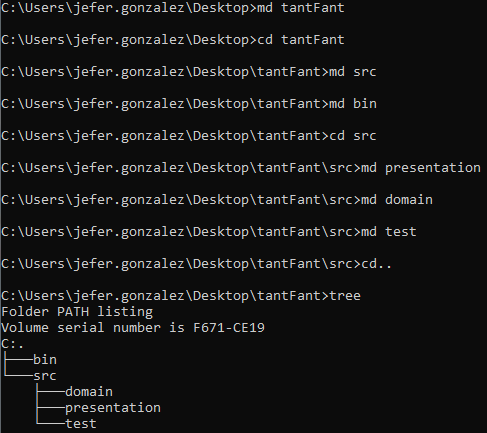
**Laboratorio 05. POOB-02**

Jefer Alexis González Romero  
Angel Nicolas Cuervo Naranjo

**DESARROLLO**

**Directorios**

**1.** Preparen un directorio llamado tantFant con los directorios src y bin y los subdirectorios para presentación, dominio y pruebas de unidad.

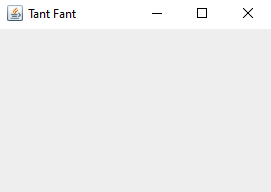


**Ciclo 0: Ventana vacía – Salir**

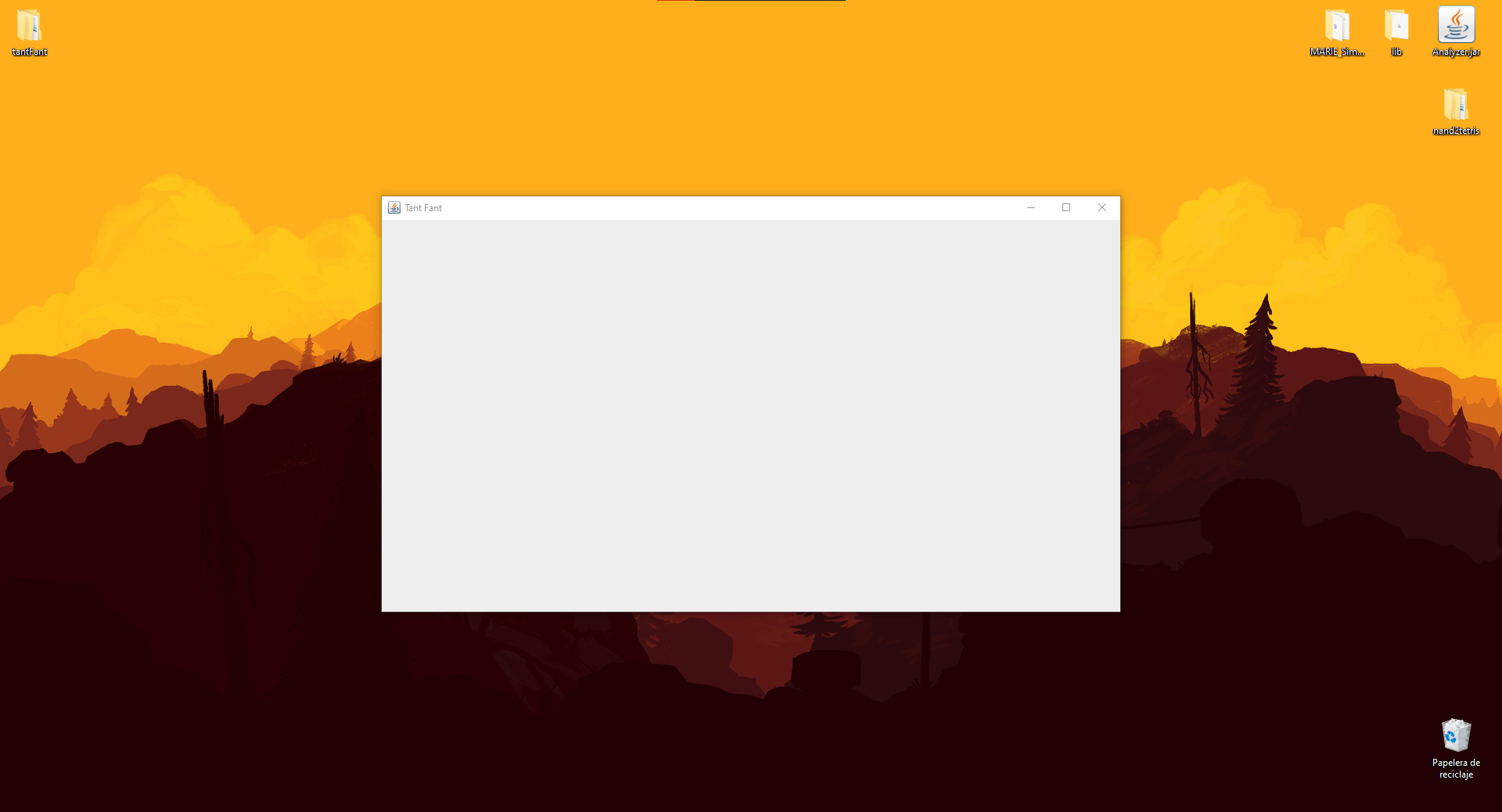
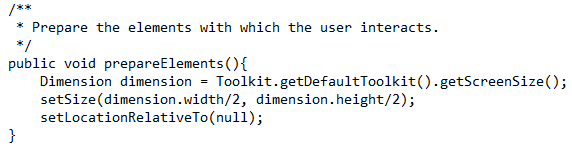
**1.** Construyan el primer esquema de la ventana de Tant Fant únicamente con el título “Tant Fant”. Para esto cree la clase Tant Fant como un JFrame con su creador (que sólo coloca el título) y el método main que crea un objeto Tant Fant y lo hace visible. Ejecútenlo. Capturen la pantalla.

Texto, Carta

Descripción generada automáticamente



**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.



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

No se termina la ejecución.

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media

Para terminar la ejecución toca oprimir *CTRL + C*.



Porque por defecto un *JFrame* cuando se cierra esconde la ventana, pero la deja disponible (HIDE\_ON\_CLOSE), así que no finaliza la aplicación.

**4.** Estudien en JFrame el método setDefaultCloseOperation. ¿Para qué sirve? ¿Cómo lo usarían en este caso?

Sirve para establecer el comportamiento de un *JFrame* al cerrarse, para este caso al método se le pasaría como parámetro JFrame.DO\_NOTHING\_TO\_CLOSE que su función es que cuando se cierra no hace nada, de modo que se le pregunte al jugador si está seguro de que quiere cerrar el juego y si oprime *cancel* no se hace nada, o si por el contrario selecciona *ok* se cerraría la aplicación.

**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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Ahora si se finaliza la aplicación.



**Ciclo 1: Ventana con menú – Salir**

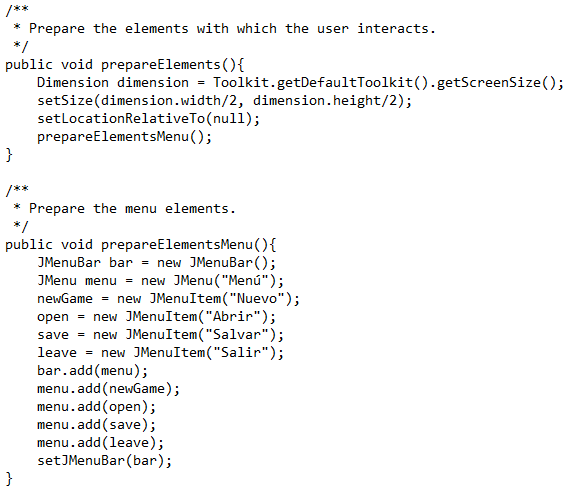
El objetivo es implementar un menú clásico para la aplicación con un final adecuado desde la opción del menú para salir. El menú debe ofrecer mínimo las siguientes opciones: Nuevo, Abrir – Salvar y Salir . Incluyan los separadores de opciones. No olviden los criterios de calidad del código.

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

Para el menú se necesitan un componente *JMenuBar: bar,* un *JMenu*: *menu* y cuatro *JMenuItem: newGame*, *open*, *save* y *exit.*

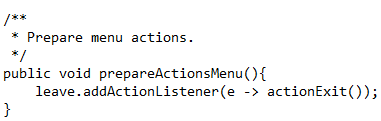
Serían atributos los componentes *JMenuItem*, ya que se necesitarán definir sus acciones en el método *prepareActions().* En cambio, el *JMenu* y el *JMenuBar* solo son necesarios en el *prepareElements()* donde se les añadirá las opciones y el menú respectivamente.

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

 Rectángulo

Descripción generada automáticamente con confianza media

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



Forma, Rectángulo

Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Ciclo 2: Salvar y abrir**

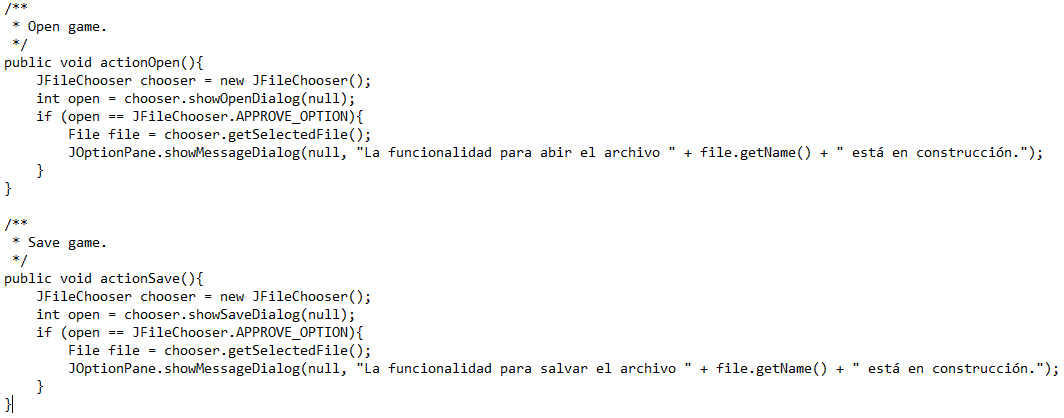
El objetivo es preparar la interfaz para las funciones de persistencia

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

*JFileChooser* es una clase de java que proporciona un mecanismo simple que muestra una ventana de herramientas para la selección de ficheros, algunos de sus métodos son:

* **JFileChooser:** Construye un JFileChooser apuntando al directorio default del usuario.
* **showOpenDialog:** Muestra un cuadro de dialogo que permite abrir un archivo.
* **showSaveDialog:** Muestra un cuadro de dialogo que permite salvar un archivo.
* **getSelectedFile:** Retorna el archivo seleccionado.

**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.

Texto, Carta

Descripción generada automáticamente

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

**Abrir un archivo:**

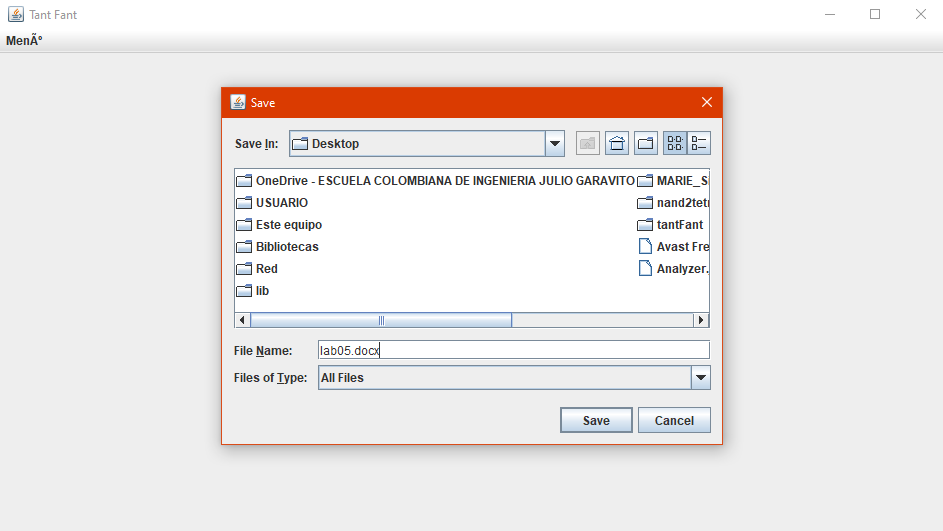
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

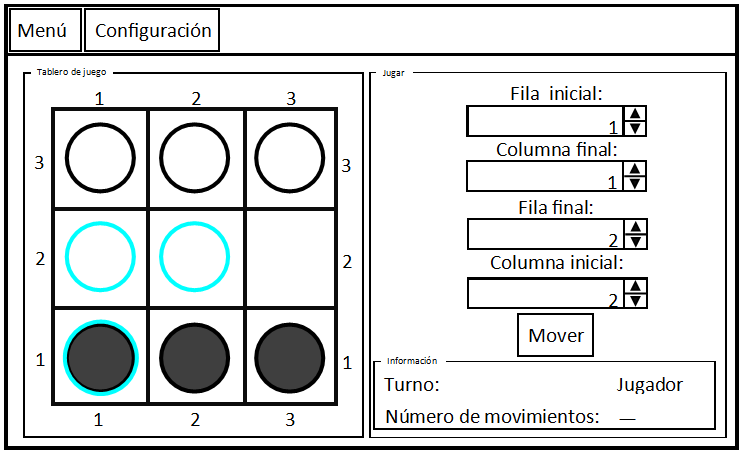
**Salvar un archivo:**

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

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

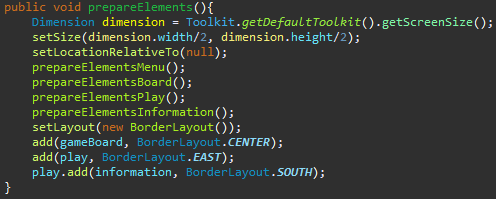
**1.** Presenten el bosquejo del diseño de interfaz con todos los componentes necesarios.

Diagrama

Descripción generada automáticamente

**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.

Texto

Descripción generada automáticamente

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

Imagen que contiene Calendario

Descripción generada automáticamente

**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.

En el menú *Configuración* se va a añadir un nuevo *JMenu* llamado *Colores* el cual abrirá un cuadro de dialogo donde habrá un *JPanel* llamado *setting* el cual tendrá 5 *JLabel:* *playerOne, playerTwo, possibleMoves, backgroundColor y borderColor,* y para cada uno de los anteriores su respectivo *JButton* que tendrán un *ActionListener* los cuales harán que al seleccionarlos abran una ventana donde se pueda seleccionar el color a través de un JColorChooser*.* Además, habrá un *JButton* para reflejar los cambios en el tablero (*ok*) al seleccionarlo también se cerrará el cuadro de dialogo de configuración, otro *JButton* para restaurar los colores predeterminados (*restore)* y un último para cancelar los cambios (*cancel*) que cerrará el cuadro de dialogo de configuración sin hacer ningún cambio. Cada uno de los últimos *JButton* tendrán su *ActionListener* para que al oprimirlos cumplan con su función.

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

* ***JColorChooser:*** Como su nombre lo indica permite elegir colores, esto a través de un panel de controles que permiten manipular y seleccionar colores.
* ***showDialog:*** Muestra un dialogo para presentar un selector de color modal.

**3.** Implementen los componentes necesarios para cambiar el color del tablero.

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

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

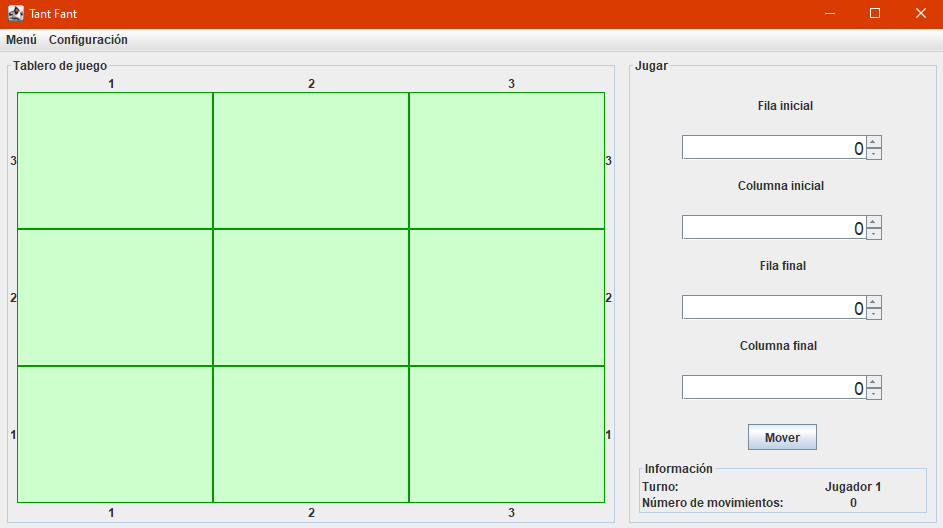
Interfaz de usuario gráfica

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente

**Ciclo 5: Modelo Tant Fant**

El objetivo es implementar la capa de dominio para Tant Fant.

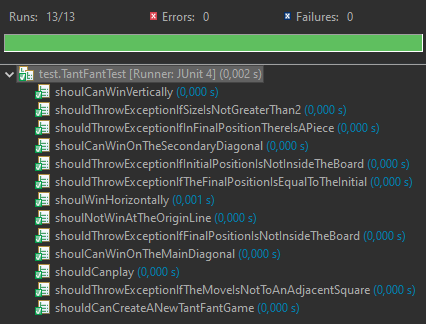
**1.** Construya los métodos básicos del juego (No olvide MDD y TDD).

Imagen que contiene Tabla

Descripción generada automáticamente

En TantFant.astah y en TantFant.java

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



En TantFantTest.java

**Ciclo 6: Jugar**

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

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

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

**Para hacer los movimientos manualmente:**

En el *JPanel play* se adicionan 4 *JLabel* para la fila inicial, columna inicial, fila y columna finales, cada una tendrá un debajo de ellas un *JSpinner* donde se pondrá el valor que se desee. Además, habrá un *JButton (move)* para realizar el movimiento.

**Para hacer los movimientos en el tablero:**

El tablero estará conformado por varios *JPanel* cada uno tendrá un botón que lo cubrirá totalmente y si tienen una ficha o es un movimiento posible de la ficha seleccionada, esta se dibujará en él.

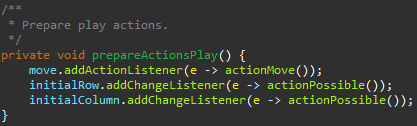
**4.** Implementen los componentes necesarios para jugar .¿Cuántos oyentes necesitan? ¿Por qué?

**Para hacer los movimientos manualmente:**

Se necesita un *ActionListener* para el botón *move* que al oprimirlo obtendrá los valores de los *JSpinner* para realizar el movimiento llamando el método *play()* del modelo, se reflejará los cambios de este si el movimiento es válido, si no, se presentará un cuadro de dialogo explicando que no se pudo realizar el movimiento. Si con el movimiento un jugador gana se presentará un cuadro de dialogo indicando esto y se reiniciará el tablero.

**Para hacer los movimientos en el tablero:**

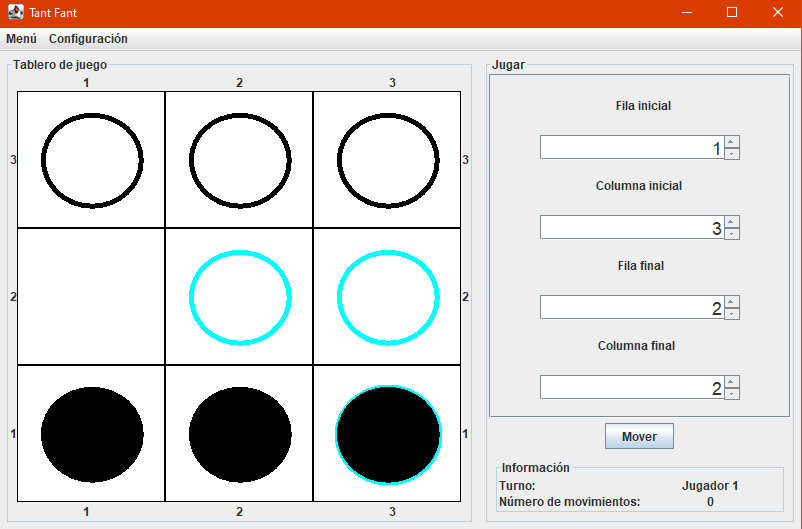
Los botones en el tablero cada uno tendrá un oyente, si en la casilla hay una ficha se reflejara en el tablero los posibles movimientos que se pueden hacer, o si por el contrario está vacía, si se presiono antes una casilla con ficha y el movimiento es válido se moverá la ficha, si no es valido se mostrará un dialogo indicando que no se pudo realizar el movimiento.

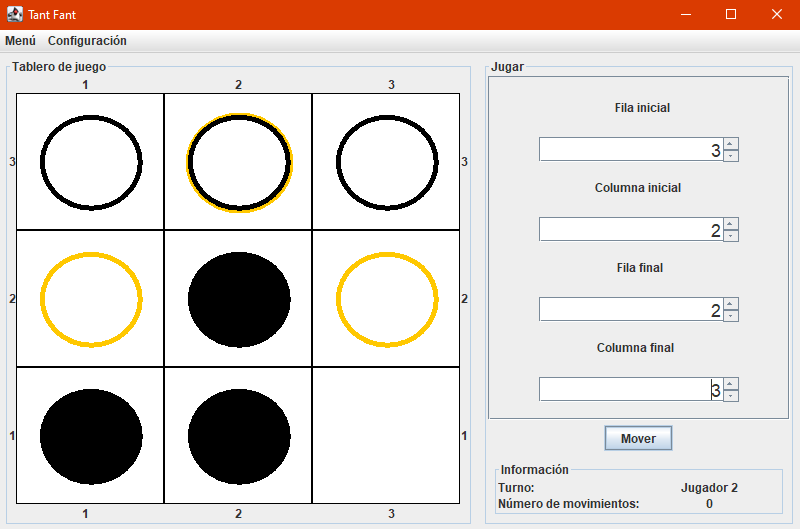
Texto

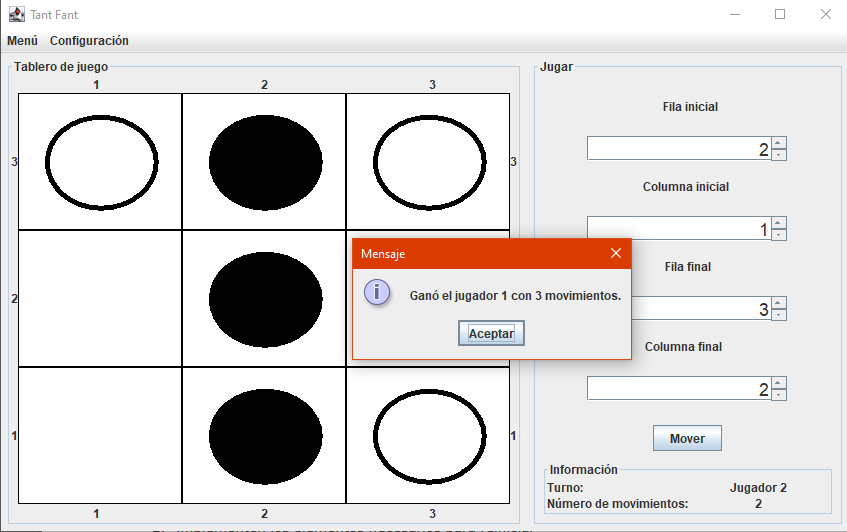
Descripción generada automáticamente

En TantFantGUI.java

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

Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica

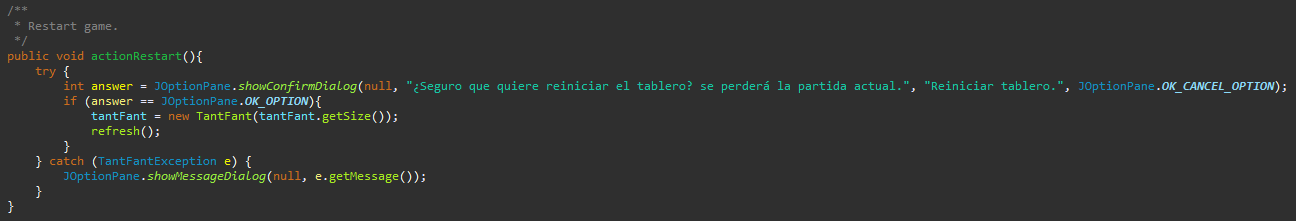
Descripción generada automáticamente

**Ciclo 7: Reiniciar**

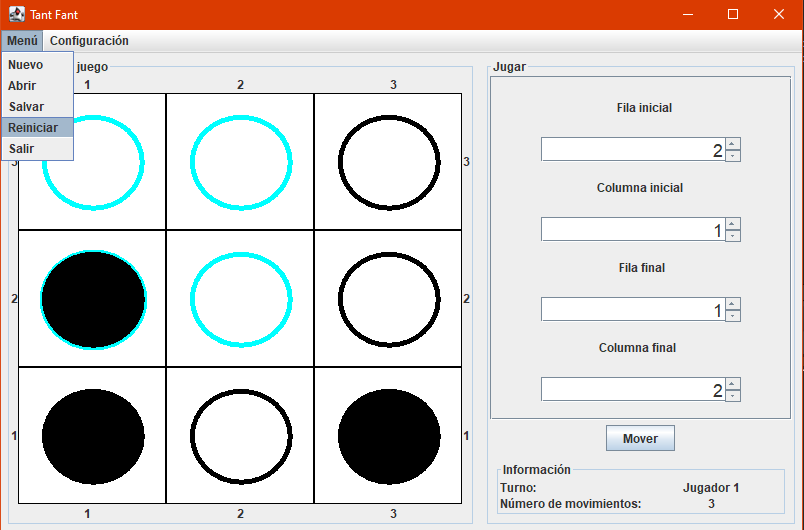
**1.** Expliquen los elementos a usar para implementar este caso de uso.

En el *JMenu Menú* se incluye un nuevo *JMenuItem* llamado *Reiniciar*, el cual tendrá un *ActionListener* que al presionarlo muestre un cuadro de dialogo que le pregunte al usuario si quiere reiniciar el tablero, si oprime *ok* se modifica el tablero, o si por el contrario le da en *Cancel* no se hace ningún cambio.

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



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

Interfaz de usuario gráfica

Descripción generada automáticamenteInterfaz de usuario gráfica

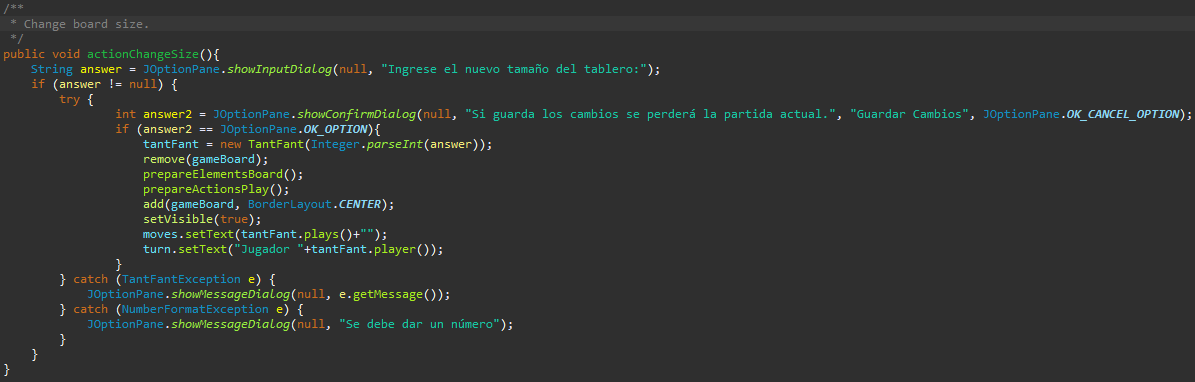
Descripción generada automáticamente

**Ciclo 8: Cambiar el tamaño**

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

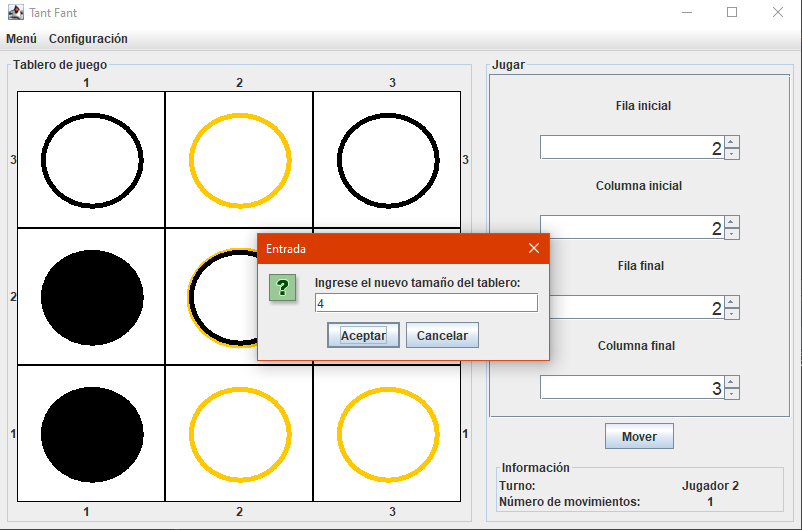
En el JMenú *Configuración* se agrega un nuevo *JMenuItem* llamado *Tamaño* el cual tendrá un *ActionListener* que hará que al presionarlo abra un cuadro de dialogo donde le preguntará por el nuevo tamaño, si ingresa un valor y le da *ok* se le preguntará si desea guardar los cambios, si es así se modifica el tablero con el nuevo tamaño y se comienza una nueva partida. Si por el contrario le da cancelar no se hace cambia nada.

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



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

Forma

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente

**RETROSPECTIVA**

**1.** ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)

15 horas por cada uno.

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

Completo.

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

La práctica *When a bug is found* es importante ya que es útil para cuando se identifica un error se creen pruebas para testearlo más rápido y cuando se solucione prevenir que no vuelva a surgir por algún cambio realizado en la aplicación.

La práctica *Acceptance tests* es útil para comprobar que al usuario se le presenta de forma adecuada el juego con todas las funcionalidades requeridas.

**4.** ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?

Nuestro mayor logro fue hacer el tablero del juego, ya que fue difícil su implementación.

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

Destinar el tiempo suficiente para el desarrollo del laboratorio, un adecuado manejo de las excepciones y los casos especiales que se podían presentar. Comprender y analizar oportunamente las implementaciones que trae consigo la clase *Swing* para futuras implementaciones de Back End y Front End.