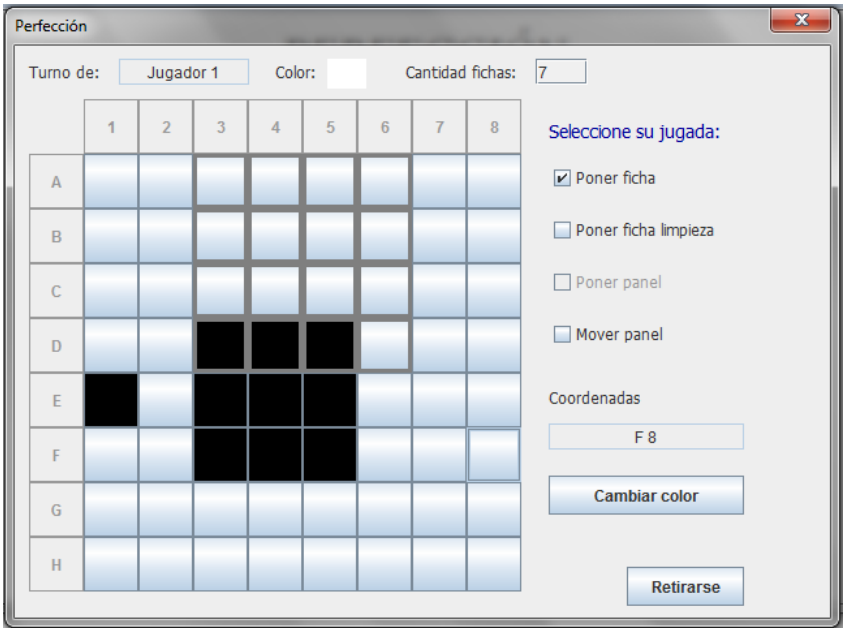


Resumen: Ok

Datos ingresados: Jugador 2 en la posición E5

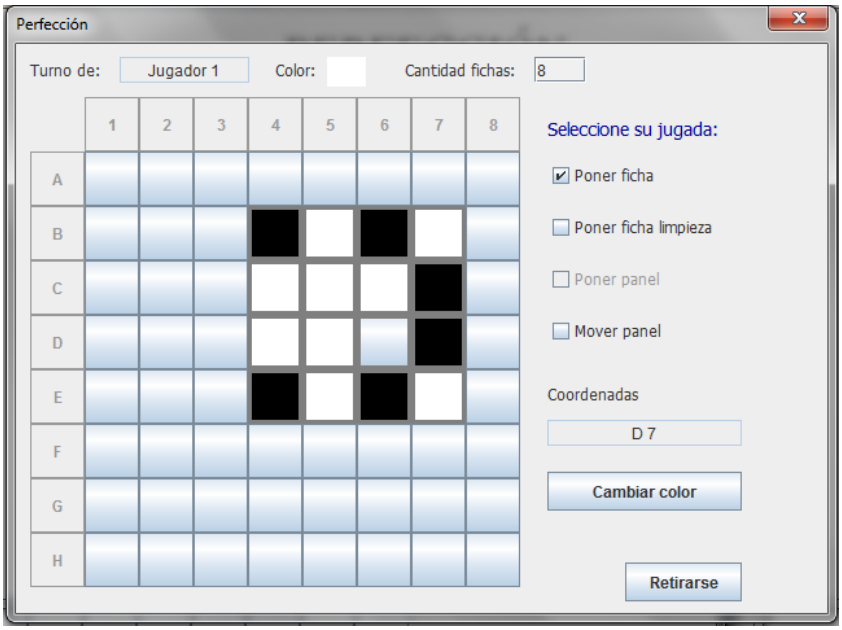


Resultado obtenido:

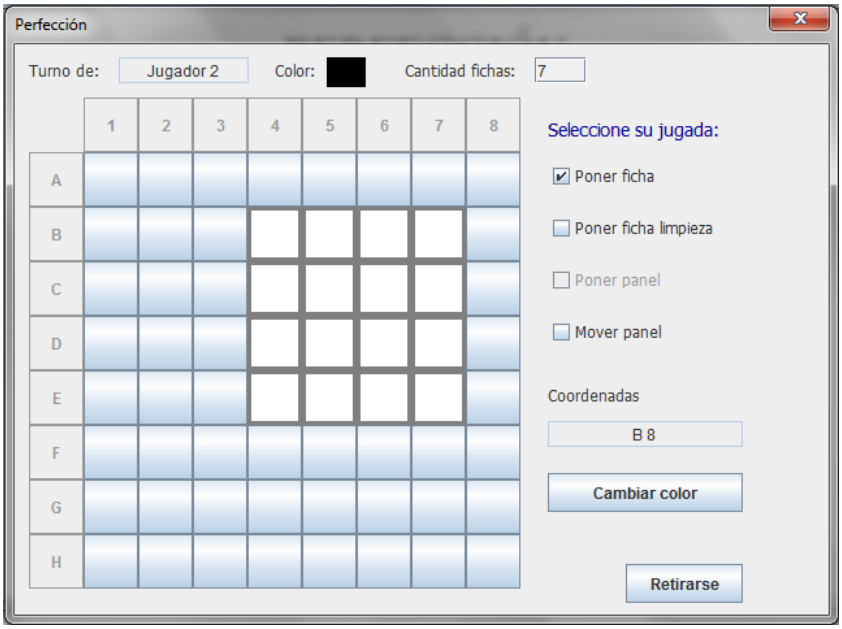


Resumen: Ok

Datos ingresados: Jugador 1 en la posición D6

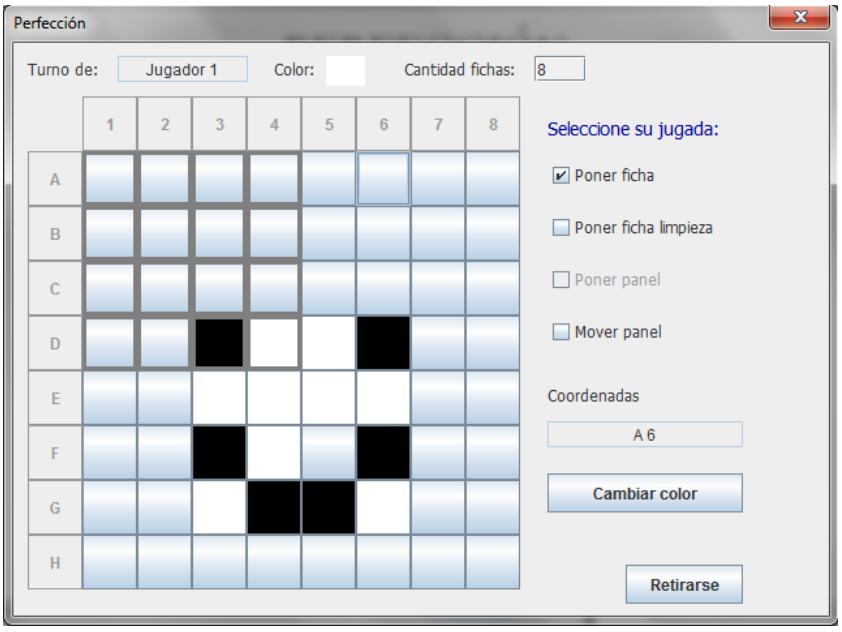


Resultado obtenido:

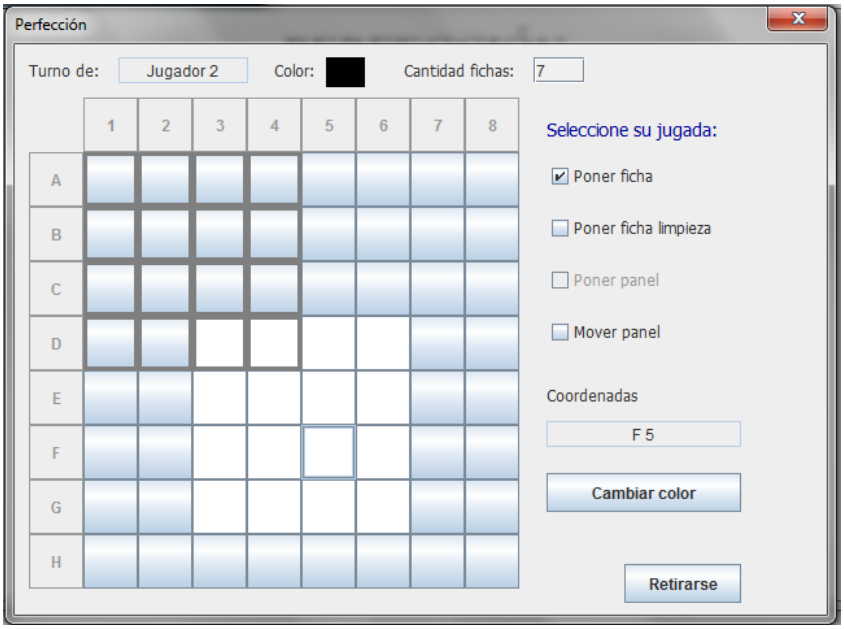


Resumen: Ok

Datos ingresados: Jugador 1 en la posición F5



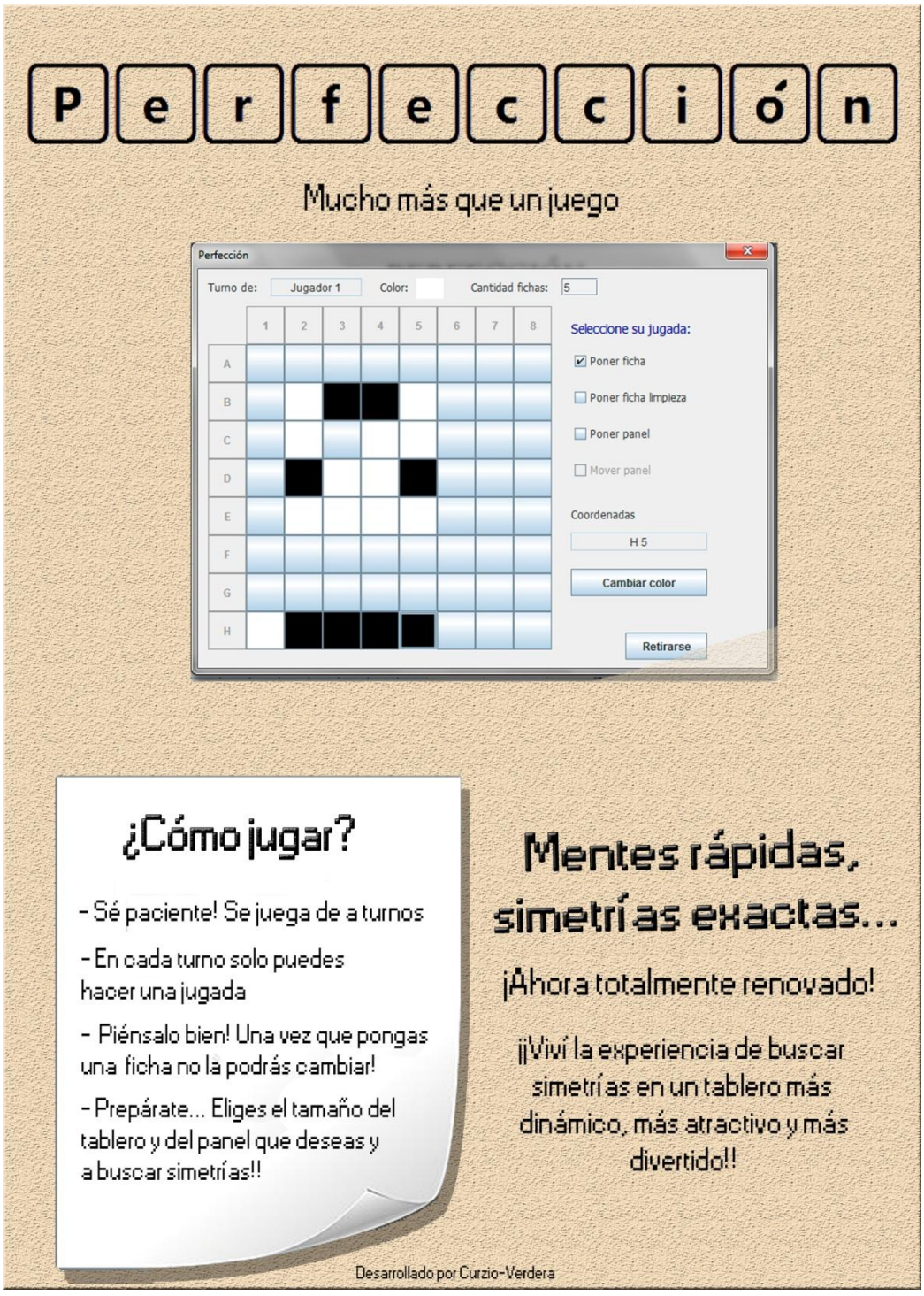
Resultado obtenido:



Resumen: Ok

Estos casos se cumplen análogamente para los tableros de 6x6 y 4x4.

Folleto publicitario:



Decisiones relevantes:

La partida sólo termina al acabarse las fichas de ambos jugadores o cuando uno de ellos se retira. Si a un jugador le quedan fichas para colocar se seguirá el juego.

En caso de que, al terminar la partida, la cantidad de fichas visibles de ambos jugadores sea la misma, el juego termina en empate y no se considera a ninguno como jugador ganador de esa partida en particular.

Si en el transcurso de la partida se cierra la ventana de juego porque se quiere terminar la misma, se hace un conteo de fichas visibles y se informa quién es el ganador. No se asume que ningún jugador se rinde.

Código:

Clase Ficha:

```
package dominio;

public class Ficha {

   //Atributos
   private Color color;

   public enum Color {

      B,
      N,
      noHay,
      asterisco
   }

   //Metodos de acceso y modificacion
   public Color getColor() {
      return color;
   }

   public void setColor(Color unColor) {
      this.color = unColor;
   }

   //Constructor
   public Ficha(Color unColor) {
      this.color = unColor;
   }

   @Override
   public String toString() {
      String retorno = " ";
      if (this.getColor().equals(Ficha.Color.B)) {
         retorno = "B";
      }
      if (this.getColor().equals(Ficha.Color.N)) {
         retorno = "N";
      }
      if (this.getColor().equals(Ficha.Color.asterisco)) {
         retorno = "*";
      }
      return retorno;
   }

   @Override
   public boolean equals(Object o) {
      return ((Ficha)o).getColor().equals(this.getColor());
```

```
    }
}
```

Clase Jugador:

```
package dominio;

import java.io.Serializable;

public class Jugador implements Comparable<Jugador> , Serializable {

    //Atributos
    private String nombre;
    private int edad;
    private String alias;
    private int cantPartidasGanadas;
    private int cantPartidasJugadas;

    // Métodos de acceso y modificación
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String unNombre) {
        this.nombre = unNombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int unaEdad) {
        this.edad = unaEdad;
    }

    public String getAlias() {
        return alias;
    }

    public void setAlias(String unAlias) {
        this.alias = unAlias;
    }

    public int getCantPartidasGanadas() {
        return cantPartidasGanadas;
    }

    public void setCantPartidasGanadas(int unaCantPartidasGanadas) {
        this.cantPartidasGanadas = unaCantPartidasGanadas;
    }
```

```
    public int getCantPartidasJugadas() {
        return cantPartidasJugadas;
    }

    public void setCantPartidasJugadas(int unaCantPartidasJugadas) {
        this.cantPartidasJugadas = unaCantPartidasJugadas;
    }

    //Constructor con parámetros
    public Jugador(String unNombre, int unaEdad, String unAlias, int unaCantPartidasGanadas,
int unaCantPartidasJugadas) {
        this.nombre = unNombre;
        this.edad = unaEdad;
        this.alias = unAlias;
        this.cantPartidasGanadas = unaCantPartidasGanadas;
        this.cantPartidasJugadas = unaCantPartidasJugadas;
    }

    //Constructor sin parámetros
    public Jugador() {
        this.nombre = "Sin nombre";
        this.edad = 0;
        this.alias = "Sin alias";
        this.cantPartidasGanadas = 0;
        this.cantPartidasJugadas = 0;
    }

    @Override
    public String toString() {
        return  " Alias: " + this.getAlias()
                + "  Partidas ganadas: " + this.getCantPartidasGanadas()
                + "  Partidas jugadas: " + this.getCantPartidasJugadas();
    }

    @Override
    public int compareTo(Jugador j) {
        int resultado = this.getAlias().compareTo(j.getAlias());
        return resultado;
    }

    @Override
    public boolean equals (Object o){
        return ((Jugador)o).getAlias().equals(this.getAlias());
    }

}
```

Clase Panel:

```
package dominio;

public class Panel {

   //Atributos
   private Ficha[][] panel;

   //Constructor con parámetros
   public Panel(int dimension) {
      this.panel = new Ficha[dimension][dimension];
      for (int f = 0; f < this.panel.length; f++) {
         for (int c = 0; c < this.panel.length; c++) {
            this.panel[f][c] = new Ficha(Ficha.Color.noHay);
         }
      }
   }

   public Ficha obtenerFichaPanel(int posFila, int posCol) {
      return panel[posFila][posCol];
   }

   public int dimPanel() {
      return panel.length;
   }

   public void ponerFichaPanel(int unaFila, int unaCol, Ficha ficha) {
      panel[unaFila][unaCol] = ficha;
   }
}
```

Clase Partida:

```
package dominio;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Partida {

   //Atributos
   private Jugador jugadorUno;
   private Jugador jugadorDos;
   private Jugador jugadorDeTurno;
   private Jugador jugadorPerdedor;
   private Tablero tablero;
   private int posFilaPanel;
   private int posColPanel;
   private Panel panel;
   private Ficha[][] matAux;
   private int cantidadFichasUno;
```

```
private int cantidadFichasDos;
private boolean haySimetria;
private java.awt.Color colorJugadorUno;
private java.awt.Color colorJugadorDos;
private boolean limpiezaUno;
private boolean limpiezaDos;

//Constructor con parámetros
public Partida(Jugador unJugadorUno, Jugador unJugadorDos,
      int unaDimTablero, int unaDimPanel, int cantFichasUno, int cantFichasDos,
      java.awt.Color unColorJugadorUno, java.awt.Color unColorJugadorDos) {
   this.jugadorUno = unJugadorUno;
   this.jugadorDos = unJugadorDos;
   this.tablero = new Tablero(unaDimTablero);
   this.panel = new Panel(unaDimPanel);
   this.posFilaPanel = -20;
   this.posColPanel = -20;
   this.matAux = new Ficha[unaDimTablero][unaDimTablero];
   this.cantidadFichasUno = cantFichasUno;
   this.cantidadFichasDos = cantFichasDos;
   this.jugadorDeTurno = jugadorUno;
   this.haySimetria = false;
   this.colorJugadorUno = unColorJugadorUno;
   this.colorJugadorDos = unColorJugadorDos;
   limpiezaUno = false;
   limpiezaDos = false;
   for (int i = 0; i < matAux.length; i++) {
      for (int j = 0; j < matAux.length; j++) {
         matAux[i][j] = new Ficha(Ficha.Color.noHay);
      }
   }
}

public int getDimension() {
   return tablero.largoTablero();
}

private boolean jugadaDentroRango(String jugada) {
   boolean correcto = false;
   Pattern pat;
   jugada = jugada.substring(1).toUpperCase();
   if (tablero.largoTablero() == 4) {
      pat = Pattern.compile("[A-D][1-4]");

   } else if (tablero.largoTablero() == 6) {
      pat = Pattern.compile("[A-F][1-6]");
   } else {
      pat = Pattern.compile("[A-H][1-8]");
   }
   Matcher mat = pat.matcher(jugada);
   if (mat.matches()) {
      correcto = true;
```

```
        }
        return correcto;
    }

    private boolean validarJugada(String jugada) {
        jugada = jugada.toUpperCase();
        char primerComando = jugada.charAt(0);
        boolean esValido = false;
        if (jugada.equals("X")) {
            esValido = true;
        } else if (jugada.length() == 3 && Character.isLetter(jugada.charAt(0))
                && Character.isLetter(jugada.charAt(1)) && Character.isDigit(jugada.charAt(2))) {
            char segundoComando = jugada.charAt(1);
            if (primerComando == 'J' && jugadaDentroRango(jugada)) {
                esValido = true;
            } else if (primerComando == 'P' && jugadaDentroRango(jugada)) {
                esValido = true;
            } else if ((primerComando == 'M') && (segundoComando == 'I'
                    || segundoComando == 'D' || segundoComando == 'A'
                    || segundoComando == 'B')) {
                esValido = true;
            } else if (primerComando == 'L') {
                esValido = true;
            }
        }
        return esValido;
    }

    private String validarFicha(int posFila, int posCol) {
        //!(si está vacía y la posFila y posCol están dentro del rango)
        String retorno = "ok";
        if (!((posFila < tablero.largoTablero()) && (posCol < tablero.largoTablero())
                && posFila >= 0 && posCol >= 0
                && obtenerFicha(posFila, posCol).getColor() == Ficha.Color.noHay)) {
            retorno = "Posición inválida ";
        }
        return retorno;
    }

    //<editor-fold defaultstate="collapsed" desc="Simetrias">
    private boolean hacerSimetrias(int posFila, int posCol, Jugador jugadorActual, Jugador
jugadorUno) {
        //simetrías 4x4
        for (int i = posFila - 3; i <= posFila; i++) {
            for (int j = posCol - 3; j <= posCol; j++) {
                if (buscarSimetriasCuatroH(i, j)) {
                    cambiarFichasDeColorAux(i, j, i + 3, j + 3);
                    haySimetria = true;
                }
                if (buscarSimetriasCuatroV(i, j)) {
                    cambiarFichasDeColorAux(i, j, i + 3, j + 3);
                    haySimetria = true;
```

```
            }
        }
    }
    //simetrías 3x3
    for (int i = posFila - 2; i <= posFila; i++) {
        for (int j = posCol - 2; j <= posCol; j++) {
            if (buscarSimetriasTresH(i, j)) {
                cambiarFichasDeColorAux(i, j, i + 2, j + 2);
                haySimetria = true;
            }
            if (buscarSimetriasTresV(i, j)) {
                cambiarFichasDeColorAux(i, j, i + 2, j + 2);
                haySimetria = true;
            }
        }
    }
    //simetrías 2x2
    for (int i = posFila - 1; i <= posFila; i++) {
        for (int j = posCol - 1; j <= posCol; j++) {
            if (buscarSimetriasDosH(i, j)) {
                cambiarFichasDeColorAux(i, j, i + 1, j + 1);
                haySimetria = true;
            }
            if (buscarSimetriasDosV(i, j)) {
                cambiarFichasDeColorAux(i, j, i + 1, j + 1);
                haySimetria = true;
            }
        }
    }
    return haySimetria;
}

private boolean buscarSimetriasTresV(int fi, int ci) {
    boolean ok = hayFichasEnMat(fi, ci);
    for (int i = fi; i <= fi + 2 && ok; i++) {
        if (comparaFicha(i, ci, i, ci + 2)) {
            ok = false;
        }
    }
    return ok;
}

private boolean buscarSimetriasTresH(int fi, int ci) {
    boolean ok = hayFichasEnMat(fi, ci);
    for (int i = ci; i <= ci + 2 && ok; i++) {
        if (comparaFicha(fi, i, fi + 2, i)) {
            ok = false;
        }
    }
    return ok;
}
```

```
private boolean hayFichasEnMat(int fi, int ci) {
   boolean retorno = true;
   for (int i = fi; i <= fi + 2; i++) {
      for (int j = ci; j <= ci + 2; j++) {
         if (obtenerFicha(i, j).getColor().equals(Ficha.Color.noHay)) {
            retorno = false;
         }
      }
   }

   return retorno;
}

private boolean buscarSimetriasDosH(int fi, int ci) {
   boolean ok = true;
   for (int i = ci; i <= ci + 1 && ok; i++) {
      if (comparaFicha(fi, i, fi + 1, i)) {
         ok = false;
      }
   }
   return ok;
}

private boolean buscarSimetriasDosV(int fi, int ci) {
   boolean ok = true;
   for (int i = fi; i <= fi + 1 && ok; i++) {
      if (comparaFicha(i, ci, i, ci + 1)) {
         ok = false;
      }
   }
   return ok;
}

private boolean buscarSimetriasCuatroV(int fi, int ci) {
   boolean ok = true;
   for (int i = fi; i <= fi + 3 && ok; i++) {
      if (comparaFicha(i, ci, i, ci + 3) || comparaFicha(i, ci + 1, i, ci + 2)) {
         ok = false;
      }
   }
   return ok;
}

private boolean buscarSimetriasCuatroH(int fi, int ci) {
   boolean ok = true;
   for (int i = ci; i <= ci + 3 && ok; i++) {
      if (comparaFicha(fi, i, fi + 3, i) || comparaFicha(fi + 1, i, fi + 2, i)) {
         ok = false;
      }
   }
   return ok;
}
```

```
    //</editor-fold>

    private boolean comparaFicha(int filaUno, int colUno, int filaDos, int colDos) {
        boolean ok = !(obtenerFicha(filaUno, colUno).getColor().equals(obtenerFicha(filaDos,
colDos).getColor()));
        boolean ok2 = obtenerFicha(filaUno, colUno).getColor().equals(Ficha.Color.noHay);
        return ok || ok2;
    }

    private void cambiarFichasDeColor() {
        for (int i = 0; i < matAux.length; i++) {
            for (int j = 0; j < matAux.length; j++) {
                if (matAux[i][j].getColor().equals(Ficha.Color.asterisco)) {
                    if (jugadorDeTurno.equals(jugadorUno)) {
                        obtenerFicha(i, j).setColor(Ficha.Color.B);
                    } else {
                        obtenerFicha(i, j).setColor(Ficha.Color.N);
                    }
                }
            }
        }
    }

    private void cambiarFichasDeColorAux(int fi, int ci, int ff, int cf) {
        for (int f = fi; f <= ff; f++) {
            for (int c = ci; c <= cf; c++) {
                if (Ficha.Color.N.equals(obtenerFicha(f, c).getColor())
                        || Ficha.Color.B.equals(obtenerFicha(f, c).getColor())) {
                    matAux[f][c].setColor(Ficha.Color.asterisco);
                }
            }
        }
    }

    private String moverPanel(char direccion, int cantMovimiento) {
        String retorno = "El movimiento del panel no es válido ";
        int posFPanel = posFilaPanel;
        int posCPanel = posColPanel;
        if (direccion == 'A') {
            posFPanel -= cantMovimiento + 1;
        }
        if (direccion == 'B') {
            posFPanel += cantMovimiento + 1;
        }
        if (direccion == 'I') {
            posCPanel -= cantMovimiento + 1;
        }
        if (direccion == 'D') {
            posCPanel += cantMovimiento + 1;
        }
        if (cantMovimiento != -1 && ponerPanelValido(posFPanel, posCPanel)) {
            posFilaPanel = posFPanel;
```

```
        posColPanel = posCPanel;
        retorno = "ok";
    }
    return retorno;
}

private String ponerPanel(int unaFila, int unaCol) {
    String retorno = "Posición del panel inválida";
    if (ponerPanelValido(unaFila, unaCol) && !(posColPanel >= 0 && posFilaPanel >= 0)) {
        posFilaPanel = unaFila;
        posColPanel = unaCol;
        retorno = "ok";
    }
    return retorno;
}

private boolean ponerPanelValido(int posFila, int posCol) {
    boolean retorno = false;
    int dentroRango = tablero.largoTablero() - panel.dimPanel();
    if (posFila >= 0 && posCol >= 0 && posFila <= dentroRango
            && posCol <= dentroRango) {
        retorno = true;
    }
    return retorno;
}

public boolean fichaDentroPanel(int posFila, int posCol) {
    boolean retorno = false;
    if (posFila >= posFilaPanel && posCol >= posColPanel
            && posFila <= posFilaPanel + panel.dimPanel() - 1
            && posCol <= posColPanel + panel.dimPanel() - 1) {
        retorno = true;
    }
    return retorno;
}

public void cambiarTurno() {
    if (jugadorDeTurno.equals(jugadorUno)) {
        jugadorDeTurno = jugadorDos;
    } else {
        jugadorDeTurno = jugadorUno;
    }
}

private Jugador conteoFichas() {
    Jugador jugadorGano = new Jugador();
    int cantNegras = 0;
    int cantBlancas = 0;
    for (int i = 0; i < tablero.largoTablero(); i++) {
        for (int j = 0; j < tablero.largoTablero(); j++) {
            if (obtenerFicha(i, j).getColor().equals(Ficha.Color.B)) {
                cantBlancas++;
```

```
            }
            if (obtenerFicha(i, j).getColor().equals(Ficha.Color.N)) {
               cantNegras++;
            }
         }
      }
      if (cantNegras > cantBlancas) {
         jugadorGano = jugadorDos;
         jugadorDos.setCantPartidasGanadas(jugadorDos.getCantPartidasGanadas() + 1);
      }
      if (cantBlancas > cantNegras) {
         jugadorGano = jugadorUno;
         jugadorUno.setCantPartidasGanadas(jugadorUno.getCantPartidasGanadas() + 1);
      }
      jugadorUno.setCantPartidasJugadas(jugadorUno.getCantPartidasJugadas() + 1);
      jugadorDos.setCantPartidasJugadas(jugadorDos.getCantPartidasJugadas() + 1);
      return jugadorGano;
   }

   public void reiniciarMatAux() {
      for (int i = 0; i < matAux.length; i++) {
         for (int j = 0; j < matAux.length; j++) {
            matAux[i][j].setColor(Ficha.Color.noHay);
         }
      }
   }

   public Ficha obtenerFicha(int posFila, int posCol) {
      Ficha ficha = new Ficha(Ficha.Color.noHay);
      if (posFila >= 0 && posCol >= 0 && posFila < tablero.largoTablero()
            && posCol < tablero.largoTablero()) {
         if (posFilaPanel > posFila || posColPanel > posCol
               || posCol > (posColPanel + panel.dimPanel() - 1)
               || posFila > (posFilaPanel + panel.dimPanel() - 1)) {
            //está en el tablero base
            ficha = tablero.obtenerFichaTablero(posFila, posCol);
         } else {
            //está en el panel
            ficha = panel.obtenerFichaPanel(posFila - posFilaPanel, posCol - posColPanel);
         }
      }
      return ficha;
   }

   public String entenderJugada(String jugada) {
      String retorno = "ok";
      if (validarJugada(jugada)) {
         int posFila = 0;
         char[] filaLetras = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
         jugada = jugada.toUpperCase();
         char primerComando = jugada.charAt(0);
         if (jugada.equals("X")) {
```

```
            jugadorPerdedor = jugadorDeTurno;
            if (jugadorDeTurno.equals(jugadorUno)) {
               jugadorDos.setCantPartidasGanadas(jugadorDos.getCantPartidasGanadas() + 1);
               jugadorUno.setCantPartidasJugadas(jugadorUno.getCantPartidasJugadas() + 1);
               jugadorDos.setCantPartidasJugadas(jugadorDos.getCantPartidasJugadas() + 1);
            } else {
               jugadorUno.setCantPartidasGanadas(jugadorUno.getCantPartidasGanadas() + 1);
               jugadorUno.setCantPartidasJugadas(jugadorUno.getCantPartidasJugadas() + 1);
               jugadorDos.setCantPartidasJugadas(jugadorDos.getCantPartidasJugadas() + 1);
            }
            jugadorDeTurno = null;
         } else {
            char segundoComando = jugada.charAt(1);
            int tercerComando = ((int) (jugada.charAt(2))) - 49;
            if (primerComando == 'J') {
               for (int i = 0; i < filaLetras.length; i++) {
                  if (filaLetras[i] == segundoComando) {
                     posFila = i;
                  }
               }
               retorno = validarFicha(posFila, tercerComando);
               if (validarFicha(posFila, tercerComando).equals("ok")) {
                  if ((jugadorDeTurno.equals(jugadorUno) && cantidadFichasUno != 0)
                        || jugadorDeTurno.equals(jugadorDos) && cantidadFichasDos != 0) {
                     if (!fichaDentroPanel(posFila, tercerComando)) {
                        //Si no está dentro del panel
                        if (jugadorDeTurno.equals(jugadorUno)) {
                           tablero.ponerFichaTablero(posFila, tercerComando, new
Ficha(Ficha.Color.B));
                           cantidadFichasUno--;
                        } else {
                           tablero.ponerFichaTablero(posFila, tercerComando, new
Ficha(Ficha.Color.N));
                           cantidadFichasDos--;
                        }
                     } else {
                        //Si está dentro del panel
                        if (jugadorDeTurno.equals(jugadorUno)) {
                           panel.ponerFichaPanel(posFila - posFilaPanel, tercerComando -
posColPanel, new Ficha(Ficha.Color.B));
                           cantidadFichasUno--;
                        } else {
                           panel.ponerFichaPanel(posFila - posFilaPanel, tercerComando -
posColPanel, new Ficha(Ficha.Color.N));
                           cantidadFichasDos--;
                        }
                     }
                  } else {
                     retorno = "No hay más fichas";
                  }

               }
```

```
            if (retorno.equals("ok")) {
               hacerSimetrias(posFila, tercerComando, jugadorDeTurno, jugadorUno);
            }
            if (haySimetria) {
               cambiarFichasDeColor();
            }
            if (retorno.equals("ok")) {
               cambiarTurno();
            }
         } else if (primerComando == 'P') {
            for (int i = 0; i < filaLetras.length; i++) {
               if (filaLetras[i] == segundoComando) {
                  posFila = i;
               }
            }
            retorno = ponerPanel(posFila, tercerComando);
            if (retorno.equals("ok")) {
               cambiarTurno();
            }
         } else if (primerComando == 'M') {
            retorno = moverPanel(segundoComando, tercerComando);
            if (retorno.equals("ok")) {
               cambiarTurno();
            }
         } else if (primerComando == 'L') {
            for (int i = 0; i < filaLetras.length; i++) {
               if (filaLetras[i] == segundoComando) {
                  posFila = i;
               }
            }
            retorno = limpiar(posFila, tercerComando);
            if (retorno.equals("ok")) {
               cambiarTurno();
            }
         }
      }
   } else {
      //la jugada no es válida
      retorno = "Ingreso una jugada inválida ";
   }
   return retorno;
}

private String limpiar(int unaFila, int unaCol) {
   String retorno = "ok";
   if (!limpiezaUno && jugadorDeTurno.equals(jugadorUno)) {
      for (int i = 0; i < tablero.largoTablero(); i++) {
         for (int j = 0; j < tablero.largoTablero(); j++) {
            obtenerFicha(i, unaCol).setColor(Ficha.Color.noHay);
            obtenerFicha(unaFila, j).setColor(Ficha.Color.noHay);
         }
      }
```

```
            limpiezaUno = true;
        } else if (!limpiezaDos && jugadorDeTurno.equals(jugadorDos)) {
            for (int i = 0; i < tablero.largoTablero(); i++) {
                for (int j = 0; j < tablero.largoTablero(); j++) {
                    obtenerFicha(i, unaCol).setColor(Ficha.Color.noHay);
                    obtenerFicha(unaFila, j).setColor(Ficha.Color.noHay);
                }
            }
            limpiezaDos = true;
        } else {
            retorno = "Ya se ha utilizado la ficha de limpieza.";
        }
        return retorno;
    }

    public boolean haySimetria() {
        boolean retorno = haySimetria;
        this.haySimetria = false;
        return retorno;
    }

    public String obtenerJugadorActual() {
        String retorno = "negro";
        if (jugadorDeTurno.equals(jugadorUno)) {
            retorno = "blanco";
        }
        return retorno;
    }

    public boolean seguirJugando() {
        boolean retorno = true;
        if ((cantidadFichasUno == 0 && cantidadFichasDos == 0) || jugadorDeTurno == null) {
            retorno = false;
        }
        return retorno;
    }

    public String resultadoJuego() {
        String resultado = "El jugador que gano es ";
        if (cantidadFichasDos == 0 && cantidadFichasUno == 0) {
            if (conteoFichas().equals(jugadorUno)) {
                resultado += jugadorUno.getAlias();
            } else if (conteoFichas().equals(jugadorDos)) {
                resultado += jugadorDos.getAlias();
            } else {
                resultado = "La partida ha terminado en empate";
            }
        } else {
            if (jugadorUno.equals(jugadorPerdedor)) {
                resultado += jugadorDos.getAlias();
            }
            if (jugadorDos.equals(jugadorPerdedor)) {
```

```
            resultado += jugadorUno.getAlias();
        }
    }
    return resultado;
}

public String resultadoJuegoNoTerminado() {
    String resultado = "El jugador que gano es ";
    if (conteoFichas().equals(jugadorUno)) {
        resultado += jugadorUno.getAlias();
    } else if (conteoFichas().equals(jugadorDos)) {
        resultado += jugadorDos.getAlias();
    } else {
        resultado = "La partida ha terminado en empate";
    }
    return resultado;
}

public String imprimirTablero() {
    char[] filaLetras = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
    //imprimo cabecera números
    String retorno = "   ";
    for (int i = 1; i <= tablero.largoTablero(); i++) {
        retorno += " " + i;
    }
    // fin imprimo cabecera números

    // comienzo imprimo +-
    retorno += "\n   ";
    for (int i = 0; i < tablero.largoTablero(); i++) {
        if (fichaDentroPanel(0, i)) {
            retorno += "##";
        } else if (fichaDentroPanel(0, i - 1)) {
            retorno += "#-";
        } else {
            if (i == tablero.largoTablero()) {
                retorno += "#";
            } else {
                retorno += "+-";
            }

        }
    }

    if (fichaDentroPanel(0, tablero.largoTablero() - 1)) {
        retorno += "#\n";
    } else {
        retorno += "+\n";
    }

    // fin imprimo +-
    //imprimo letras en fila y letras (or espacio)(+-)
```

```
    for (int i = 0; i < tablero.largoTablero(); i++) {
        retorno += String.valueOf(filaLetras[i]) + "  ";
        for (int j = 0; j < tablero.largoTablero(); j++) {

            if (fichaDentroPanel(i, j)) {
                retorno += "#" + obtenerFicha(i, j);
            } else if (fichaDentroPanel(i, j - 1)) {
                retorno += "#" + obtenerFicha(i, j);
            } else {
                retorno += "|" + obtenerFicha(i, j);
            }

        }
        if (fichaDentroPanel(i, tablero.largoTablero() - 1)) {
            retorno += "#";
        } else {
            retorno += "|";
        }

        retorno += "\n   ";
        for (int k = 0; k < tablero.largoTablero(); k++) {
            if (fichaDentroPanel(i, k) || fichaDentroPanel(i + 1, k)) {
                retorno += "##";
            } else if (fichaDentroPanel(i, k - 1) || fichaDentroPanel(i + 1, k - 1)) {
                retorno += "#-";
            } else {
                retorno += "+-";
            }
        }

        if (fichaDentroPanel(i, tablero.largoTablero() - 1)
                || fichaDentroPanel(i + 1, tablero.largoTablero() - 1)) {
            retorno += "#\n";
        } else {
            retorno += "+\n";
        }
    }
    return retorno;
}

public String imprimirTableroAux() {
    char[] filaLetras = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
    //imprimo cabecera números
    String retorno = "   ";
    for (int i = 1; i <= tablero.largoTablero(); i++) {
        retorno += " " + i;
    }
    // fin imprimo cabecera números

    // comienzo imprimo +-
    retorno += "\n   ";
    for (int i = 0; i < tablero.largoTablero(); i++) {
```

```
   if (fichaDentroPanel(0, i)) {
      retorno += "##";
   } else if (fichaDentroPanel(0, i - 1)) {
      retorno += "#-";
   } else {
      if (i == tablero.largoTablero()) {
         retorno += "#";
      } else {
         retorno += "+-";
      }

   }
}

if (fichaDentroPanel(0, tablero.largoTablero() - 1)) {
   retorno += "#\n";
} else {
   retorno += "+\n";
}

// fin imprimo +-
//imprimo letras en fila y letras (or espacio)(+-)
for (int i = 0; i < matAux.length; i++) {
   retorno += String.valueOf(filaLetras[i]) + "  ";
   for (int j = 0; j < matAux.length; j++) {

      if (fichaDentroPanel(i, j)) {
         retorno += "#" + matAux[i][j];
      } else if (fichaDentroPanel(i, j - 1)) {
         retorno += "#" + matAux[i][j];
      } else {
         retorno += "|" + matAux[i][j];
      }

   }
   if (fichaDentroPanel(i, matAux.length - 1)) {
      retorno += "#";
   } else {
      retorno += "|";
   }

   retorno += "\n   ";
   for (int k = 0; k < matAux.length; k++) {
      if (fichaDentroPanel(i, k) || fichaDentroPanel(i + 1, k)) {
         retorno += "##";
      } else if (fichaDentroPanel(i, k - 1) || fichaDentroPanel(i + 1, k - 1)) {
         retorno += "#-";
      } else {
         retorno += "+-";
      }
   }
```

```
            if (fichaDentroPanel(i, matAux.length - 1)
                    || fichaDentroPanel(i + 1, matAux.length - 1)) {
                retorno += "#\n";
            } else {
                retorno += "+\n";
            }
        }
        reiniciarMatAux();
        return retorno;
    }

    public String jugadorDeTurnoToString() {
        return jugadorDeTurno.getAlias();
    }

    public java.awt.Color getColorJugadorActual() {
        java.awt.Color color;
        if (jugadorDeTurno.equals(jugadorUno)) {
            color = colorJugadorUno;
        } else {
            color = colorJugadorDos;
        }
        return color;
    }

    public java.awt.Color getColorElOtroJugador() {
        java.awt.Color color;
        if (jugadorDeTurno.equals(jugadorUno)) {
            color = colorJugadorDos;
        } else {
            color = colorJugadorUno;
        }
        return color;
    }

    public java.awt.Color setColorJugadorActual(java.awt.Color unColor) {
        if (jugadorDeTurno.equals(jugadorUno)) {
            colorJugadorUno = unColor;
        } else {
            colorJugadorDos = unColor;
        }
        return unColor;
    }

    public Ficha[][] obtenerMatAux() {
        return matAux;
    }

    public int getPosFilaPanel() {
        return posFilaPanel;
    }
```

```
    public int getPosColPanel() {
        return posColPanel;
    }

    public Panel obtenerPanel() {
        return this.panel;
    }

    public java.awt.Color obtenerColorUno() {
        return colorJugadorUno;
    }

    public java.awt.Color obtenerColorDos() {
        return colorJugadorDos;
    }

    public String cantFichas() {
        String cant = " ";
        if (jugadorDeTurno.equals(jugadorUno)) {
            cant = String.valueOf(cantidadFichasUno);
        } else {
            cant = String.valueOf(cantidadFichasDos);
        }
        return cant;
    }
}
```

Clase Sistema:

```
package dominio;

import java.io.Serializable;
import java.util.*;
import persistencia.ArchivoLectura;

public class Sistema implements Serializable {

    //Atributos
    private ArrayList<Jugador> listaJugadores;

    // Métodos acceso y modificacion
    public Sistema() {
        listaJugadores = new ArrayList<Jugador>();
    }

    public ArrayList<Jugador> getListaJugadores() {
        Collections.sort(listaJugadores);
        return listaJugadores;
    }

    public void setListaJugadores(ArrayList<Jugador> lista) {
        listaJugadores = lista;
```

```
}

public void registrarJugador(Jugador unJugador) {
   this.listaJugadores.add(unJugador);
}

public boolean importarJugadores(String pathArchivo) {
   boolean importoOk = true;
   ArchivoLectura arch = new ArchivoLectura(pathArchivo);
   while (arch.hayMasLineas()) {
      try {
         String linea = arch.linea();
         if (importeValido(linea)) {
            String[] tokens = linea.split("#");
            String nombre = tokens[0];
            int edad = Integer.parseInt(tokens[1]);
            String alias = tokens[2];
            Jugador jugador = new Jugador(nombre,edad,alias,0,0);
            if(listaJugadores.contains(jugador)){
              int num = listaJugadores.indexOf(jugador);
              listaJugadores.get(num).setNombre(nombre);
              listaJugadores.get(num).setEdad(edad);
            }
            else {
               listaJugadores.add(jugador);
            }
         }
         else {
            importoOk = false;
         }
      } catch (Exception err) {
         importoOk = false;
      }
   }
   arch.cerrar();
   return importoOk;
}

public boolean importeValido(String linea) {
   char uno = ' ';
   char dos = ' ';
   boolean esValido = true;
   String[] tokens = linea.split("#");
   if (tokens.length != 3 || tokens[0].isEmpty()
         || tokens[1].length() != 2 || tokens[2].isEmpty()) {
      esValido = false;
   } else {
      uno = tokens[1].charAt(0);
      dos = tokens[1].charAt(1);
      if (!Character.isDigit(uno) || !Character.isDigit(dos)) {
         esValido = false;
      }
```

```
        }
        return esValido;
    }
}
```

Clase Tablero:

```
package dominio;

import dominio.Ficha.Color;

public class Tablero {

    //Atributos
    private Ficha[][] tableroBase;

    public int largoTablero() {
        return tableroBase.length;
    }

    //Constructor con parámetros
    public Tablero(int dimTableroBase) {
        tableroBase = new Ficha[dimTableroBase][dimTableroBase];
        for (int i = 0; i < tableroBase.length; i++) {
            for (int j = 0; j < tableroBase.length; j++) {
                this.tableroBase[i][j] = new Ficha(Color.noHay);
            }
        }
    }

    public void ponerFichaTablero(int unaFila, int unaCol, Ficha ficha) {
        tableroBase[unaFila][unaCol] = ficha;
    }

    public Ficha obtenerFichaTablero(int posFila, int posCol){
        return tableroBase[posFila][posCol];
    }
}
```

Inversor Angel

Un Inversor Angel es un individuo o una organización que brinda apoyo financiero y, en casos, conocimiento empresarial para patrocinar un startup. Se les llama startup a los emprendimientos nuevos e innovadores que son lanzados al mercado orientados a las nuevas tecnologías.

Esta es una inversión de alto riesgo ya que se financia un proyecto no estable, es decir, que se destina capital a un proyecto que no ha salido al mercado y puede no ser exitoso. Generalmente, el ángel inversor busca "devolver al menos 10 veces o más de sus originales la inversión en 5 años".

El perfil del ángel inversor es, en general, una persona apasionada por los proyectos que no sólo busca la remuneración monetaria sino que es una persona emprendedora. " Para Susana García-Robles (principal Investment Officer del Banco Interamericano de Desarrollo), «el ángel agrega capital inteligente, no es sólo dinero sino apoyo para desarrollar la estrategia y visión del negocio, abre puertas, conecta con otras iniciativas. Tiene un rol muy importante para una empresa en estado embrionario»."

Fuentes:

http://es.wikipedia.org/wiki/Inversor_angelical

http://www.elpais.com.uy/el-empresario/ecosistema-local-inversores-endeavor-ieem.html