PROGRAMACIÓN ORIENTADA A OBJETOS

## INTERFAZ 2022-1

## Laboratorio 5/6

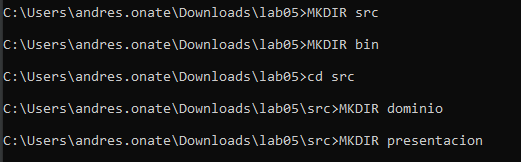
## Andrés Camilo Oñate Quimbayo, Nicolas Ariza Barbosa

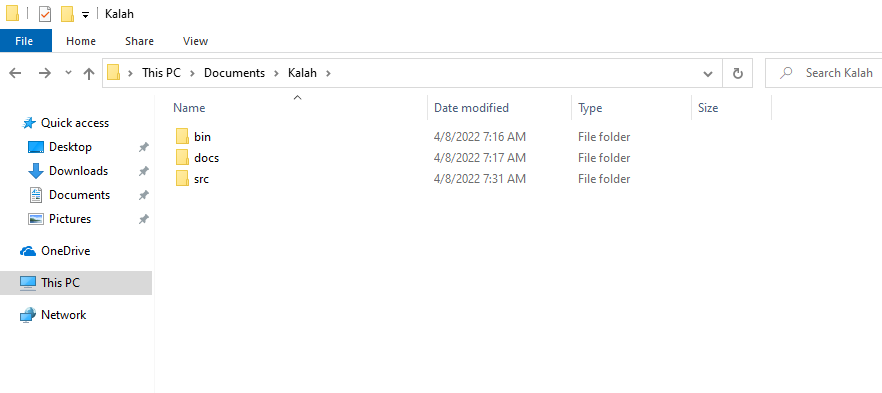
## DESARROLLO

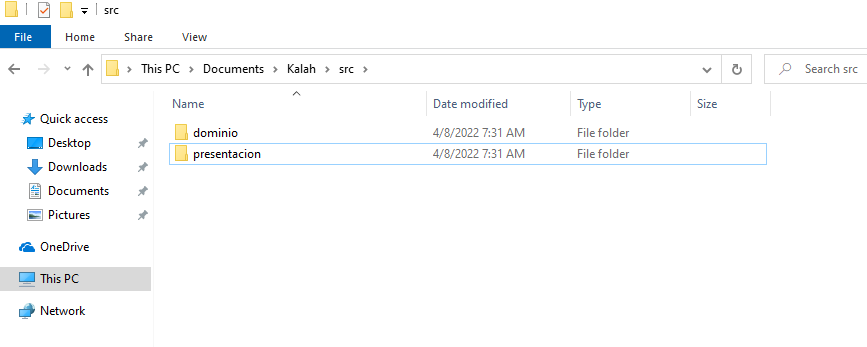
## Directorios

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

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



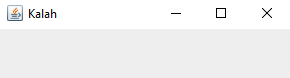


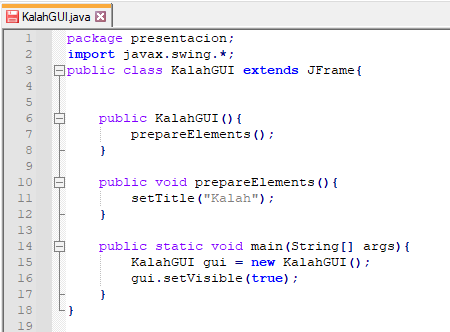


## Ciclo 0: Ventana vacía – Salir

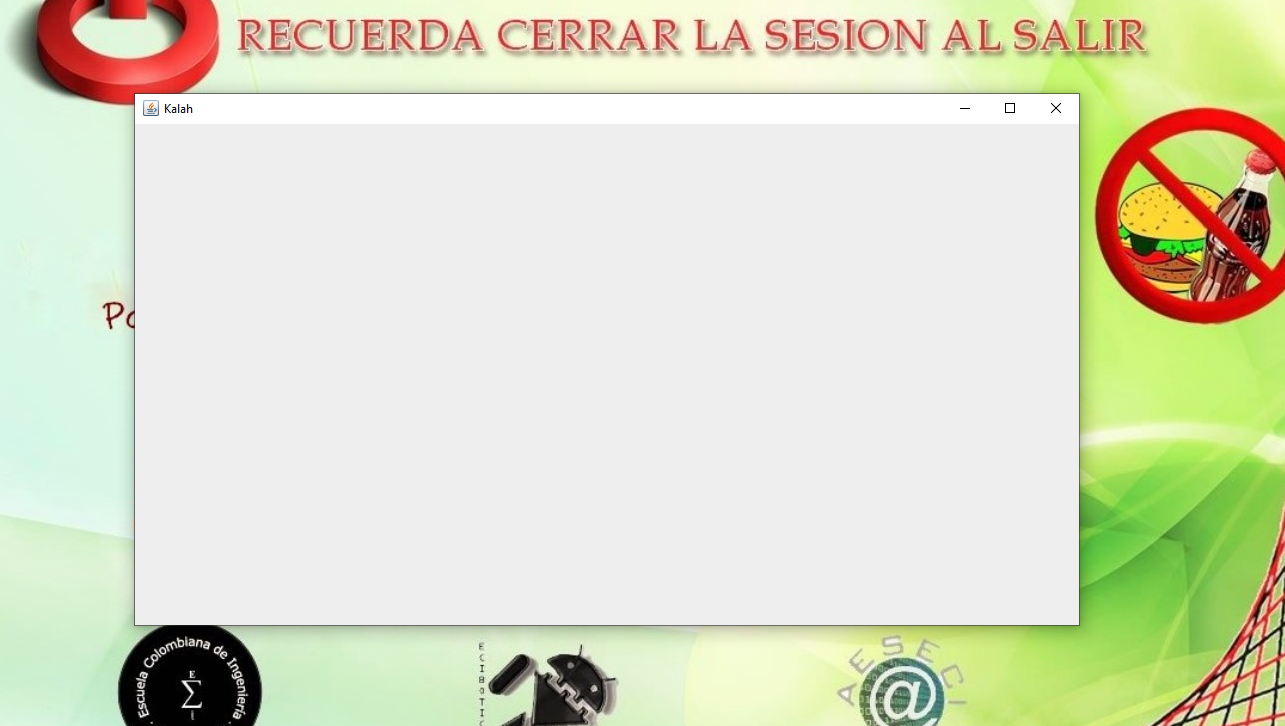
El objetivo es implementar la ventana principal de Kalah con un final adecuado a la aplicación desde el icono de cerrar (pidiendo confirmación a usuario). Utilizar el esquema de prepareElements-prepareActions

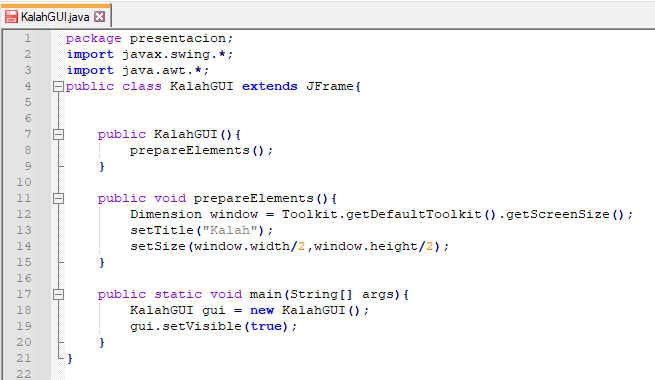
1. Construyan el primer esquema de la ventana de Kalah únicamente con el título “Kalah”. Para esto implemente la clase KalahGUI como un JFrame con su creador (que sólo coloca el título) y el método main que crea un objeto KalahGUI y lo hace visible. Ejecútenlo. Capturen la pantalla.





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





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

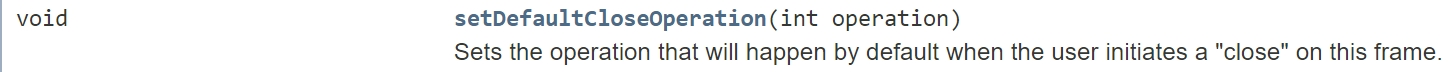
La ejecución no termina, se usa el comando Ctrl + C.



Tiene que ver con el hecho de que Java es un lenguaje de subprocesos múltiples; es decir, la JVM puede iniciar múltiples procesos o subprocesos simultáneos, según sea necesario. Resulta que cada vez que iniciamos una GUI, la JVM genera dos subprocesos: además del subproceso principal, que se inicia para cualquier programa, un segundo subproceso, el subproceso de despacho de eventos, lanzado que está dedicado a manejar eventos GUI. El subproceso principal de una aplicación Java termina automáticamente cuando se alcanza la llave de cierre del método principal; mientras el subproceso de la GUI siga ejecutándose, la JVM nose apaga automáticamente. Y, el subproceso de la GUI continuará ejecutándose siempre que haya al menos un componente viable de la GUI aún activo.

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

Consultando en el API de la clase JFrame se aprecia que este método funciona para que (nosotros como desarrolladores) seamos capaces de crear una acción por defecto para cerrar la ventana trabajada



En este caso lo usaríamos para poder cerrar nuestra ventana de prueba sin tener que ejecutar el comando CTRL + C desde consola

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

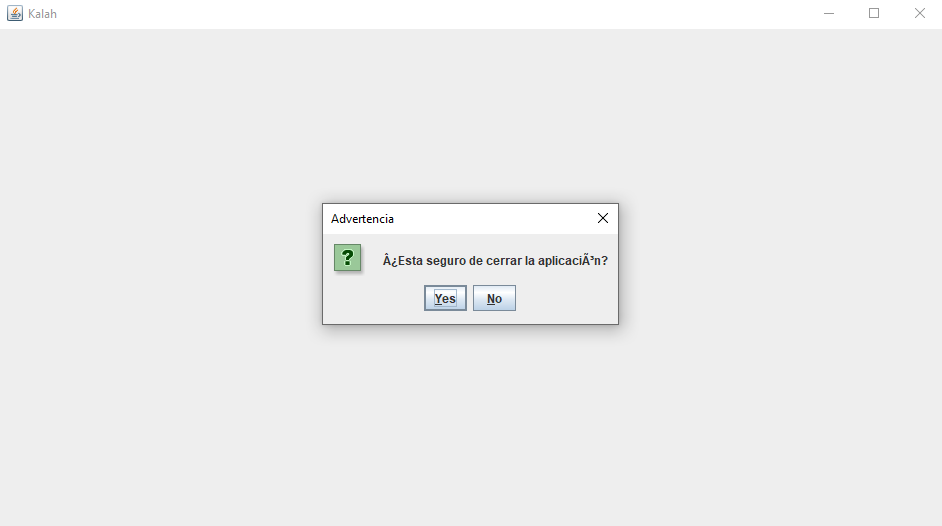
Se prepara el “oyente” y se codifica el método asociado “actionExit”

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



El programa termina la ejecución.



## 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. Defina como atributos los componentes visuales necesarios del menú.

Texto

Descripción generada automáticamente

1. Construyan la forma del menú propuesto (prepareElements - prepareElementsMenu) . Ejecuten. Capturen la pantalla

Se agregaron los elementos del menú como atributos de la clase, y se hace el llamado a este método desde prepareElements.

Imagen que contiene texto, tabla, computer, verde

Descripción generada automáticamenteImagen que contiene Forma

Descripción generada automáticamente

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

Word

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica

Descripción generada automáticamente

## Ciclo 2: Salvar y abrir

El objetivo es preparar la interfaz para las funciones de persistencia

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

JFileChooser: La Clase JFileChooser, nos presenta una ventana de herramientas, que, dependiendo de nuestra lógica, nos permitirá Abrir o Guardar archivos.

showOpenDialog: Crea una ventana para abrir un archivo.

showSaveDialog: Crea una ventana para guardar un archivo.

getSelectedFile: Retorna el archivo seleccionado.

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

Se implementaron los métodos encargados de las acciones:

, . Se preparan los “oyentes” correspondientes a los iconos Abrir y Salvar.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Ejecuten las dos opciones y capturen las pantallas más significativas.

**Salvar:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Abrir:**

Interfaz de usuario gráfica, Texto, Aplicación

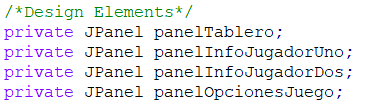
Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## Ciclo 3: Forma de la ventana principal

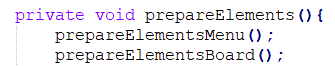
El objetivo es codificar el diseño de la ventana principal (todos los elementos de primer nivel)

1. Definan como atributos privados todos los componentes visuales necesarios.

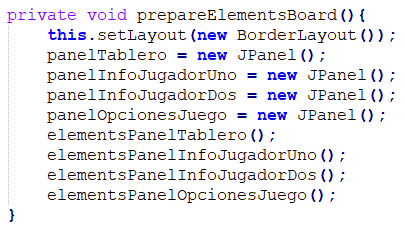


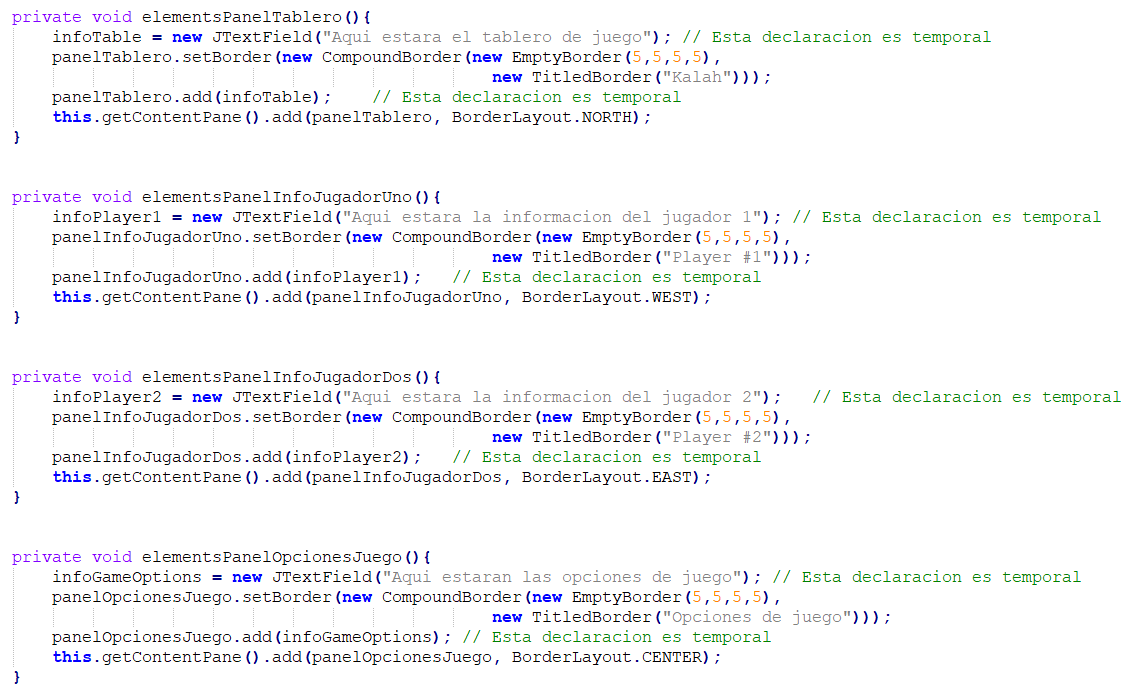
1. Continúe con la implementación del método prepareElements().

Para la zona del Board defina un método prepareElementsBoard() y un método refresh() que actualiza la vista del tablero considerando, por ahora, el tablero inicial por omisión (definido en el trabajo en clase) Este método lo vamos a implementar realmente en otros ciclos. Ejecuten y capturen esta pantalla.



Implementando el método prepareElementsBoard





Finalmente, en la interfaz gráfica se observan los cambios



## Ciclo 4: Cambiar color

El objetivo es implementar este caso de uso

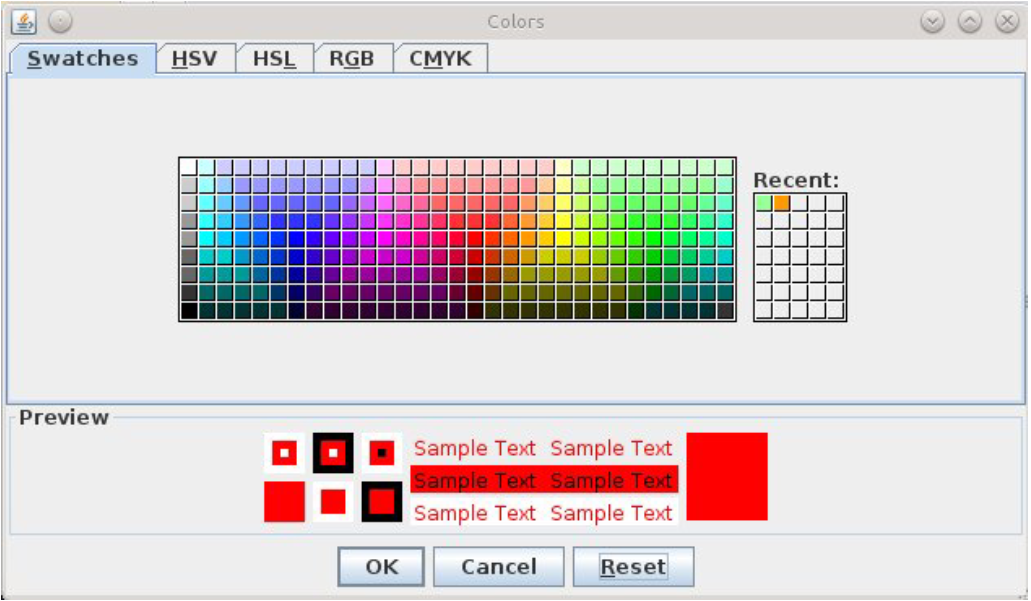
1. Expliquen los elementos (vista – controlador) necesarios para implementar este caso de uso.

Para la vista es necesario presentar al usuario una opción en la cual este pueda hacer llamado de una acción para escoger con libertad un color (a través una paleta de colores) de preferencia y mostrar dicha alteración reflejada en el tablero de juego.

En este caso nuestro controlador será la clase JColorChooser con la cual se podrá presentar al usuario una interfaz de selección de color alterando internamente el atributo de color que posee nuestra ventana.

1. Detallen el comportamiento de JColorChooser especialmente el método estático showDialog.

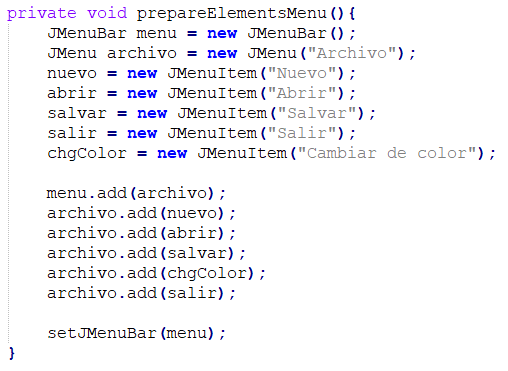
JColorChooser es una implementación del paquete *Swing* de java, esta le permite al desarrollador alterar aspectos gráficos de las interfaces graficas que implementa. También hace las veces de puente entre el usuario y el dominio del programa, ya que implementa una ventana integrada con una paleta de colores siendo usada para recoger información de preferencia sobre cambios que este desee hacer sobre algún aspecto de la ventana que este manipulando.



Finalmente, el método *ShowDialog* es usado por la clase JColorChooser para invocar la ventana que contiene la paleta de colores anteriormente mencionada.

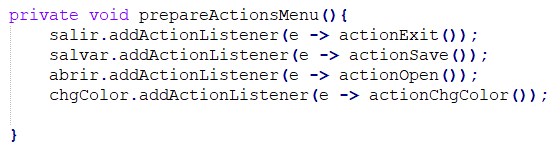
1. Implementen los componentes necesarios para cambiar los colores del juego.

Para nuestra implementación decidimos agregar un nuevo item a nuestro menú, nombrándolo “Cambiar color”.

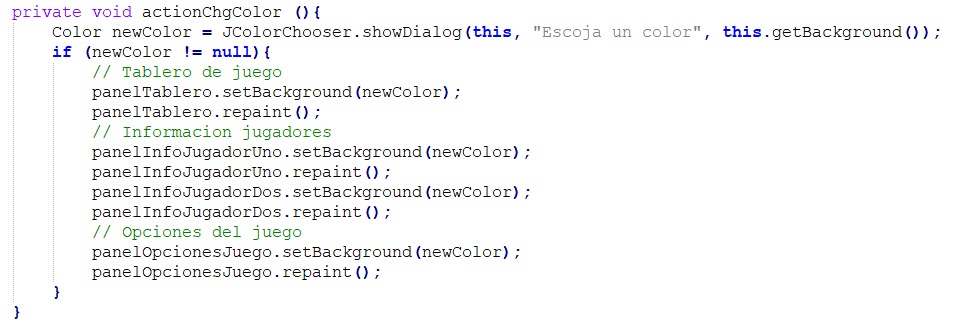




Se prepara el oyente de esta acción



Y por último se describen los cambios que debe hacer esta acción



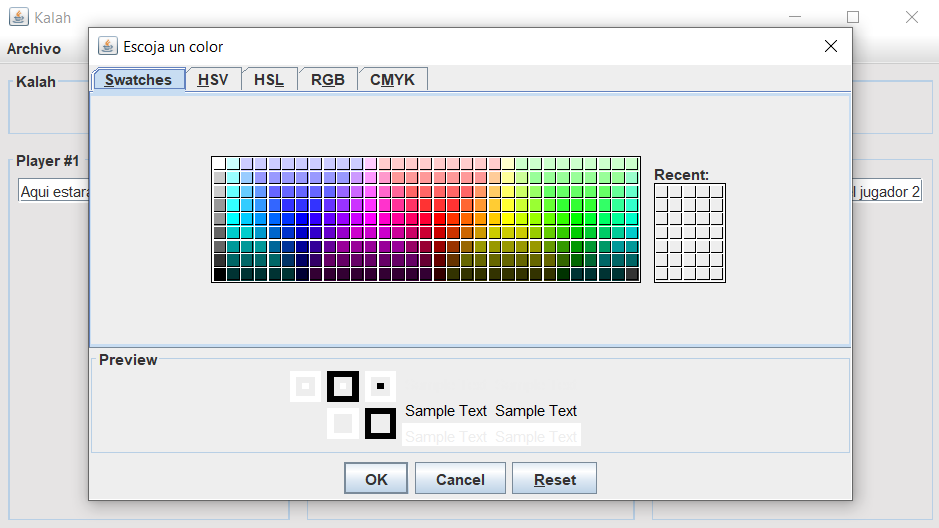
1. Ejecuten el caso de uso y capture las pantallas más significativas.

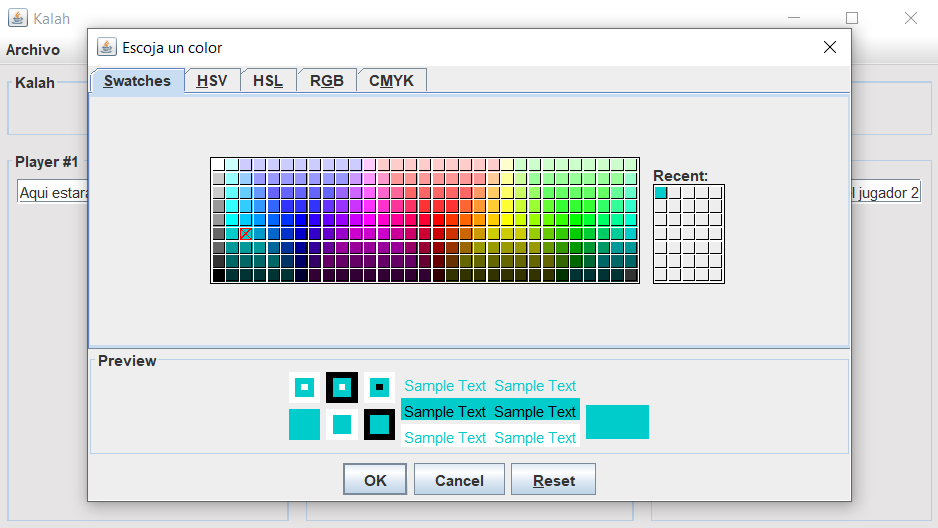
A continuación, se presenta un paso a paso desde nuestra GUI de cómo se ejecuta este nuevo método

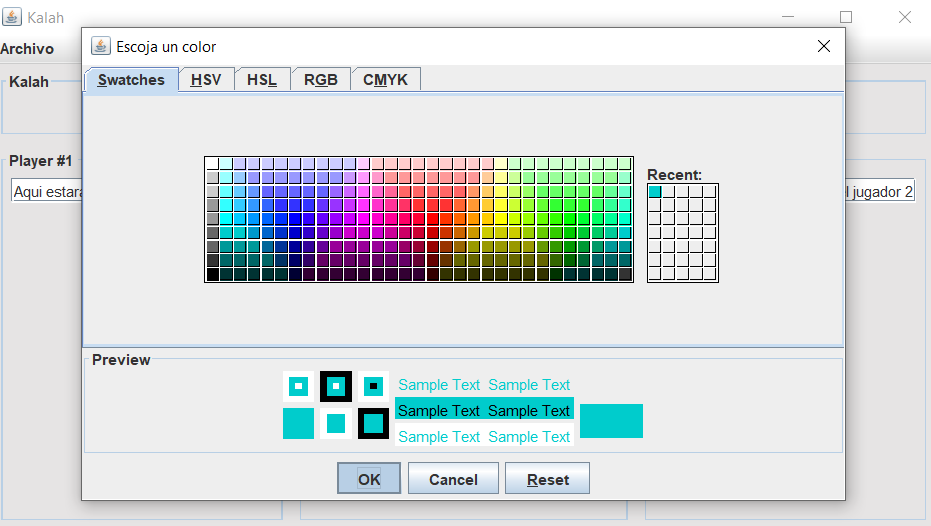


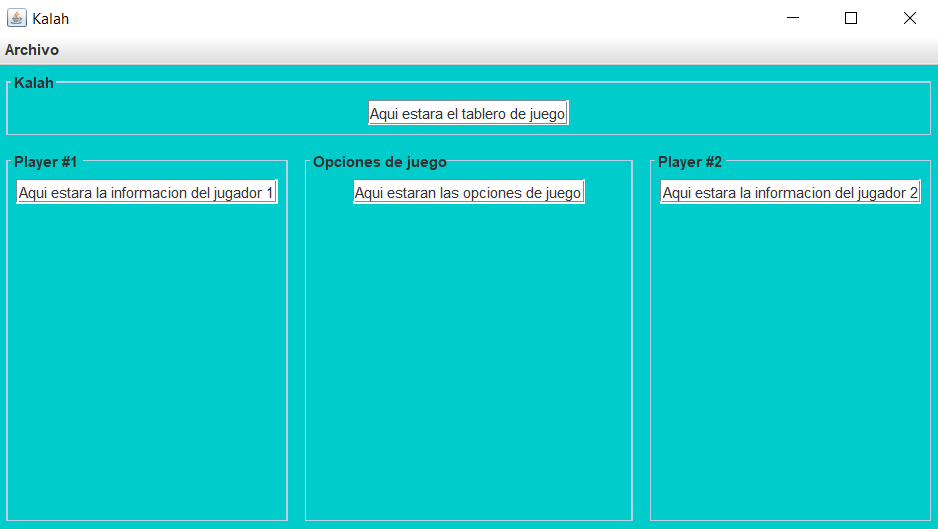












## Ciclo 5: Modelo Kalah

El objetivo es implementar la capa de dominio para Kalah

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

Se crea una clase Kalah que se encarga de la parte estructural de juego y una clase KalahGame en la cual se implementan los métodos básicos del juego:

Métodos básicos:

* Constructor del KalahGame: Crea un tablero con un numero de casas y semillas iniciales pasadas por parámetro.
* Play: Esparce las semillas de una determinada casa de un jugador de acuerdo con las reglas establecidas.
* Board: Se obtiene el estado de las casas en el juego.
* Player: Devuelve el jugador que tiene el turno actual.

1. Ejecuten las pruebas y capturen el resultado.

Pruebas de la estructura:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Pruebas de las dinámicas del juego.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

## Ciclo 6: Jugar

El objetivo es implementar la capa de dominio para Kalah

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

Dentro de KalahGUI se añade el atributo *game* el cual guarda un objeto tipo *KalahGame* haciendo las veces de operador de acciones, correcciones en la lógica y jugar.



## 

## Perfeccionen el método refresh() considerando la información del modelo de

## Dominio.

## Expliquen los elementos necesarios para implementar este caso de uso. Implementen los componentes necesarios para jugar .

## Texto Descripción generada automáticamente

## Primero se tiene un panel para el tablero de juego, este se encarga de almacenar las casa con la semillas y los almacenes de los jugadores.

## Texto Descripción generada automáticamente

## Dicho panel interactúa con el jugador al disponer botones en cada una de las casas para *Jugar* en dicha posición.

## Imagen que contiene electrónica, computer, computadora, teclado Descripción generada automáticamente

## Se utilizan dos paneles más que contienen la información de los jugadores, resumiendo la cantidad de semillas que poseen en sus almacenes y el turno actual de cada uno de ellos.

## Texto Descripción generada automáticamente

## Texto Descripción generada automáticamente

|  |  |
| --- | --- |
|  |  |

## Finalmente se tiene un panel que le permite a los usuarios decidir, en el transcurso del juego, si desean reiniciar o terminar el juego actual.

## Interfaz de usuario gráfica, Aplicación Descripción generada automáticamente

## Interfaz de usuario gráfica Descripción generada automáticamenteEjecuten 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

## Interfaz de usuario gráfica Descripción generada automáticamenteInterfaz de usuario gráfica Descripción generada automáticamenteInterfaz de usuario gráfica Descripción generada automáticamenteInterfaz de usuario gráfica Descripción generada automáticamente

## Ciclo 7: Reiniciar

El objetivo es implementar este caso de uso.

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

Bajo el diseño de la interface de la implementación del juego se tiene un panel especifico para las opciones del juego “Game Options”

Se agregará a este panel el botón “Restart” que ejecutará una acción adecuada al ser oprimido.

1. Implementen los elementos necesarios para reiniciar

Se agrega un botón como un atributo privado para reiniciar el juego:



Se codifica un método prepareGameOptionsActions y se prepara el “oyente” para este caso de uso.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este componente será visible para los usuarios en un panel con título Game Options.

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

Un juego en proceso:

Interfaz de usuario gráfica

Descripción generada automáticamente

Al después de presionar en “Restart”

Interfaz de usuario gráfica

Descripción generada automáticamente

## 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

Se agrega en el menú principal la opción “Configuración” que tendrá las opciones adecuadas para la configuración del juego:

* Cambiar número de casas.
* Cambiar número de semillas.

Se codifican los métodos correspondientes a las acciones de cada una de las opciones, estas deben estar preparadas para atender las excepciones que se puedan generar tras los datos ingresados por el usuario.

1. Implementen los elementos necesarios para cambiar la configuración del juego.

Se adicionan los elementos correspondientes a la nueva opción del menú:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Los métodos correspondientes a las acciones:

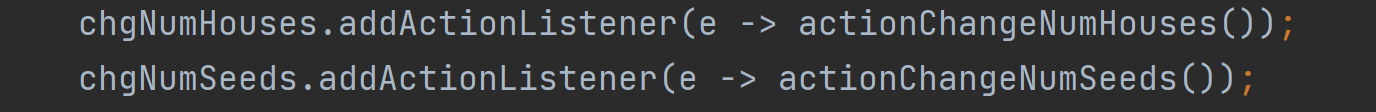
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Se preparan los oyentes:



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

Cambiar número de casas:

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Cambiar número de semillas:

Interfaz de usuario gráfica

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

## RETROSPECTIVA

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

(30) (Horas/ Hombre)

¿Cuál es el estado actual de laboratorio? ¿Por qué? (Para cada método incluya su estado)

El laboratorio se completó exitosamente.

Considerando las prácticas XP del laboratorio de hoy ¿por qué consideran que son importante?

ACCEPTANCE TESTS y WHEN [A BUG IS FOUND](http://www.extremeprogramming.org/rules/bugs.html) TESTS ARE CREATED, son importantes dado que permiten verificar un correcto comportamiento de la estructura y las funcionalidades del juego. Con las pruebas unitarias se comprobó que la parte lógica funcionara correctamente y con las de aceptación que al usuario se le presenta de forma adecuada el juego.

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

Implementar adecuadamente las actualizaciones graficas de la lógica detrás del juego. Escalar correctamente la relación de las imágenes presentadas en la interfaz para evitar sobre carga.

¿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.