**Daniel Alejandro Diaz Camelo**

**Miguel Ángel Motta**

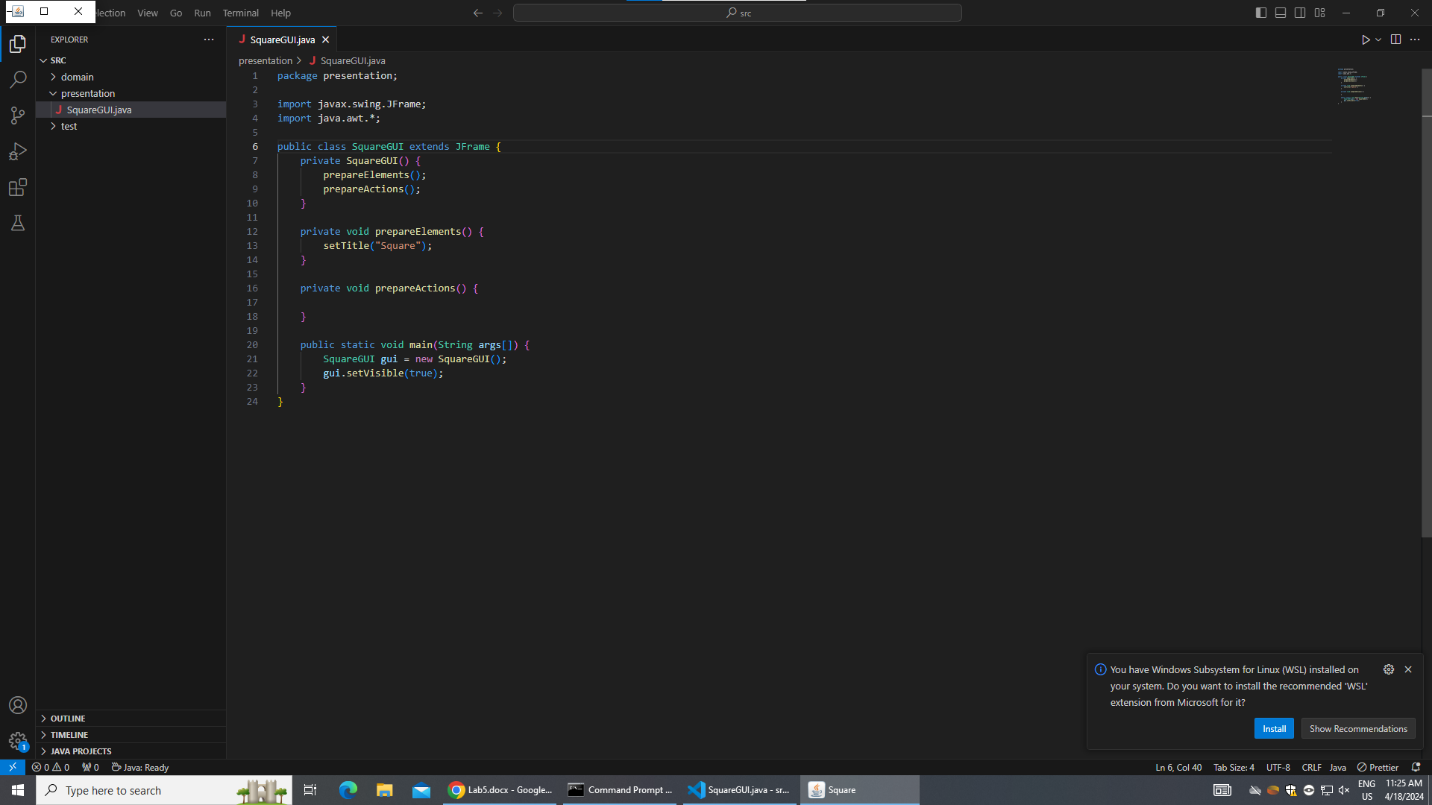
**Laboratorio 5**

**Ciclo 0: Ventana vacía – Salir [En \*.java y lab05.doc]**

El objetivo es implementar la ventana principal de Square con un final adecuado desde el icono de cerrar. Utilizar el esquema de prepareElements-prepareActions.

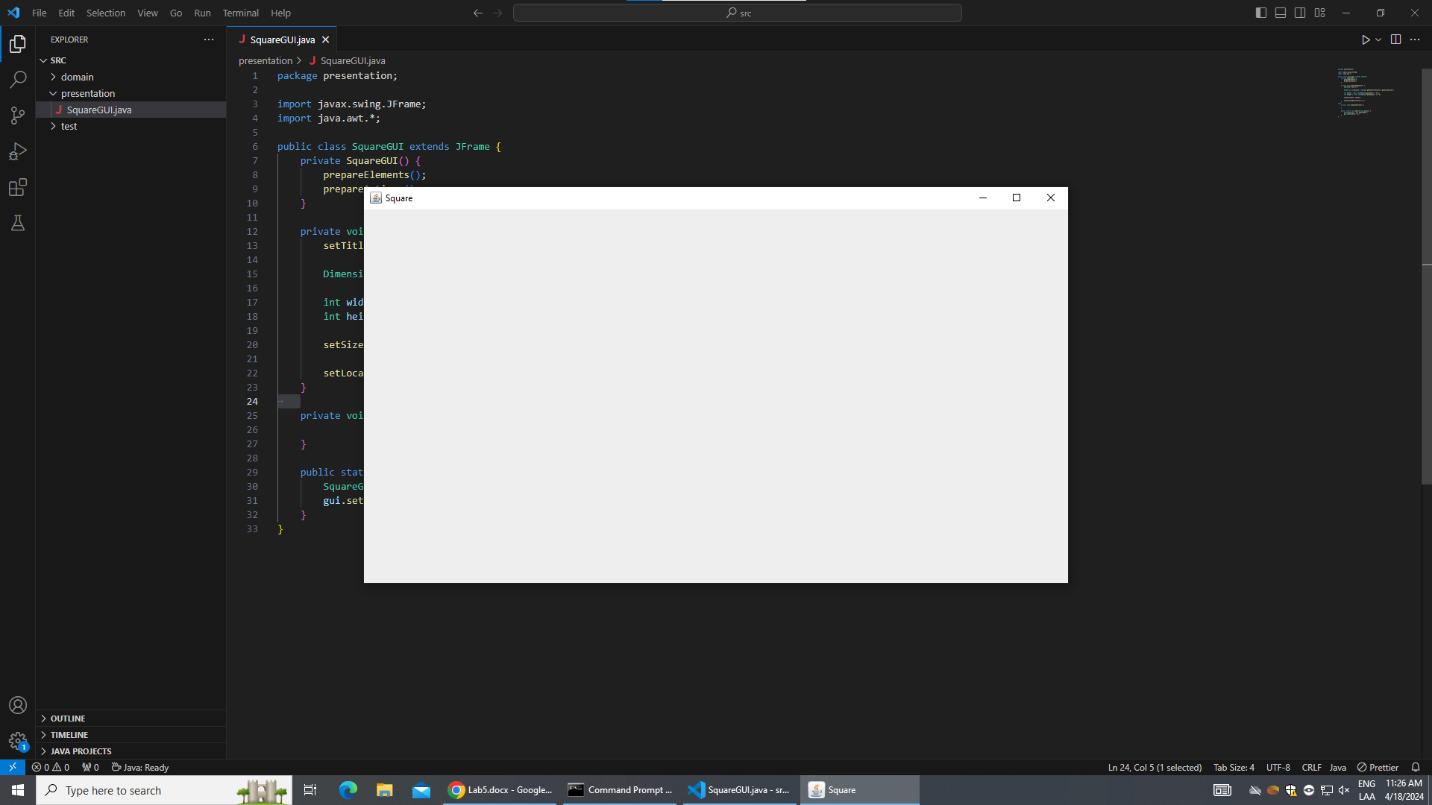
**1.** Construyan el primer esquema de la ventana de Square únicamente con el título “Square”. Para esto cree la clase SquareGUI como un JFrame con su creador (que sólo coloca el título) y el método main que crea un objeto SquareGUI y lo hace visible. Ejecútenlo. Capturen la pantalla. (Si la ventana principal no es la inicial en su diseño, después deberán mover el main al componente visual correspondiente) .

Rta:



**2.** Modifiquen el tamaño de la ventana para que ocupe un cuarto de la pantalla y ubíquenla en el centro. Para eso inicien la codificación del método prepareElements. Capturen esa pantalla.

Rta:



**3.** Traten de cerrar la ventana. ¿Termina la ejecución? ¿Qué deben hacer en consola para terminar la ejecución?

Rta:

Si damos click en el botón de x, en la parte superior derecha, la ventana si se cierra, sin embargo, el programa se sigue ejecutando, pues no se ha terminado de correr el método main(). El JVM puede lanzar múltiples procesos de concurrencia ejecutándolo en diferentes hilos.

**4.** Estudien en JFrame el método setDefaultCloseOperation. ¿Para qué sirve? ¿Cómo lo usarían si queremos confirmar el cierre de la aplicación? ¿Cómo lo usarían si queremos simplemente cerrar la aplicación?

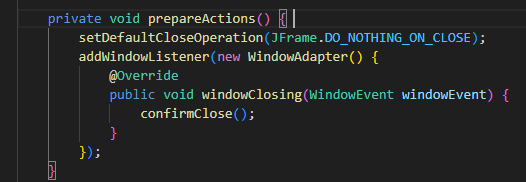
Rta:

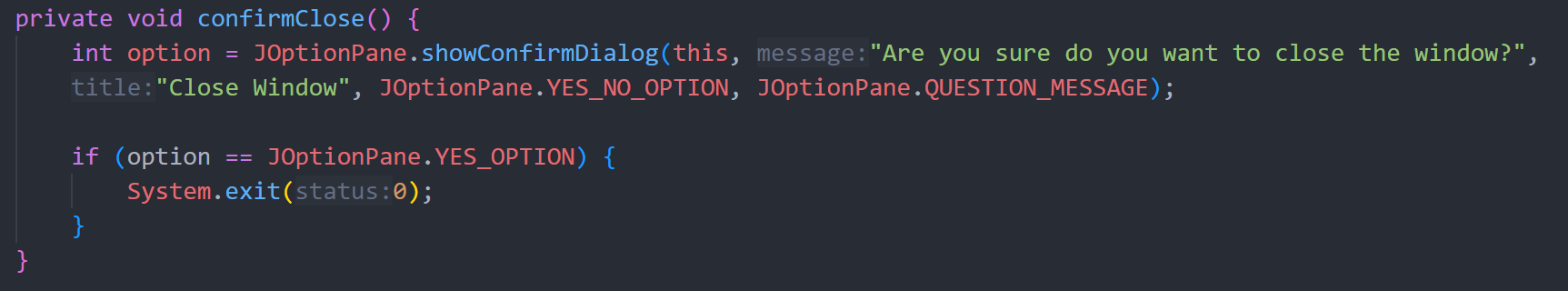
El setDefaultCloseOperation se utiliza para establecer el comportamiento de cierre de una ventana JFrame en una aplicación java swing. Como parametros tenemos:

* JFrame.DO\_NOTHING\_ON\_CLOSE: La ventana no se cierra cuando el usuario lo intenta.
* JFrame.HIDE\_ON\_CLOSE: La ventana se oculta, pero la aplicación se sigue ejecutando.
* JFrame.DISPOSE\_ON\_CLOSE: La ventana se cierra liberando todos los recursos relacionados con ella, pero la aplicación se sigue ejecutando
* JFrame\_CLOSE\_ON\_CLOSE: La aplicación se cierra por completo cuando el usuario cierra la ventana principal.

**5.** Preparen el “oyente” correspondiente al icono cerrar que le pida al usuario que confirme su selección. Para eso inicien la codificación del método prepareActions y el método asociado a la acción (exit). Ejecuten el programa y cierren el programa. Capturen las pantallas.

Rta:





**Ciclo 1: Ventana con menú – Salir [En \*.java y lab05.doc]**

El objetivo es implementar un menú clásico para la aplicación con un final adecuado desde la opción del menú para salir. El menú debe ofrecer mínimo las siguientes opciones: Nuevo, Abrir – Salvar y Salir. Incluyan los separadores de opciones.

1. Expliquen los componentes visuales necesarios para este menú. ¿Cuáles serían atributos y cuáles podrían ser varibles del método prepareElements? Justifique.

Rta:

Debería tener un objeto de tipo JMenuBar (que es donde se almacenan las opciones de la barra superior de la interfaz). Además, debería tener un elemento de tipo JMenu, que despliegue las opciones que se especifican. Por último, debería tener los ítems que se van a ejecutar dependiendo de la opción (Nuevo, Abrir – Salvar y Salir).

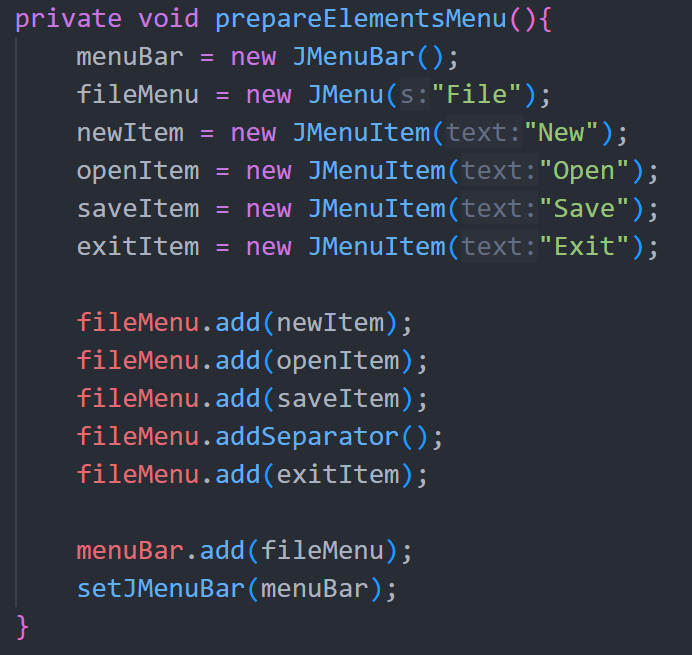
Los atributos son:

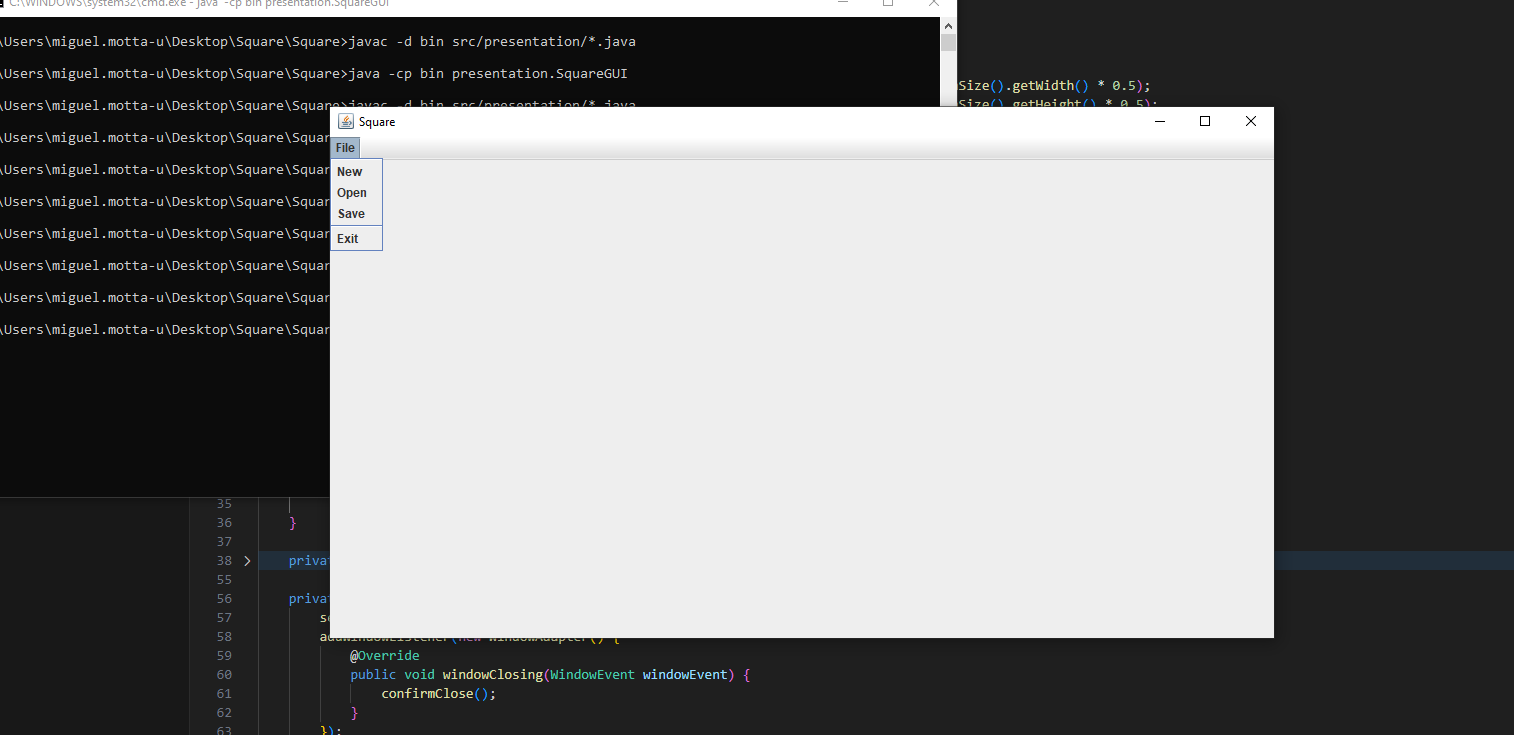
* private JMenuBar menuBar;
* private JMenu fileMenu;
* private JMenuItem newItem;
* private JMenuItem OpenItem;
* private JMenuItem saveItem;
* private JMenuItem exitItem;

Y en el método no tendrá variables de momento puesto que el método prepareElementsMenu() construye el menú por sí solo.

**2.** Construya la forma del menú propuesto (prepareElements - prepareElementsMenu). Ejecuten. Capturen la pantalla.

Rta:





**3.** Preparen el “oyente” correspondiente al icono cerrar con confirmación (prepareActions - prepareActionsMenu). Ejecuten el programa y salgan del programa. Capturen las pantallas.

Rta:

Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

**Ciclo 2: Salvar y abrir [En \*.java y lab05.doc]**

El objetivo es preparar la interfaz para las funciones de persistencia

**1.** Detalle el componente JFileChooser especialmente los métodos: JFileChooser, showOpenDialog, showSaveDialog, getSelectedFile.

* JFileChooser: Permite a los usuarios seleccionar archivos o directorios desde su sistema de archivos locales. Muestra un cuadro de dialogo de selección de archivos.
* showOpenDialog: Muestra un cuadro de dialogo para que el usuario seleccione un archivo para abrir. Devuelve un entero que indica si el usuario selecciono un archivo y, en caso afirmativo, si eligió abrir o cancelar la operación,
* showSaveDialog: Es similar al metodo previo, con la diferencia de que el usuario selecciona el archivo donde guardar algo.
* getSelectedFile: Después de que el usuario seleccione un archivo con cualquiera de los dos métodos anteriores, este devuelve un objeto tipo “File” que representa el archivo seleccionado. Si el usuario cancela la operación o no selecciona ningún archivo devuelve “null”.

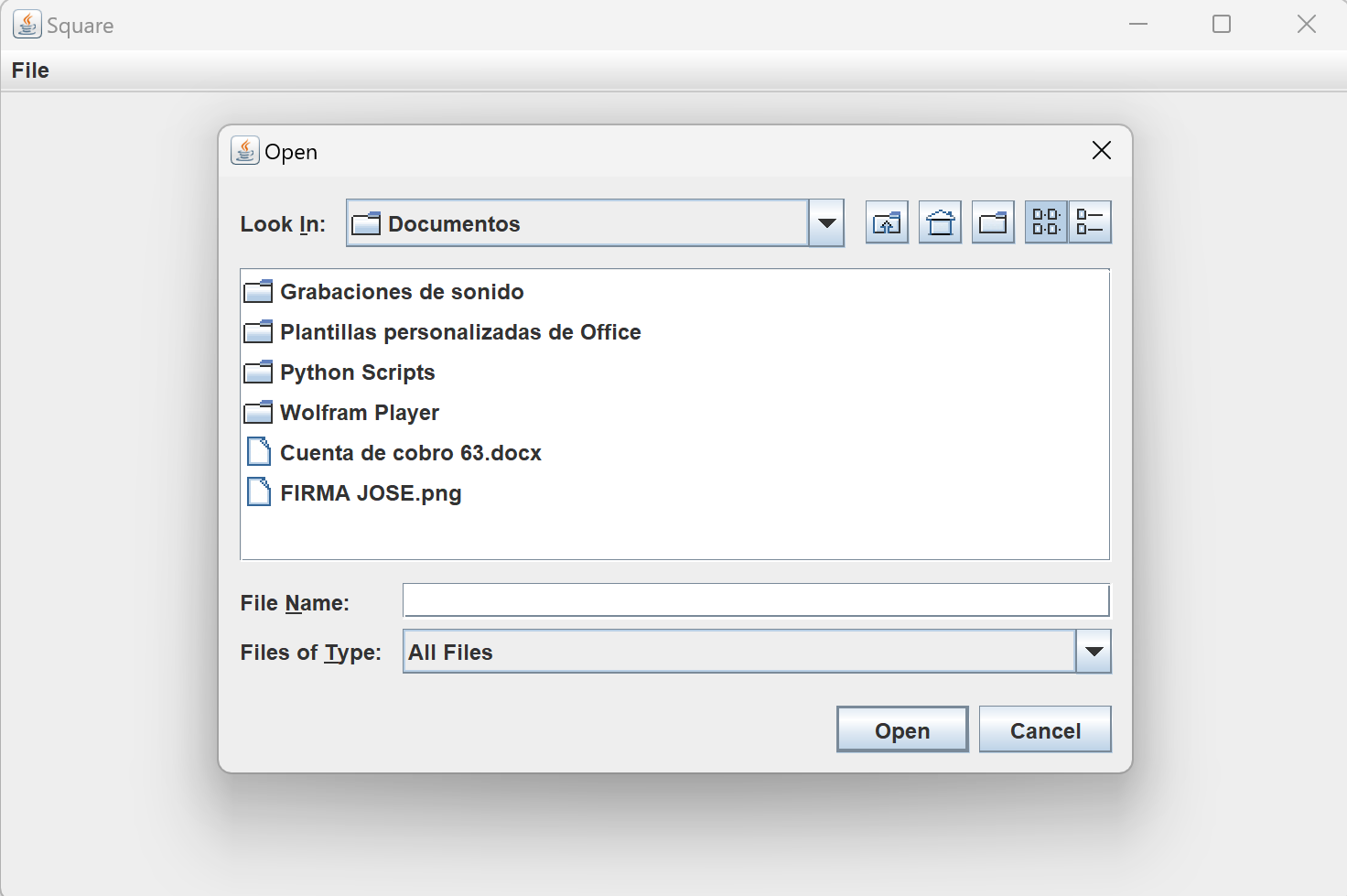
**2.** Implementen parcialmente los elementos necesarios para salvar y abrir. Al seleccionar los archivos indique que las funcionalidades están en construcción detallando la acción y el nombre del archivo seleccionado.

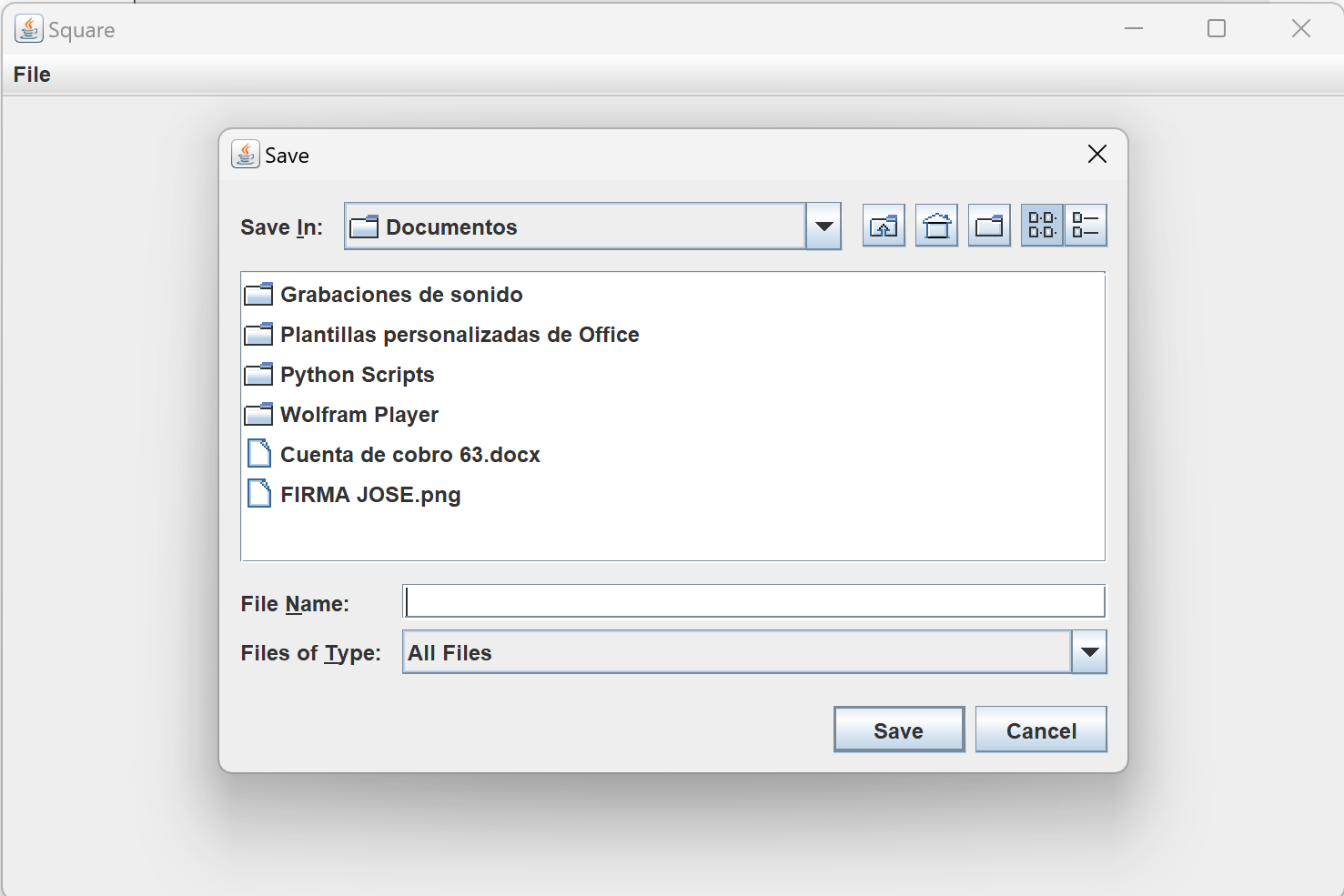
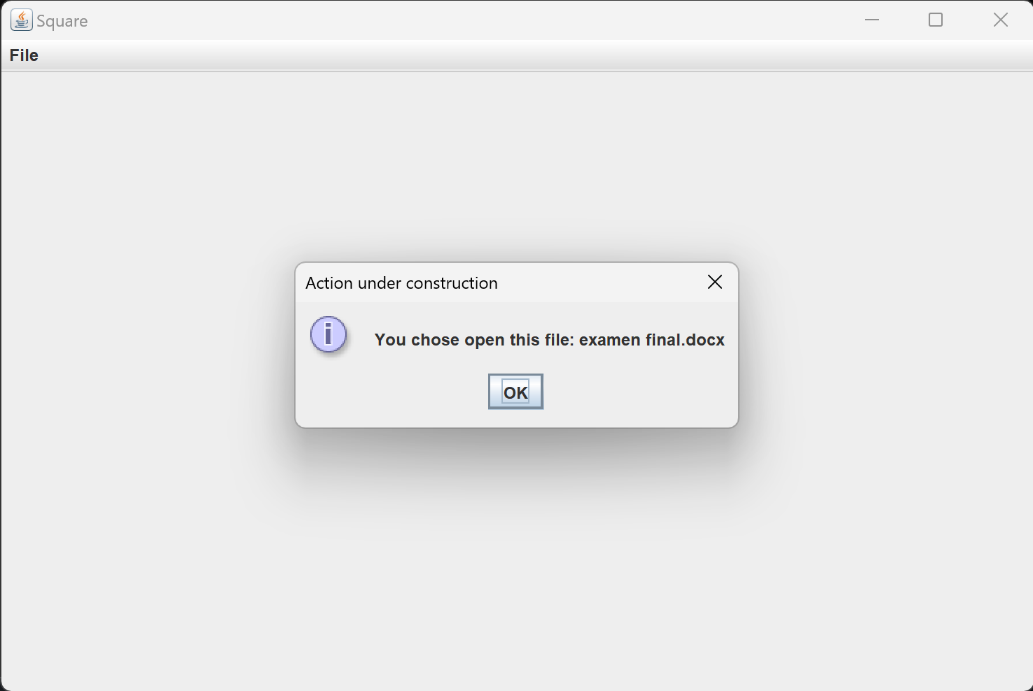
Rta:

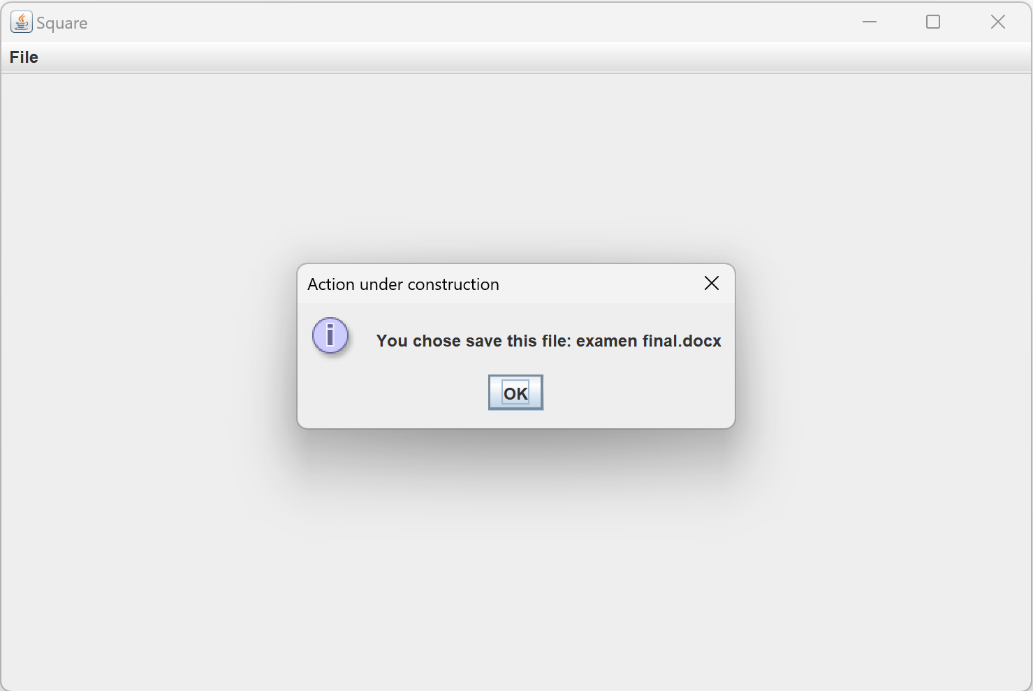


**3.** Ejecuten las dos opciones y capturen las pantallas más significativas.

Rta:





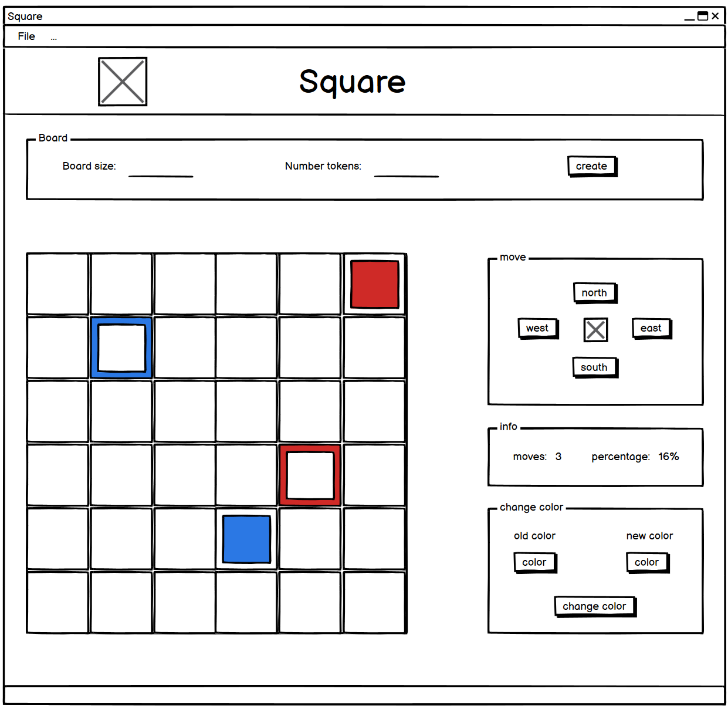


**Ciclo 3: Forma de la ventana principal**

El objetivo es codificar el diseño de la ventana principal (todos los elementos de primer nivel)

**1.** Presenten el bosquejo del diseño de interfaz con todos los componentes necesarios.

Rta:



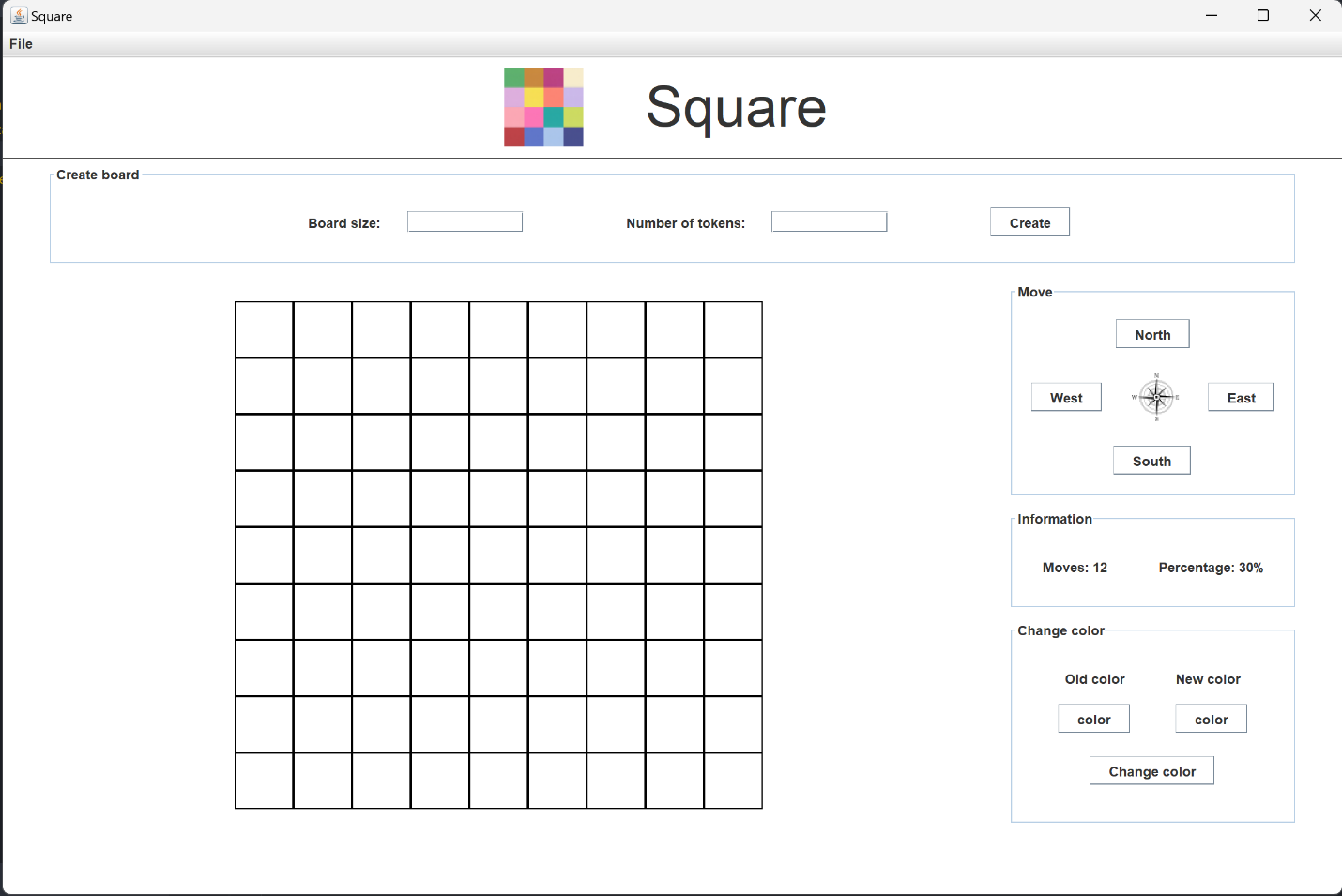
**2.** Continúe con la implementación definiendo los atributos necesarios y extendiendo el método prepareElements(). Para la zona del tablero defina un método prepareElementsBoard() y un método refresh() que actualiza la vista del tablero considerando, por ahora, el tablero inicial por omisión. Este método lo vamos a implementar realmente en otros ciclos.

Rta:

[En \*.java]

**3.** Ejecuten y capturen esta pantalla.

Rta:



**Ciclo 4: Cambiar color**

El objetivo es implementar este caso de uso.

**1.** Expliquen los elementos (vista – controlador) necesarios para implementar este caso de uso.

Rta:

Vista:

Se utilizarán dos etiquetas de texto para indicar al usuario qué color representa cada botón. Esto proporciona una orientación visual clara al usuario sobre qué acción realizar con cada botón. Tres botones serán utilizados:

El primer botón permitirá al usuario seleccionar el color de la ficha que se desea cambiar. Esto se logrará utilizando un JColorChooser para que el usuario pueda seleccionar un color de manera interactiva.

El segundo botón permitirá al usuario seleccionar el nuevo color para la ficha. Al igual que el primer botón, se utilizará un JColorChooser para que el usuario pueda elegir el color deseado.

El tercer botón será el encargado de aplicar los cambios de color efectivamente. Cuando el usuario haga clic en este botón, se llamará al controlador para que inicie el proceso de cambio de color.

Controlador:

El método prepareActionsChangeColor() actuará como el controlador en este caso. Cuando el usuario haga clic en los dos primeros botones para seleccionar los colores, los colores elegidos se almacenarán en atributos.

Cuando el usuario haga clic en el tercer botón para aplicar los cambios, se llamará a un método en el dominio para realizar el cambio de color.

Una vez que se haya completado el cambio de color en el dominio, se actualizará la vista para reflejar los cambios realizados.

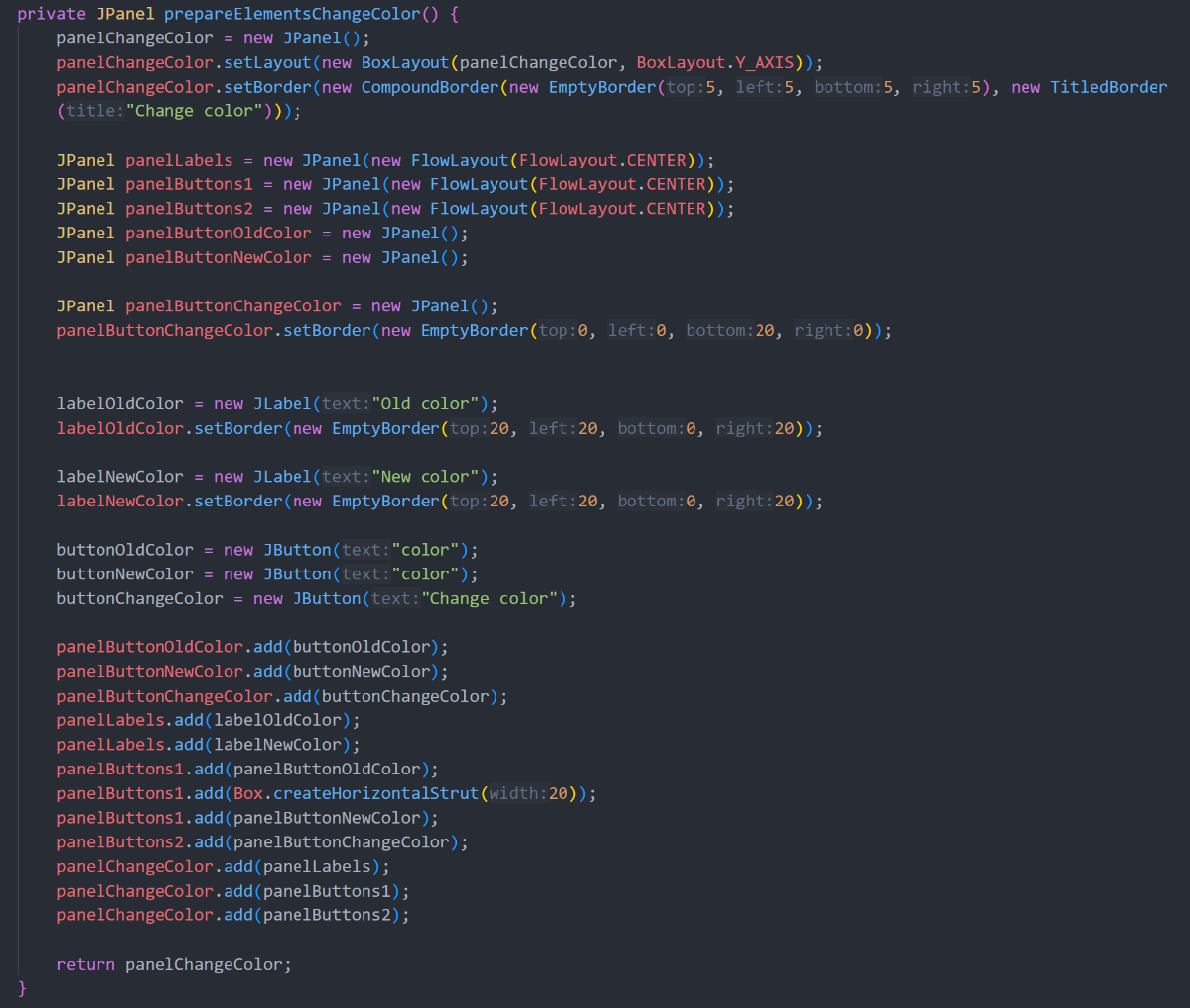
**2.** Detalle el comportamiento de JColorChooser especialmente el método estático showDialog.

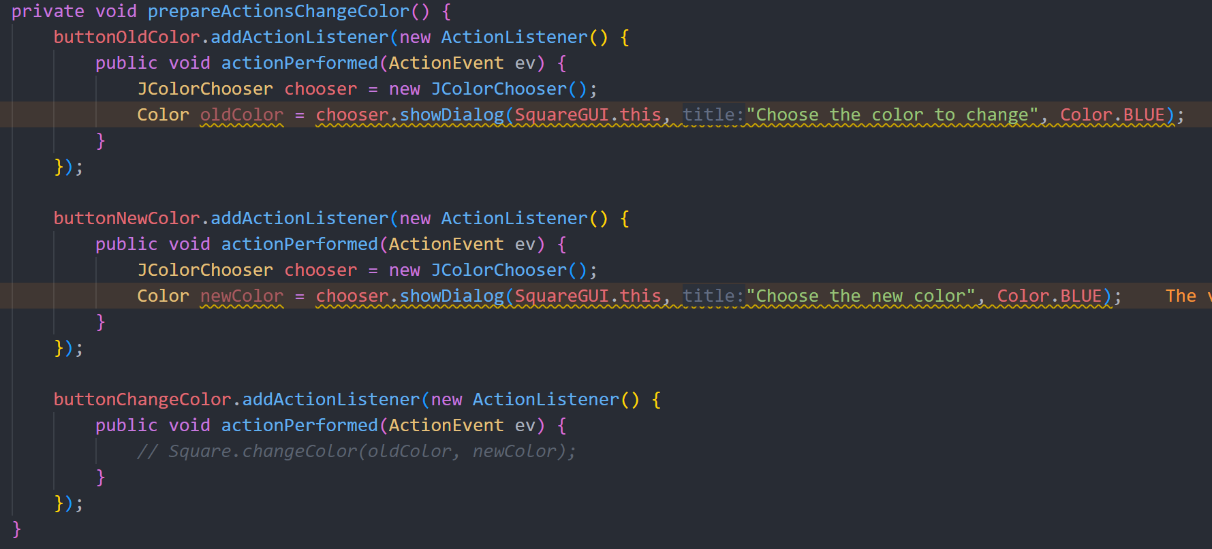
Rta:

JColorChooser es un componente de Swing que permite al usuario seleccionar un color de manera interactiva. El método showDialog muestra un cuadro de dialogo con un selector de color, permitiendo al usuario seleccionar un color y devolviendo el color seleccionado o null si el usuario cancela la selección.

**3.** Implementen los componentes necesarios para cambiar el color de las fichas.

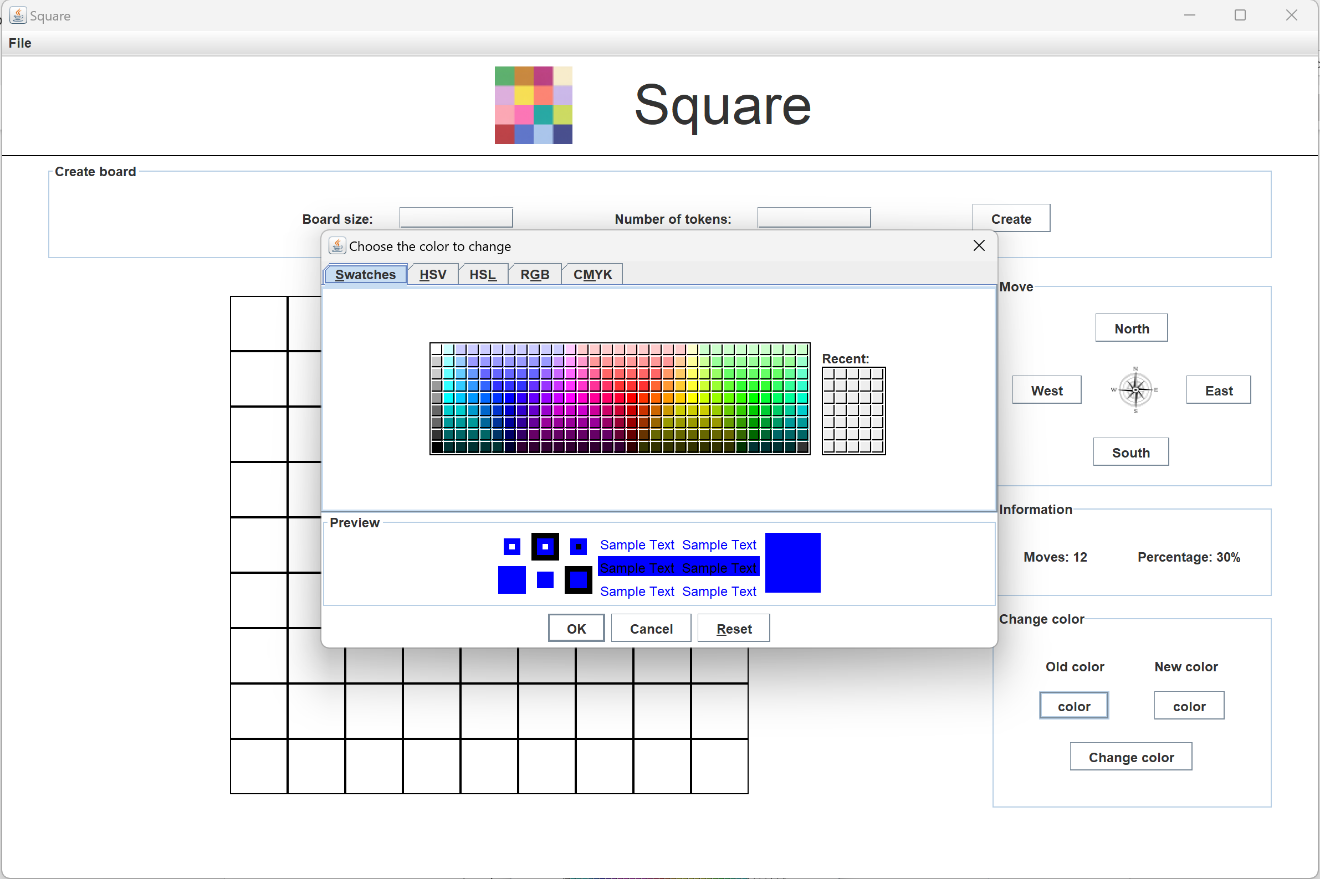
Rta:

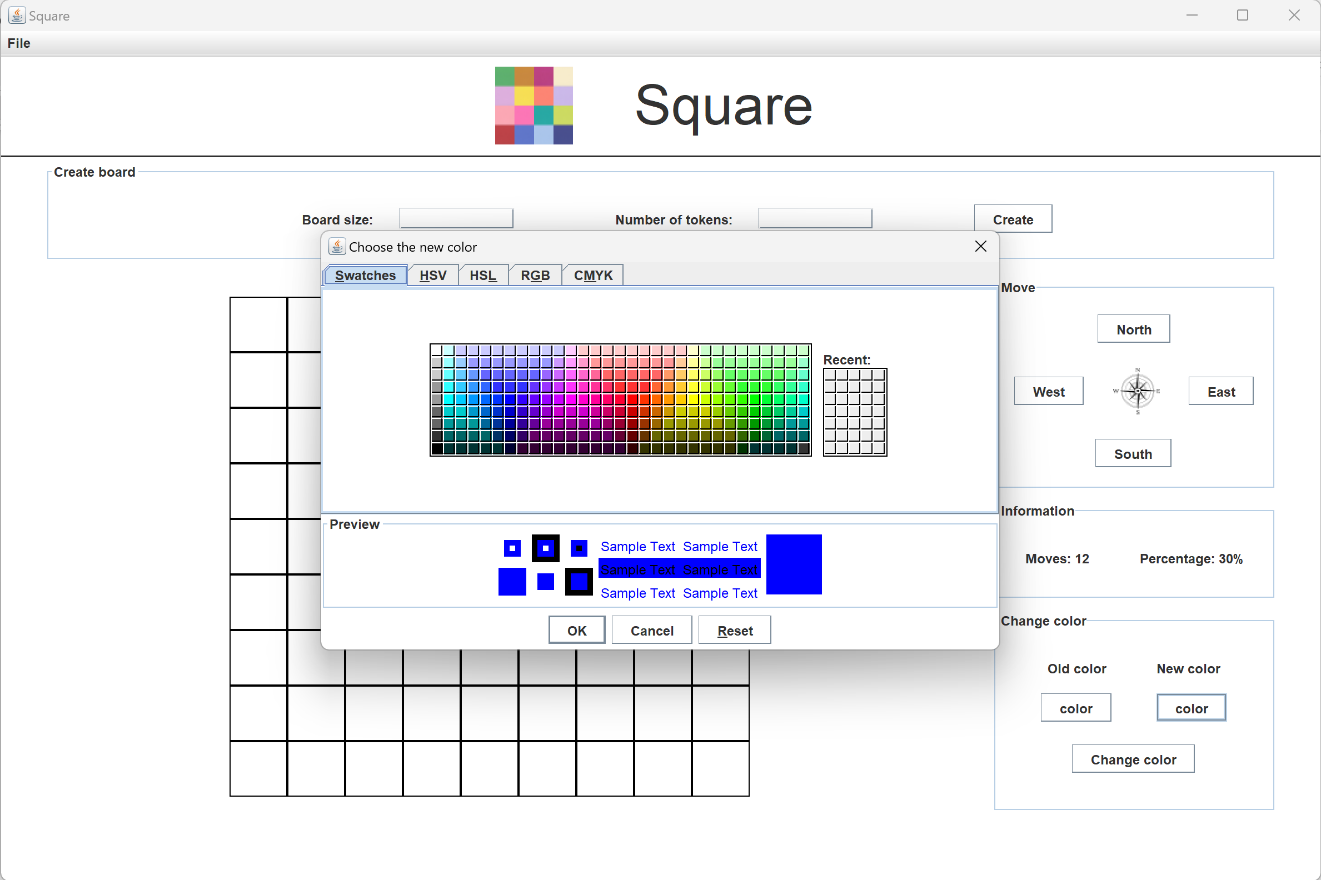




**4.** Ejecuten el caso de uso y capture las pantallas más significativas.

Rta:





**Ciclo 5: Modelo Square**

El objetivo es implementar la capa de dominio para Square.

**1.** Construya los métodos básicos del juego (No olvide MDD y TDD)

Rta:

Texto

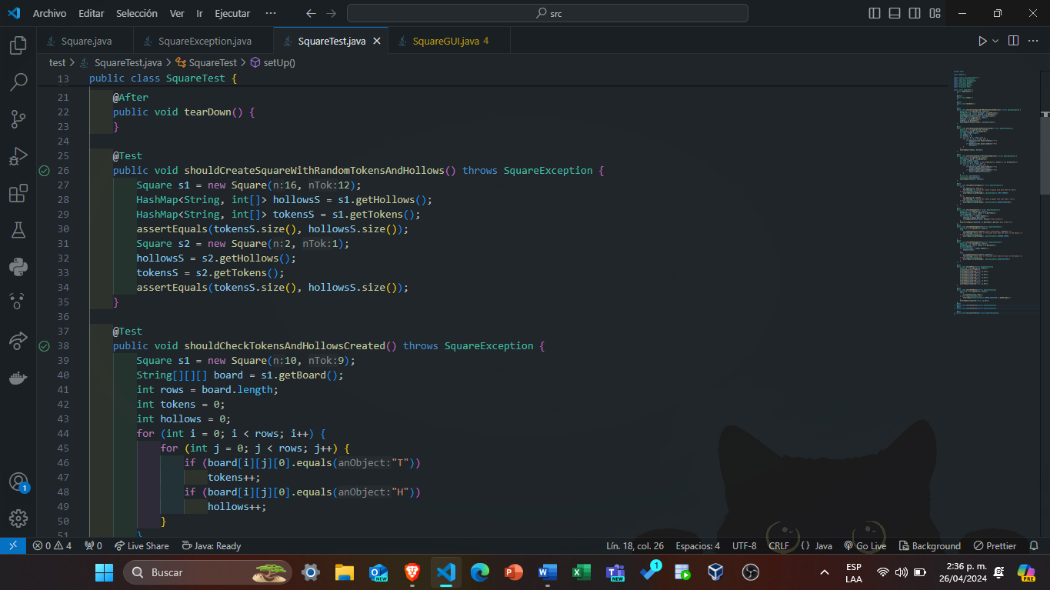
Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**2.** Ejecuten las pruebas y capturen el resultado.

Rta:



Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Pantalla de computadora

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**Ciclo 6: Jugar**

El objetivo es implementar el caso de uso jugar.

**1.** Adicione a la capa de presentación el atributo correspondiente al modelo.

Rta:



**2.** Perfeccionen el método refresh() considerando la información del modelo de dominio.

Rta:



**3.** Expliquen los elementos necesarios para implementar este caso de uso.

Rta:

Vista:

Para la vista se contará con:

Un apartado para crear el juego, donde el usuario podrá ingresar el tamaño de este junto con el número de fichas. Por último, abra un botón, el cual llamara al controlador para que inicie el proceso de crear el tablero

Un apartado para realizar los movimientos, el cual contar con 4 botones que llamaran al controlador para que inicie el proceso de mover la ficha según la dirección deseada

Un apartado para cambiar el color de las fichas, que ya se explicó antes.

Controlador:

Los métodos prepareActionsBoard(), prepareActionsMove() y prepareActionsChangeColor() actuaran como el controlador en este caso.

Cuando el usuario de clic en crear tablero, se creará un objeto Square que representa al tablero.

Cuando el usuario haga clic cualquier botón para mover la ficha, se llamará a un método en el dominio para mover las fichas según la orientación dada.

Para el caso de cambiar el color ya se explicó antes.

Una vez que se haya completado cualquier de estas acciones, se actualizará la vista para reflejar los cambios realizados.

**4.** Implementen los componentes necesarios para jugar. ¿Cuántos oyentes necesitan? ¿Por qué?

Rta:

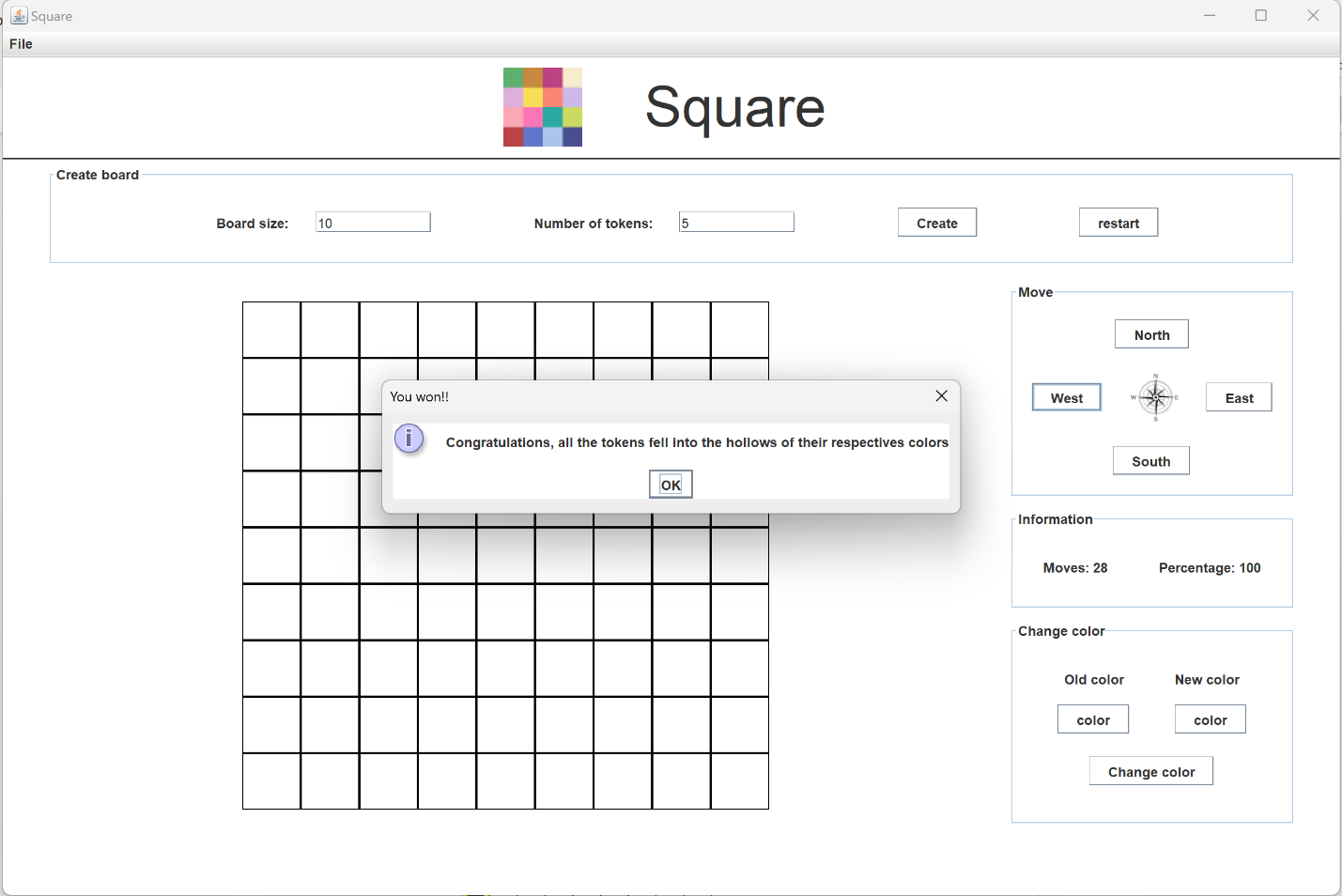
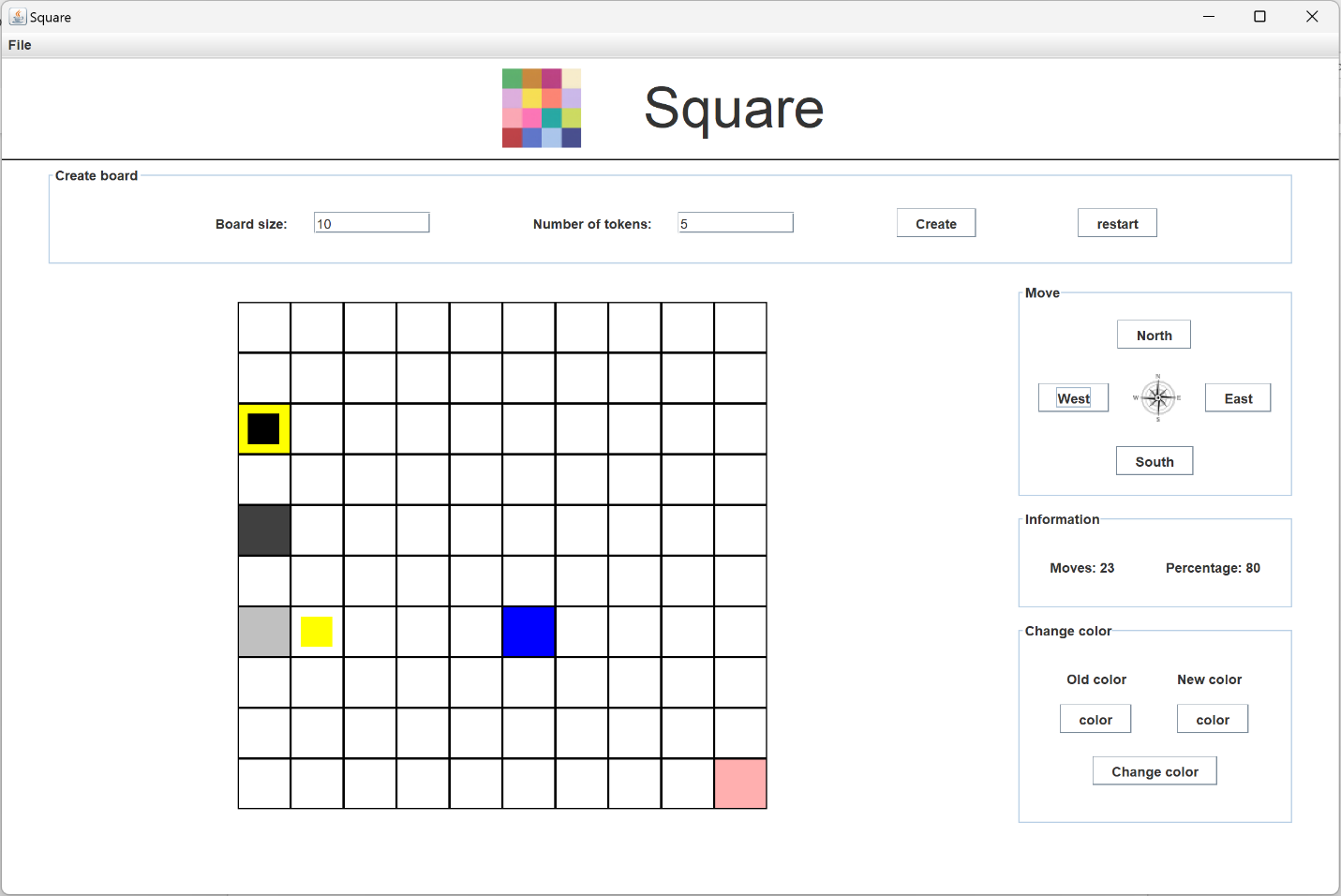
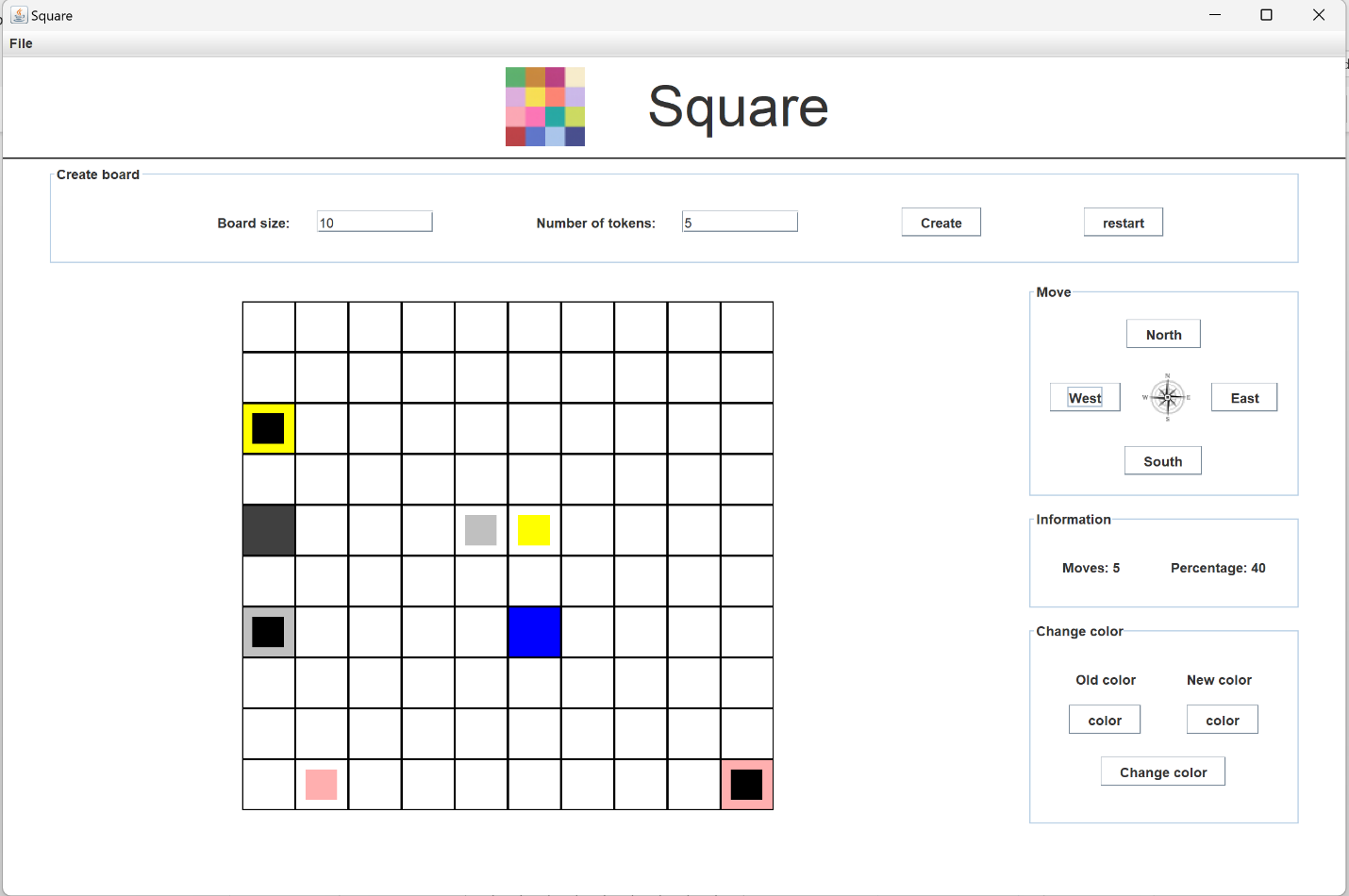
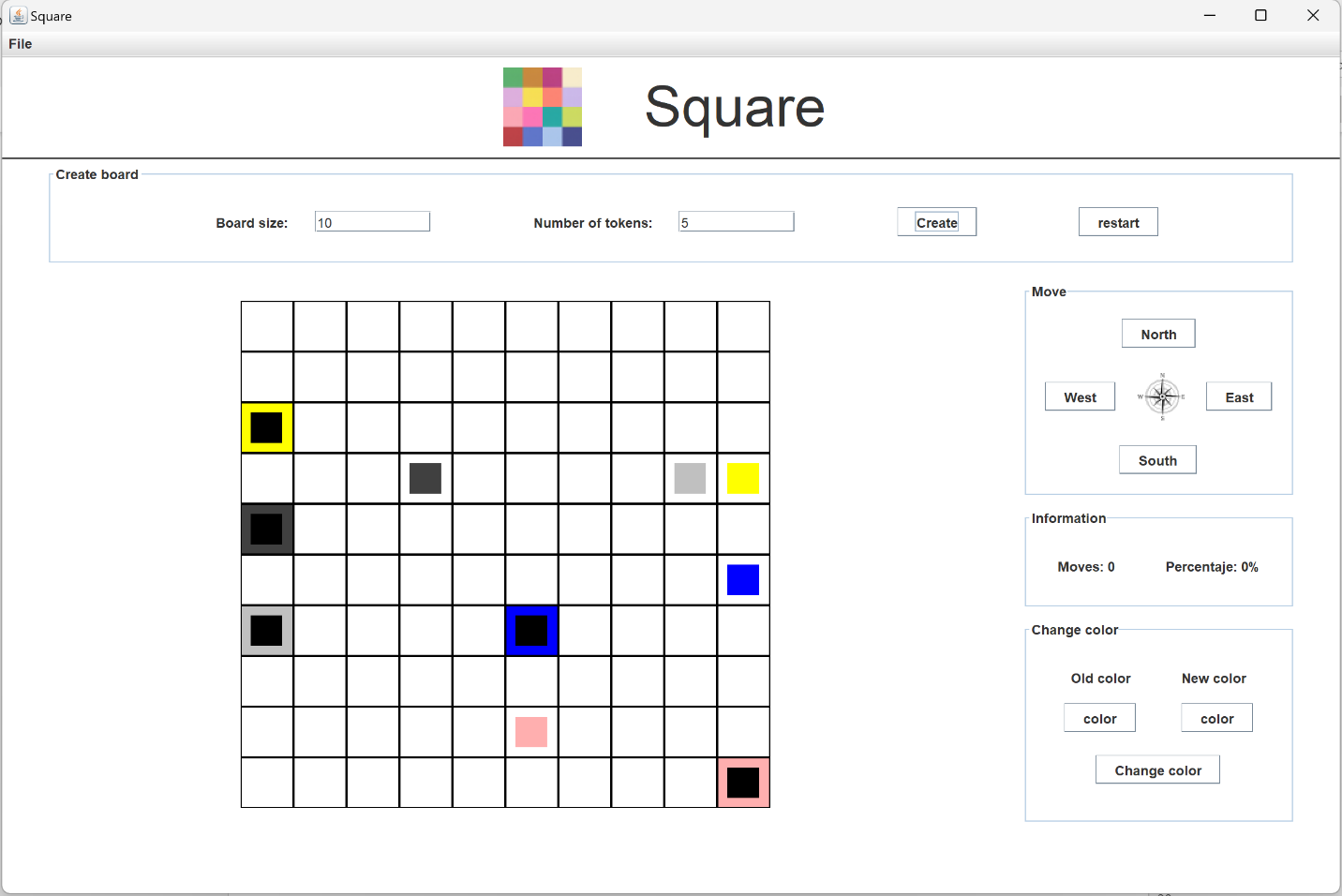
Se necesitan 12 oyentes:

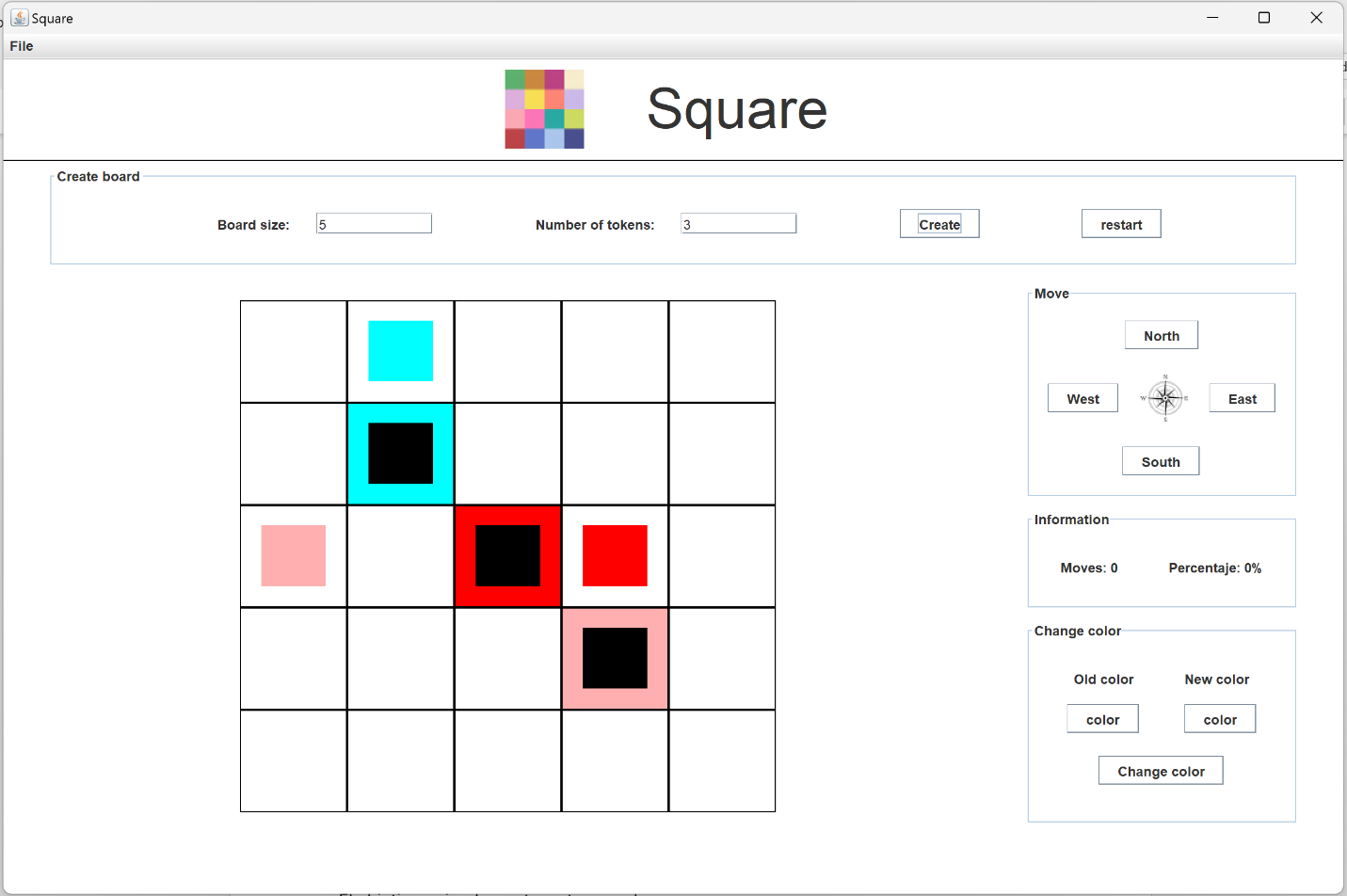
* 3 oyentes para el menú: Uno para guardar la partida, otro para cargar una partida y otro para salir del juego. Estos oyentes gestionarán las acciones del usuario en el menú principal del juego.
* 1 oyente para crear el tablero: Este oyente manejará la acción del usuario al hacer clic en el botón para crear el tablero. Se encargará de iniciar el proceso de creación del tablero en el dominio del juego.
* 4 oyentes para mover las fichas: Uno para cada dirección posible (norte, sur, este y oeste). Estos oyentes serán responsables de mover las fichas en el tablero según la dirección seleccionada por el usuario.
* 3 oyentes para cambiar el color de las fichas: Uno para seleccionar el color de la ficha a cambiar, otro para seleccionar el nuevo color y otro para aplicar efectivamente el cambio de color en el juego.

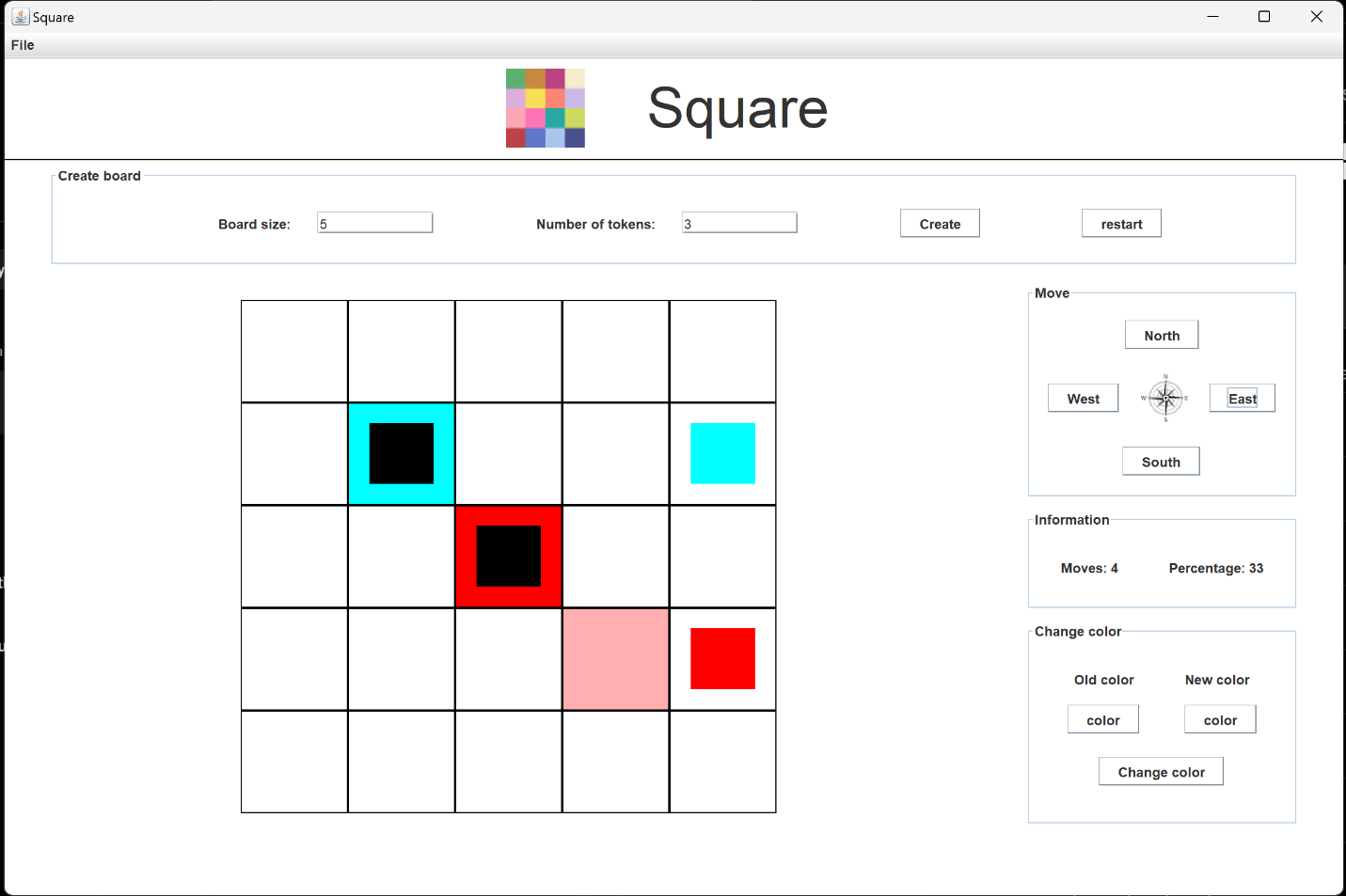
**5.** Ejecuten el caso de uso y capture las pantallas más significativas.

Rta:

Ganar:

Perder:





**Ciclo 7: Reiniciar**

El objetivo es implementar este caso de uso.

**1.** Expliquen los elementos a usar para implementar este caso de uso.

Rta:

Vista:

Para reiniciar el juego se agregó un botón adicional en el apartado de crear el tablero, el cual llamara al controlador para que inicie el proceso de reiniciar el juego.

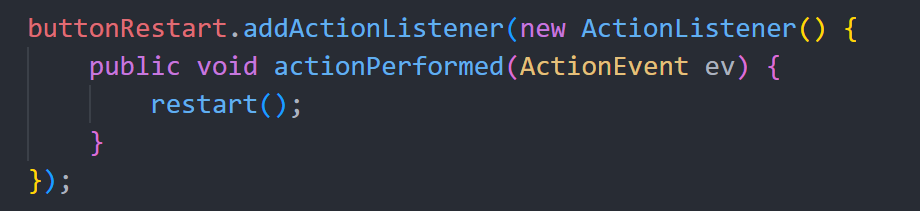
Controlador:

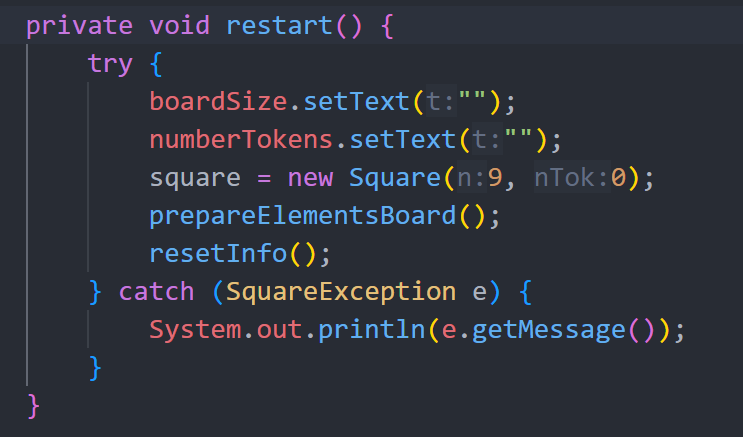
Los métodos prepareActionsBoard() y restart() actuaran como el controlador en este caso.

Cuando el usuario de clic en reiniciar se llamará al método restart(), el cual actualiza la vista a su estado por defecto y se crea un nuevo objeto Square por defecto.

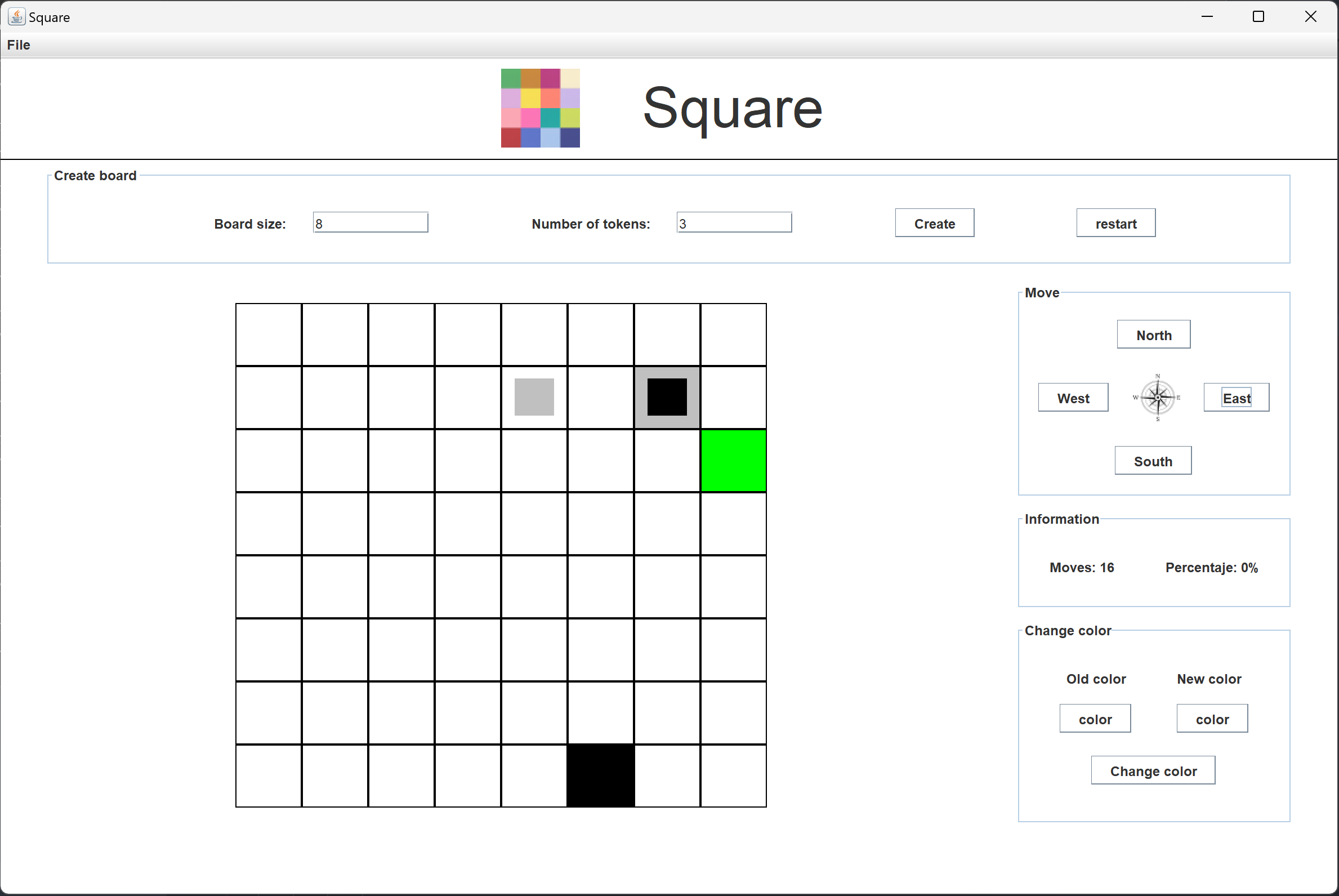
**2.** Implementen los elementos necesarios para reiniciar.

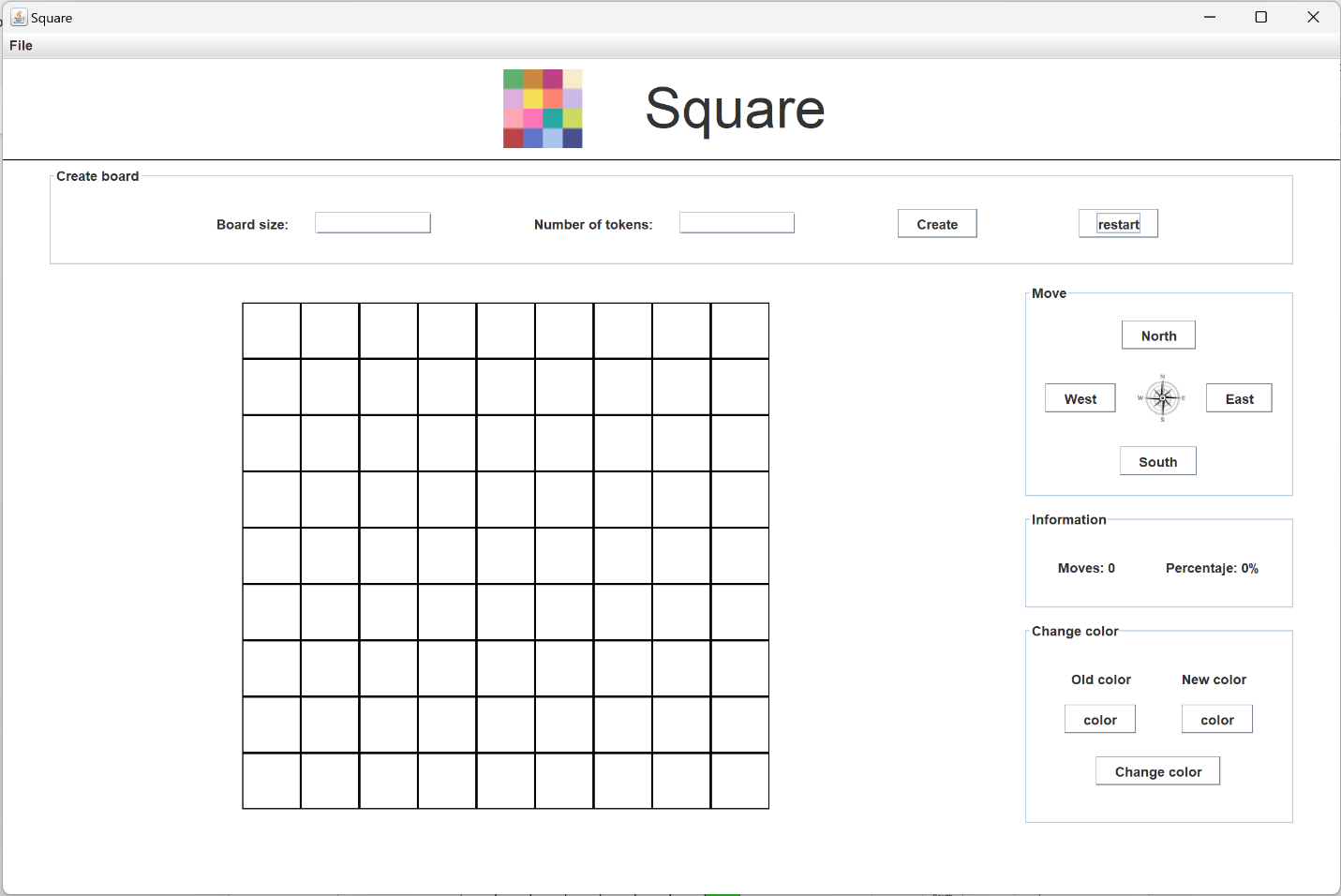
Rta:





**3.** Ejecuten el caso de uso y capture las pantallas más significativas.





**Ciclo 8: Cambiar el tamaño**

**El objetivo es implementar este caso de uso.**

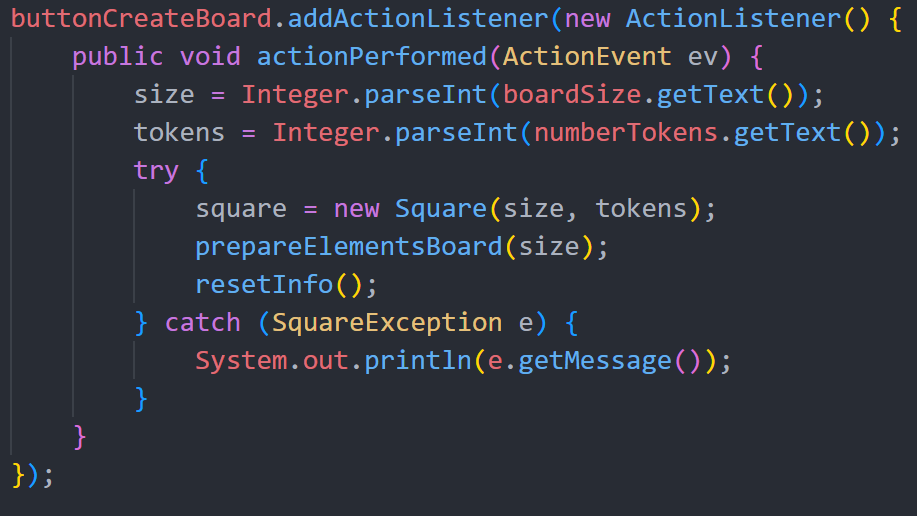
**1.** Expliquen los elementos a usar para implementar este caso de uso.

Rta:

Para el caso de modificar el tamaño del tablero se usará la misma implementacion para crear el tablero (ya antes descrito), ya que este "reinicia” el juego con el tablero del nuevo tamaño.

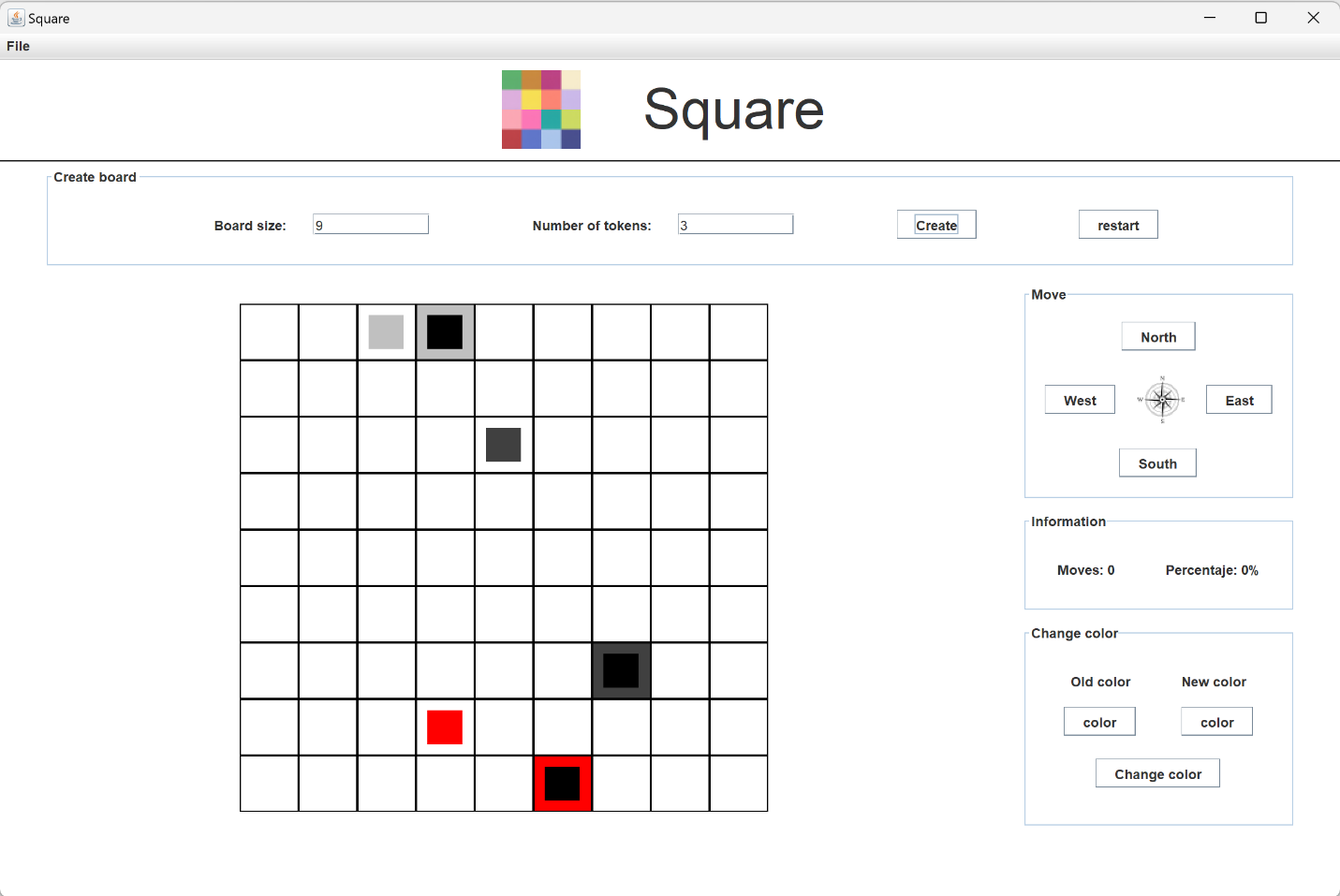
**2**. Implementen los elementos necesarios para cambiar el tamaño del juego.

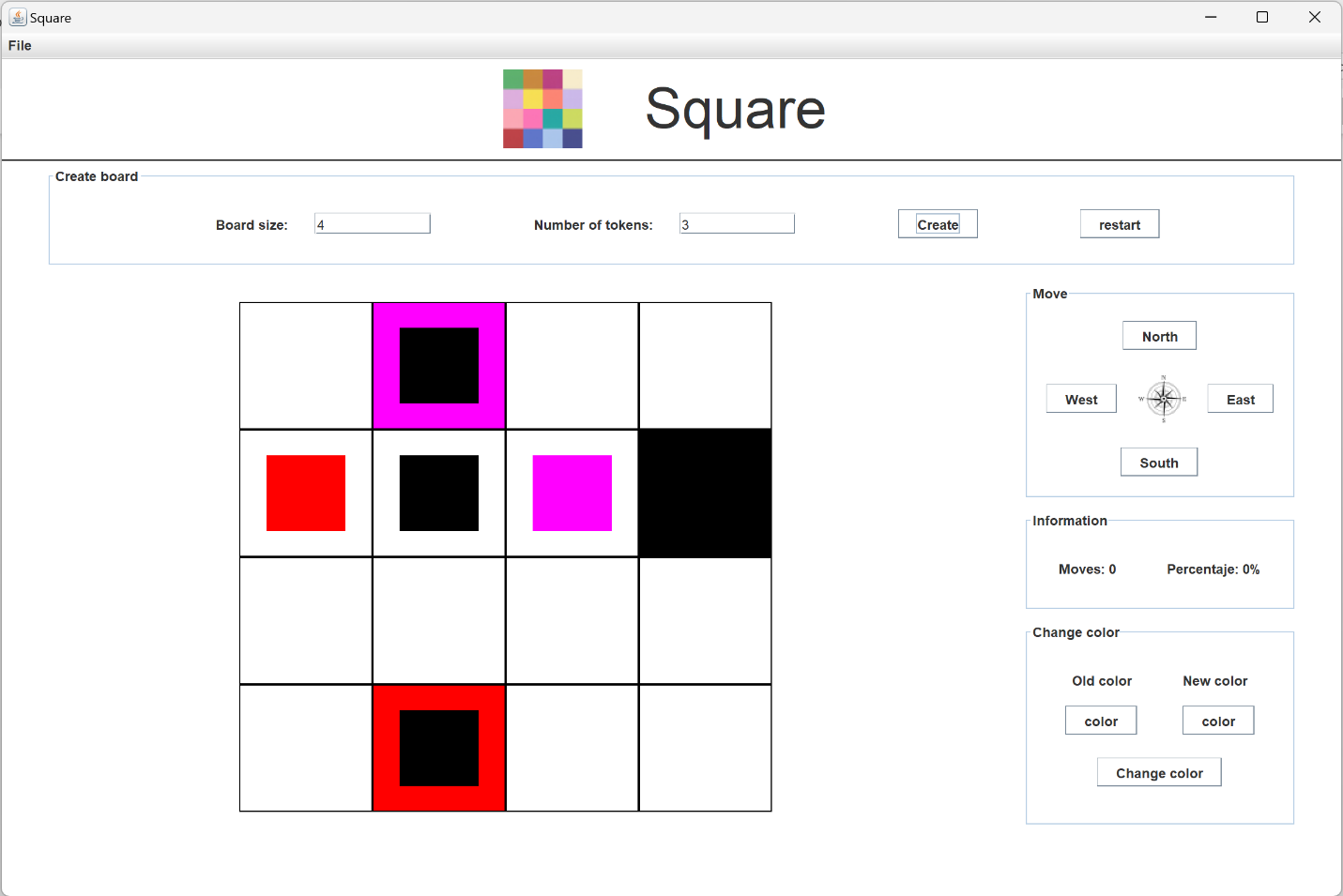
Rta:



**3.** Ejecuten el caso de uso y capture las pantallas más significativas.

Rta:





**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?

Daniel à 20h

Miguel 🡪 15h

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

Completado, porque esta vez distribuimos mejor el tiempo y trabajamos por ciclos, turnando los ciclos y distribuyendo las tareas de acuerdo a la afinidad de cada uno.

3. Considerando la práctica XP del laboratorio ¿por qué consideran que es importante?

Porque cuando veíamos que se presentaba un error, teníamos que corroborar que no se repitiera luego de haberlo arreglado, además, las pruebas exhaustivas demuestran que el código está diseñado para soportar los diferentes escenarios y manejarlos como nosotros lo designado. Es importante para mantener un código correcto.

4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor

problema? ¿Qué hicieron para resolverlo?

El haber representado el tablero y que mantuviera su relación con el tablero de la parte lógica, pues así el tablero se puede calcular desde la parte lógica y la interfaz hace una correcta representación de esta y se encarga de manejar los errores que pueda presentar la interacción con el usuario.

5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

Pudimos trabajar con una herramienta de Visual Studio Code que nos permitió codificar al mismo tiempo, así pudimos avanzar con mayor velocidad y así ambos tener un mejor dominio del código.