Angie Medina – Jose Pérez

POOB - 1

2020 - 2

Laboratorio 5/6 GUI

**CONTEXTO**

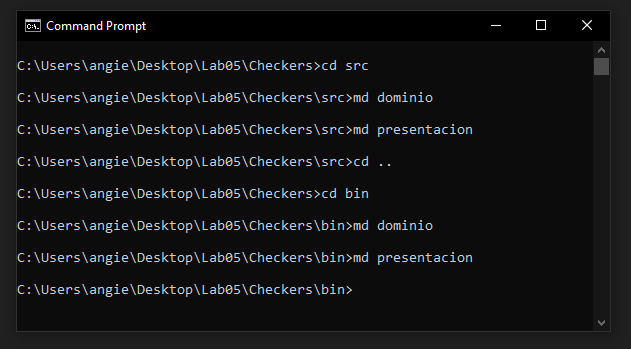
El objetivo es implementar el juego ***Checkers*** en versión usuario-computador. El usuario juega desde la zona sur, el usuario siempre inicia y el computador juega al azar. En caso de salto automático, si se debe decidir se selecciona dando prioridad a los movimientos de avance y al movimiento a la derecha.

**DESARROLLO**

**Directorios**

El objetivo de este punto es construir un primer esquema para el juego *Checkers*.

1. Preparen un directorio llamado *Checkers* con los directorios *src* y *bin* y los subdirectorios presentación y dominio



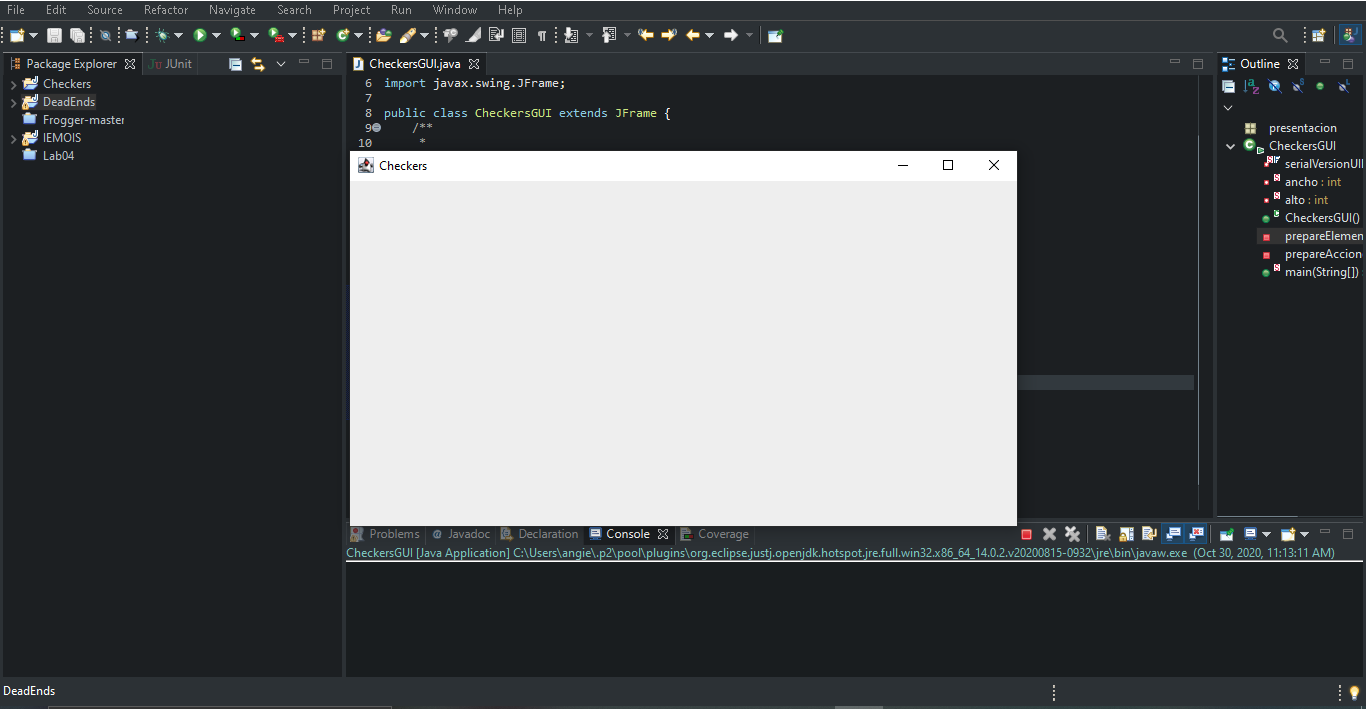
**Ciclo 0: Ventana vacía – Salir**

El objetivo es implementar la ventana principal de *Checkers* con un final adecuado a la aplicación desde el icono de cerrar. Utilizar el esquema de *prepareElementos-prepareAcciones.*

1. Construyan el primer esquema de la ventana de *Checkers* únicamente con el título *“Checkers”*. Para esto cree la clase *CheckersGUI* como un *JFrame* con su creador, que sólo coloca el título, y el método *main* que crea un objeto *CheckersGUI* 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)



1. Modifiquen el tamaño de la ventana para que ocupe un cuarto de la pantalla y ubíquenla en el centro (*prepareElementos*). Capturen esa pantalla.



1. Traten de cerrar la ventana.

* ¿Termina la ejecución?

Se oculta la ventana

* ¿Qué deben hacer para terminar la ejecución? ¿Por qué?

Usar *setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE)*, con esto la operación por defecto al pulsar cerrar cambia de ocultarse a terminar la aplicación. (¿)

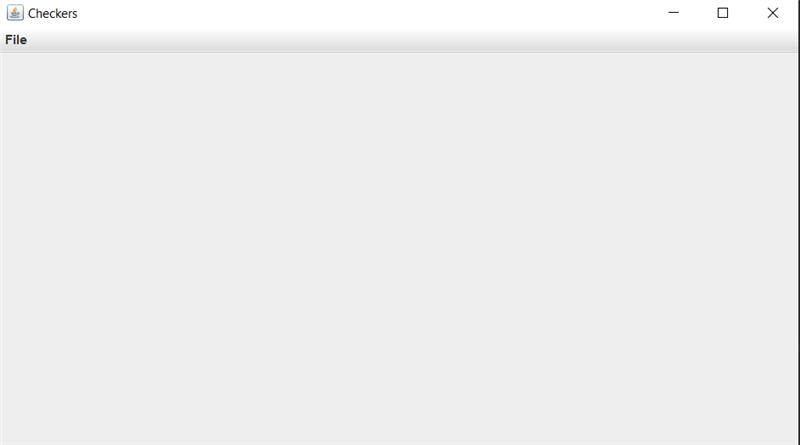
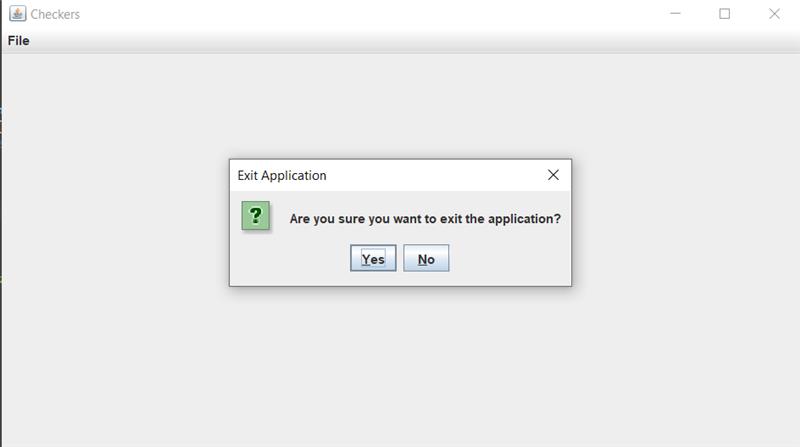
1. Estudien en *JFrame* del método *setDefaultCloseOperation*.
   * ¿Para qué sirve?

Sirve para cambiar que se hace al momento de intentar cerrar la ventana, se pueden usar las siguientes constantes:

* JFrame.EXIT\_ON\_CLOSE — Cierra la aplicación.
* JFrame.HIDE\_ON\_CLOSE — Oculta el JFrame, pero mantiene la ejecución.
* JFrame.DISPOSE\_ON\_CLOSE — Dispone el JFrame, pero mantiene la ejecución.
* JFrame.DO\_NOTHING\_ON\_CLOSE — Ignora el click.
  + ¿Cómo lo usarían en este caso?

Se usa *setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE)* para evitar que la ventana se cierre automáticamente

1. Preparen el “oyente” correspondiente al icono cerrar que le pida al usuario que confirme su selección. Para esto Implementen parcialmente el método *prepareAcciones* y el método asociado a la acción (*salga*). Ejecuten el programa y salgan del programa. Capturen las pantallas.

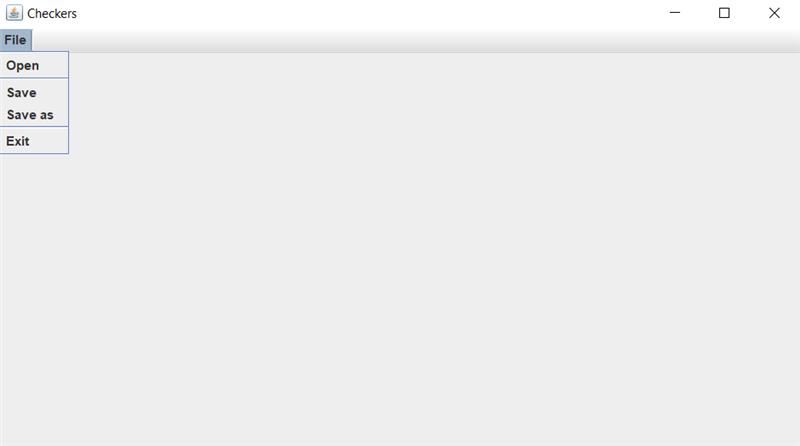
 

**NOTA:** En la versión final, la confirmación antes de cerrar solo aparece cuando se realiza al menos un movimiento y no se ha guardado la partida.

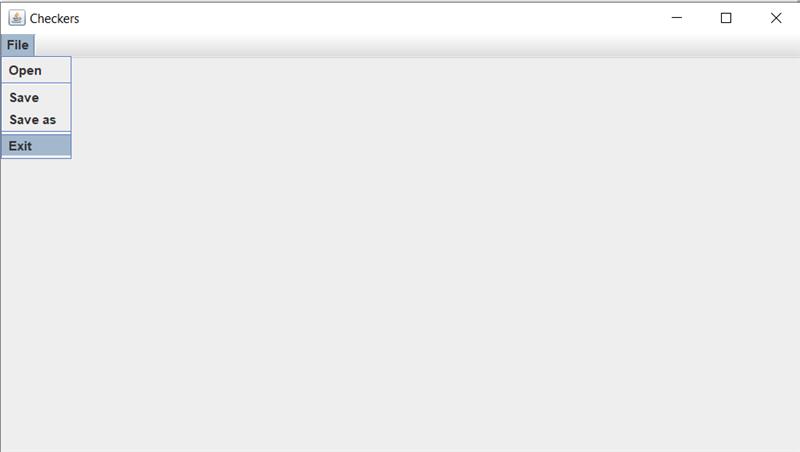
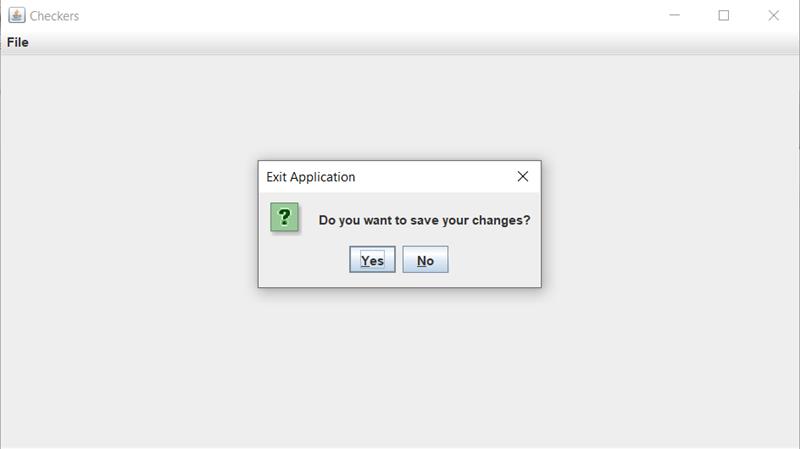
**Ciclo 1: Ventana con menú – Salir**

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, Salvar Como – Salir) . No olviden los criterios de calidad del código.

1. Defina como atributos los componentes visuales necesarios del menú.
2. Construya la forma del menú propuesto en su diseño de interfaz (*prepareElementos -prepareElementosMenu*) . Ejecuten. Capturen la pantalla.



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

**Ciclo 2: Salvar y abrir**

El objetivo es preparar la interfaz para las funciones de persistencia

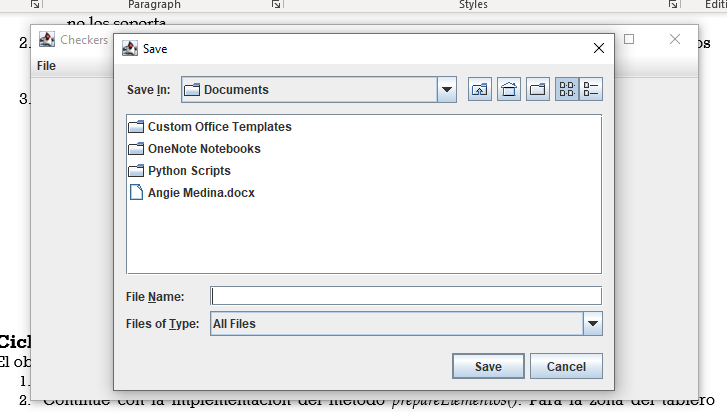
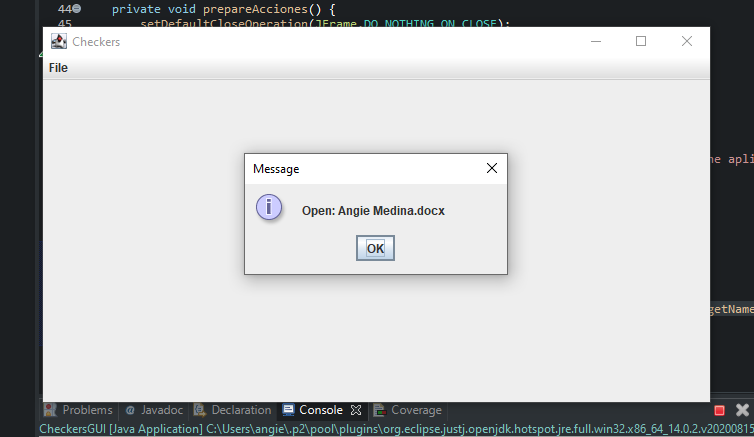
1. Detalle el componente JFileChooser especialmente los métodos :*JFileChooser*, *showOpenDialog*, *showSaveDialog*, *getSelectedFile*.

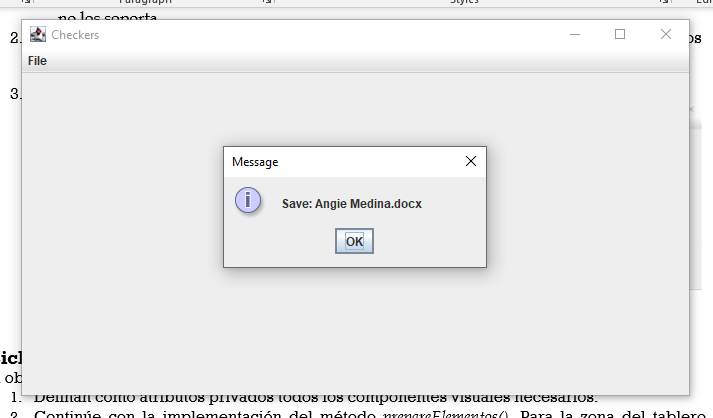
**JFileChooser** es una clase java que nos permite mostrar fácilmente una ventana para la selección de un fichero.

* **JFileChooser()**: Construye un JFileChooser apuntando al directorio default del usuario
* **showOpenDialog(Component parent):**Abre una ventana dialogo de “Abrir archivo”. El método tira la excepción *HeadlessException.*
* **showSaveDialog(Component parent):**Abre una ventana dialogo de “Guardar archivo”. El método tira la excepción *HeadlessException.*
* **getSelectedFile():**Retorna el archivo seleccionado. Este puede ser determinada por el programador vía el método *setSelectedFile* o por una acción del usuario, como escribir el nombre del archivo en el GUI o seleccionar el archivo de una lista provista por el GUI

**Nota:** La excepción HeadlessException mencionada anteriormente es lanzada cuando el código que es dependiente de un teclado, mouse, o display es llamado en un entorno que no los soporta.

1. 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.
2. Ejecuten las dos alternativas y capturen las pantallas más significativas.

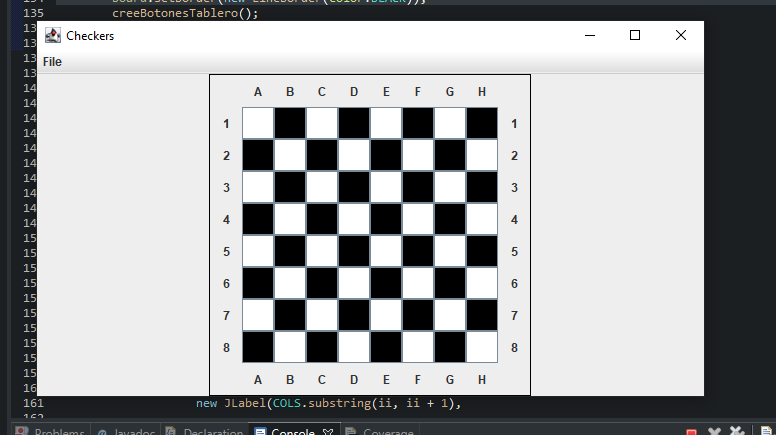
 



**Ciclo 3: Forma de la ventana principal**

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

1. Definan como atributos privados todos los componentes visuales necesarios.
2. Continúe con la implementación del método *prepareElementos()*. Para la zona del tablero defina un método *prepareElementosTablero()* y un método *refresque()* que actualiza la vista del tablero considerando, por ahora, un tablero inicial por omisión (el ejemplo del trabajo en clase) Este método lo vamos a implementar realmente en otros ciclos. Ejecuten y capturen esta pantalla.



**Ciclo 4: Cambiar color**

El objetivo es implementar este caso de uso.

1. Expliquen los elementos necesarios para implementar este caso de uso.

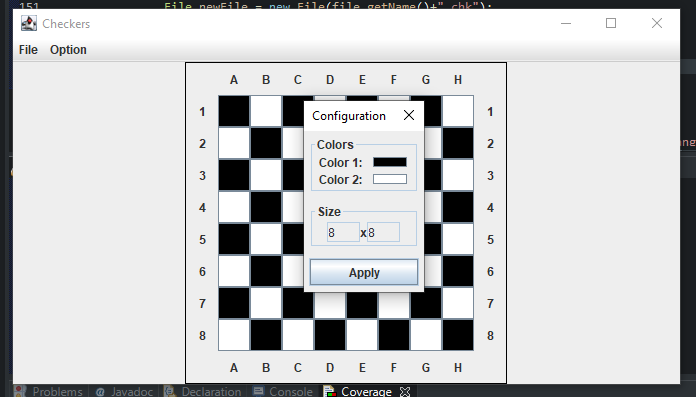
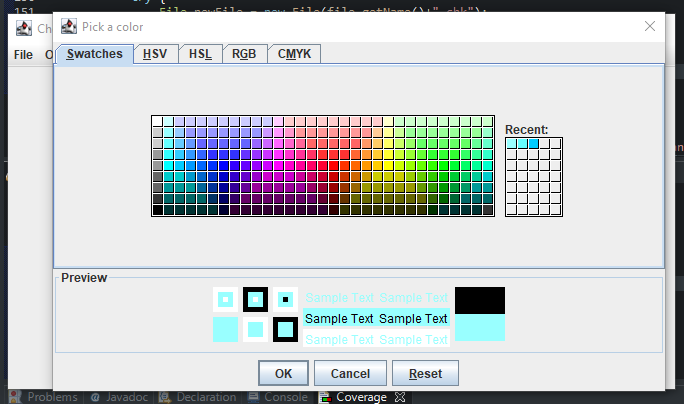
Se agregó a la barra de menú un nuevo menú llamado *Option* el cual tiene un ítem llamado *Settings,* este abrirá una ventana (se implementó con una nueva clase *ConfigWindow*) que contendrá dos botones, se tiene un *Listener* asociado a ellos para que cuando se le de click a alguno de estos aparezca la ventana dialogo característica de *JColorChooser.*

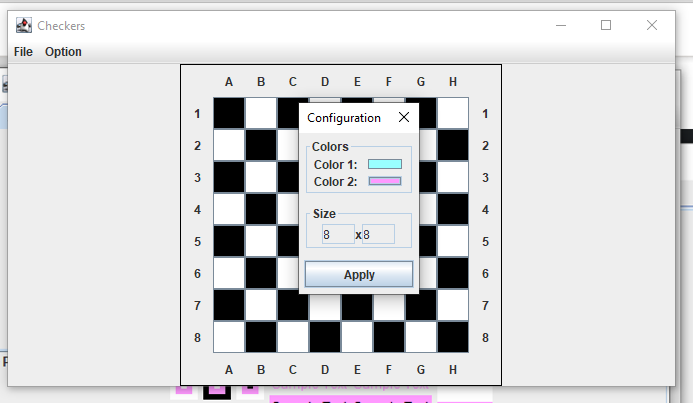
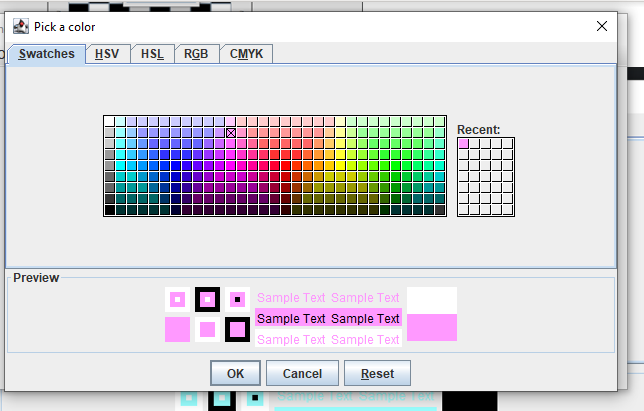
1. Detalle el comportamiento de *JColorChooser* especialmente el método estático *showDialog*

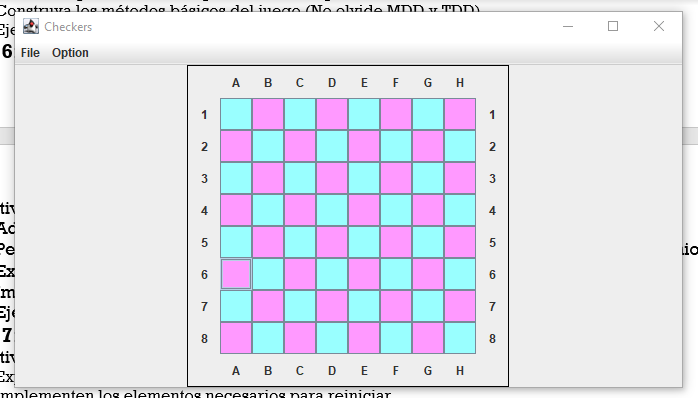
*JColorChooser* es un componente que nos ayuda a seleccionar un color de una forma más atractiva para el usuario.

**public static Color showDialog(Component, String title, Color initialColor):** Muestra un diálogo y un selector de color modal hasta que se oculta el diálogo. Si el usuario presiona el botón "Aceptar", este método oculta / elimina el cuadro de diálogo y devuelve el color seleccionado. Si el usuario presiona el botón "Cancelar" o cierra el cuadro de diálogo sin presionar "Aceptar", este método oculta / elimina el cuadro de diálogo y devuelve nulo.

1. Implementen los componentes necesarios para cambiar el color del tablero (inicialmente blanco-negro)
2. Ejecuten el caso de uso y capture las pantallas más significativas.

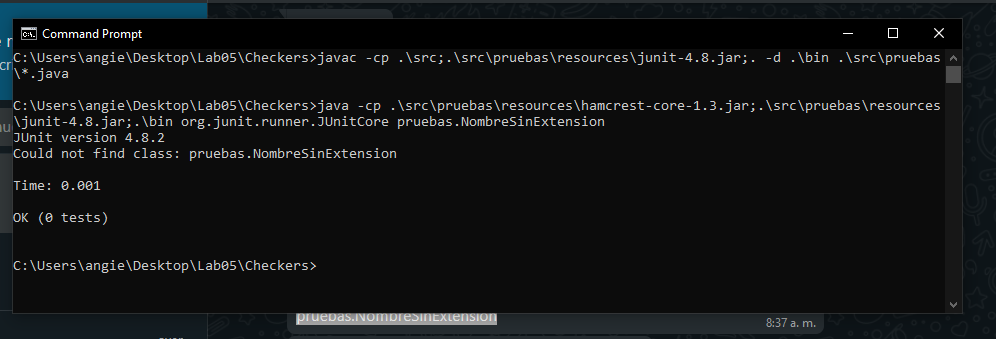




**Ciclo 5: Modelo Checkers**

El objetivo es implementar la capa de dominio para *Checkers*.

1. Construya los métodos básicos del juego (No olvide MDD y TDD)
2. Ejecuten las pruebas y capturen el resultado.



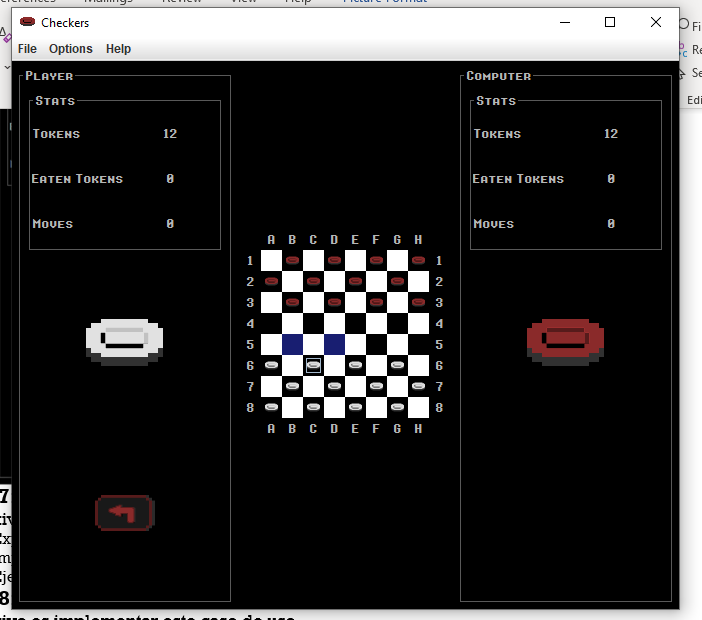
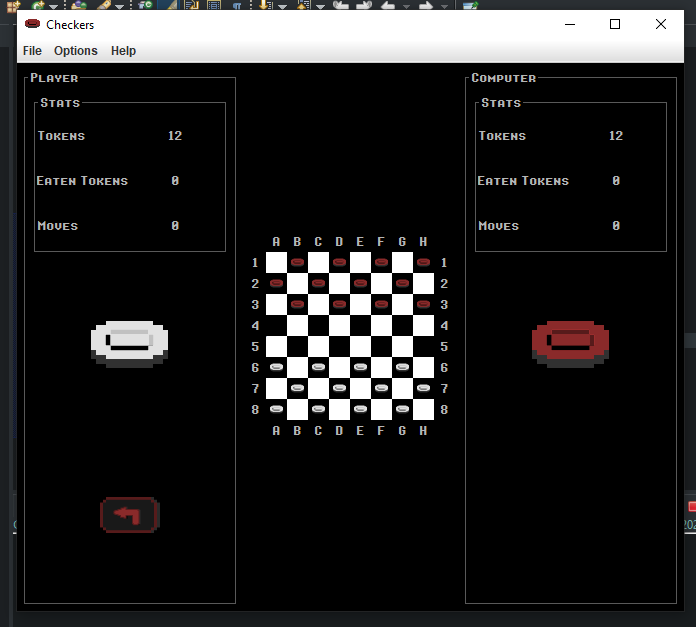
**Ciclo 6: Jugar**

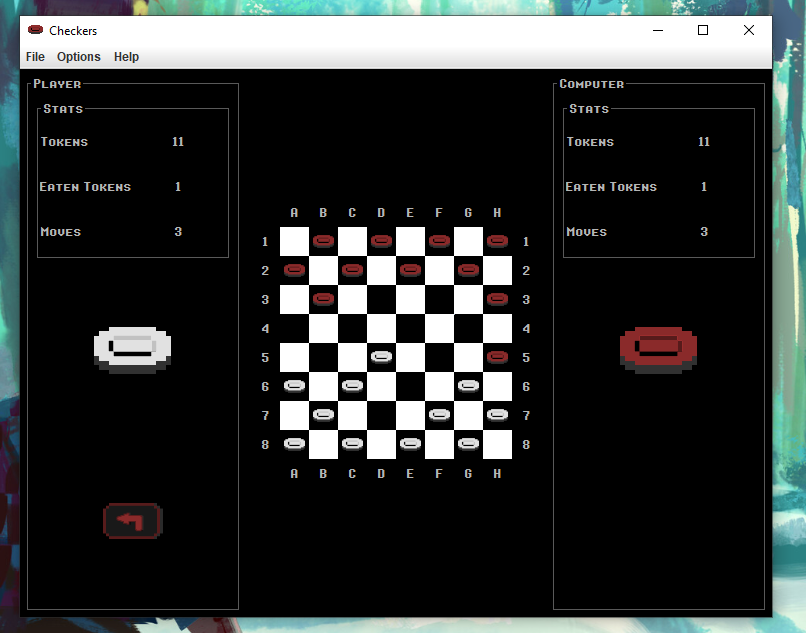
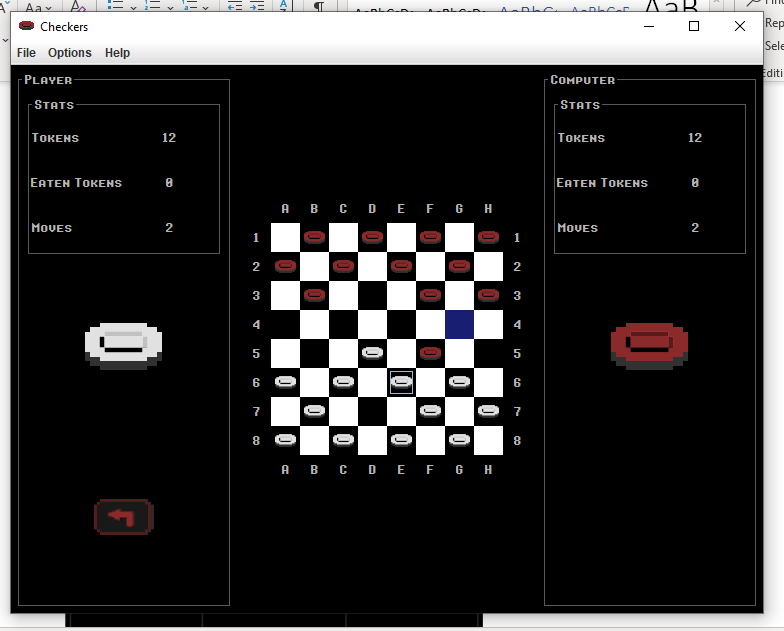
El objetivo es implementar el caso de uso jugar.

1. Adicione a la capa de presentación el atributo correspondiente al modelo.
2. Perfeccionen el método *refresque()* considerando la información del modelo de dominio.
3. Expliquen los elementos necesarios para implementar este caso de uso.

Cada casilla del tablero es un botón (un *Listener* responde al click de estos), al dar click sobre una ficha propia esta se seleccionará y se tendrá que dar click al lugar donde puede moverla. Esta selección y el movimiento se logran con métodos dentro del GUI, que se comunican con la clase en la capa de dominio, adicional a esto se pide que refresque el tablero.

1. Implementen los componentes necesarios para jugar.
2. Ejecuten el caso de uso y capture las pantallas más significativas.





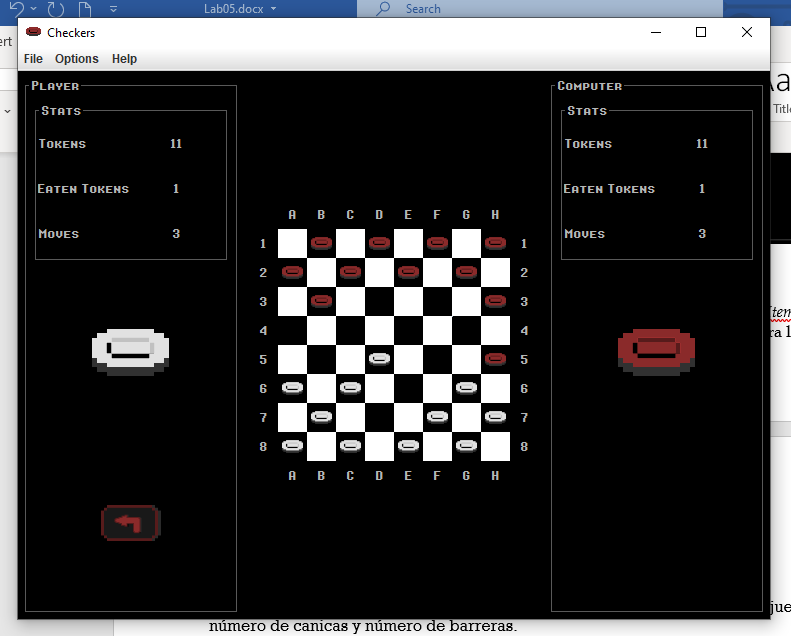
**Ciclo 7: Reiniciar**

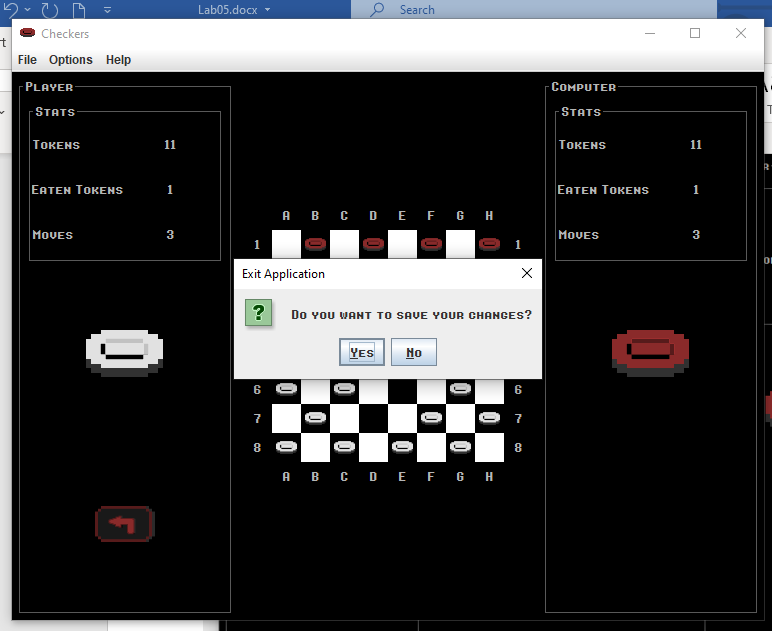
El objetivo es implementar este caso de uso.

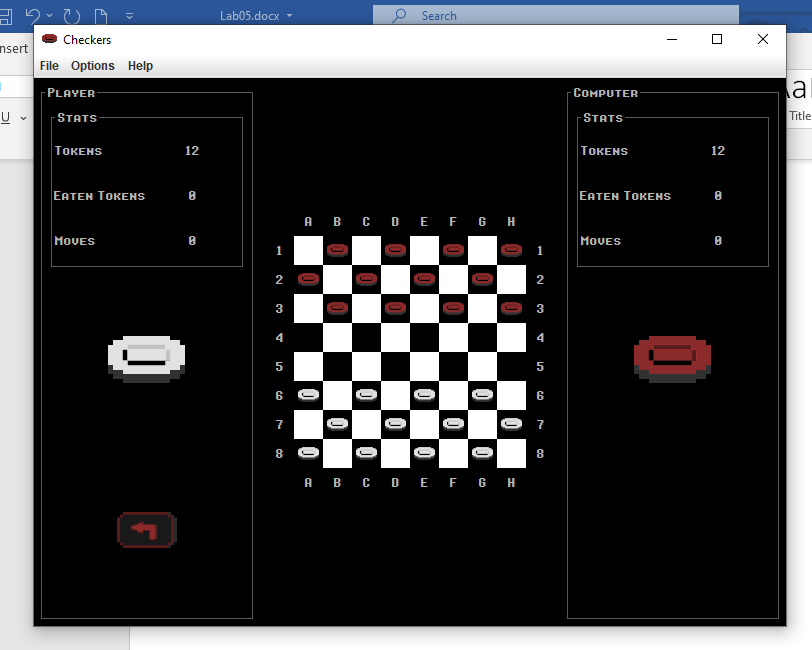
1. Expliquen los elementos a usar para implementar este caso de uso.

Se le agregó un nuevo ítem llamado *Restart,* se tiene un *Listener* asociado a este *Item* que llama a un nuevo método llamado *reinicie*, este crea una nueva instancia de *Checkers* para luego utilizar el método *refresque*

1. Implementen los elementos necesarios para reiniciar.
2. Ejecuten el caso de uso y capture las pantallas más significativas.



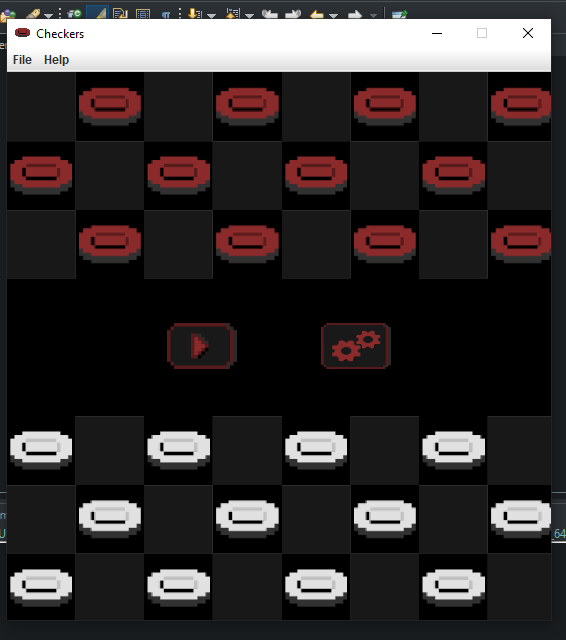
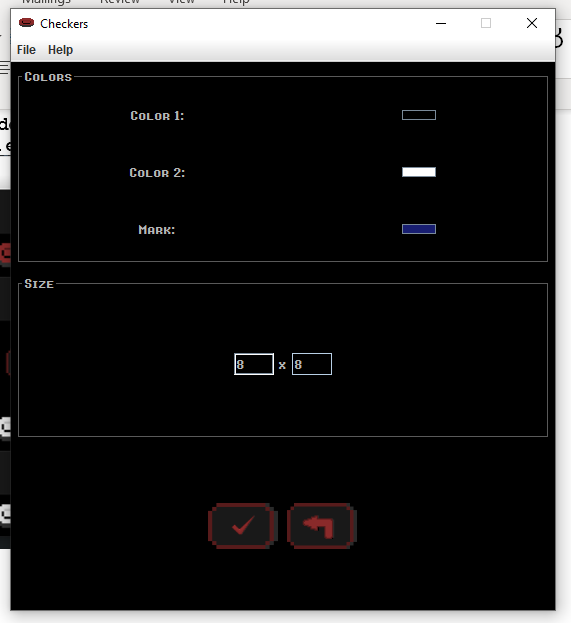


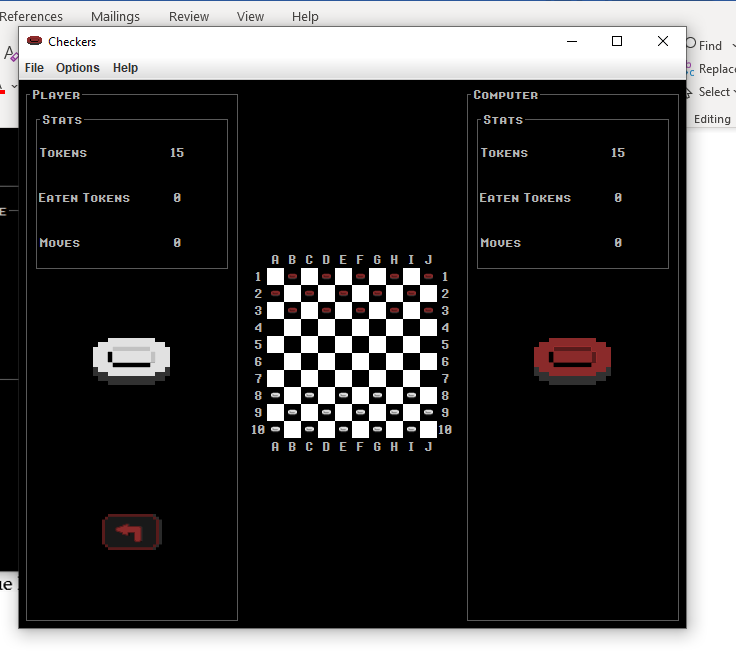
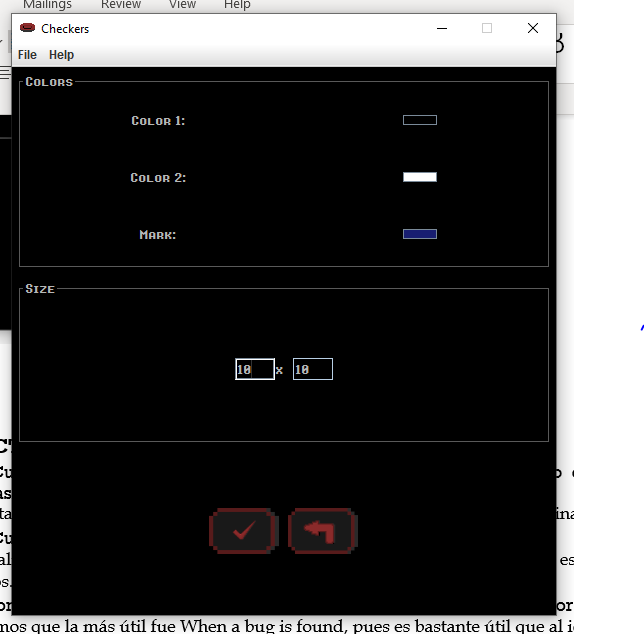


**Ciclo 8: Cambiar la configuración del juego**

El objetivo es implementar este caso de uso.

1. Expliquen los elementos a usar para implementar este caso de uso
2. Implementen los elementos necesarios para cambiar la configuración del juego: tamaño, número de canicas y número de barreras.
3. Ejecuten el caso de uso y capture las pantallas más significativas



**RETROSPECTIVA**

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

Se estima que a este laboratorio le invertimos 22 horas cada uno, 23 h/ Angie Medina – Jose Perez

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

Se realizo todo el laboratorio, pues le invertimos el suficiente tiempo.

1. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

Creemos que la más útil fue *When a bug is found*, pues es bastante útil que al identificar un bug se creen pruebas para testearlo más rápido y cuando se solucione prevenir que no vuelva a surgir por algún cambio realizado en la aplicación

1. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Consideramos que nuestro mayor logro fue hacer que todo funcionara en un conjunto, también nos sentimos que la interfaz grafica nos quedó muy bien.

1. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

Nos tomó bastante tiempo hacer que los componentes en la interfaz se comportaran y quedaran como queríamos, hicimos consultas por internet de cada componente.

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

Consideramos que nos comunicamos bien como equipo y así mismo tratamos de implementar las practicas xp, pero igual nos comprometemos a mejorar en la implementación de las practicas ya vistas en pasadas entregas junto con las nuevas vistas en este laboratorio.