Esteban Camilo Archila
Sergio Andres Otero

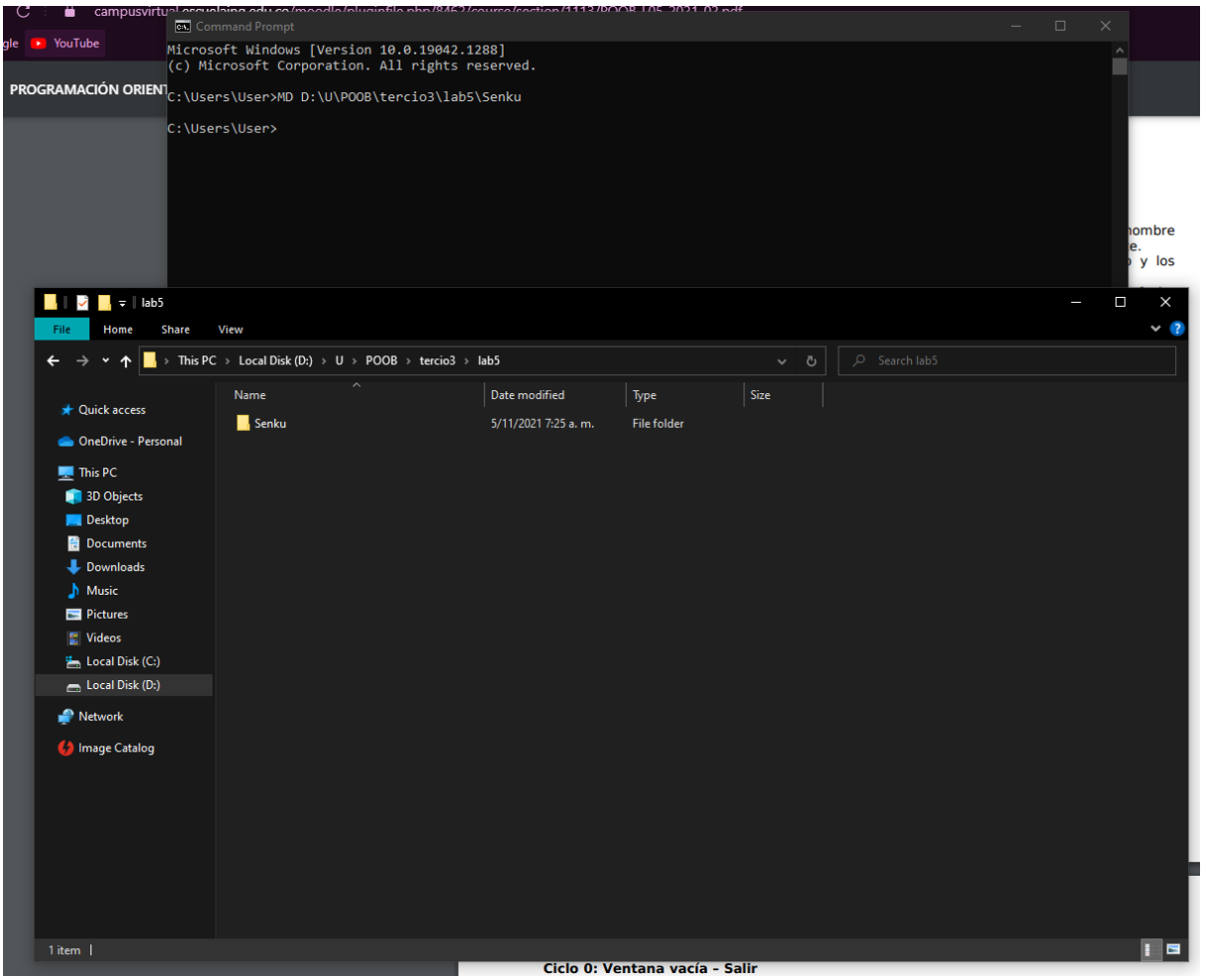
# Lab 5 - Interfaz

## DESARROLLO

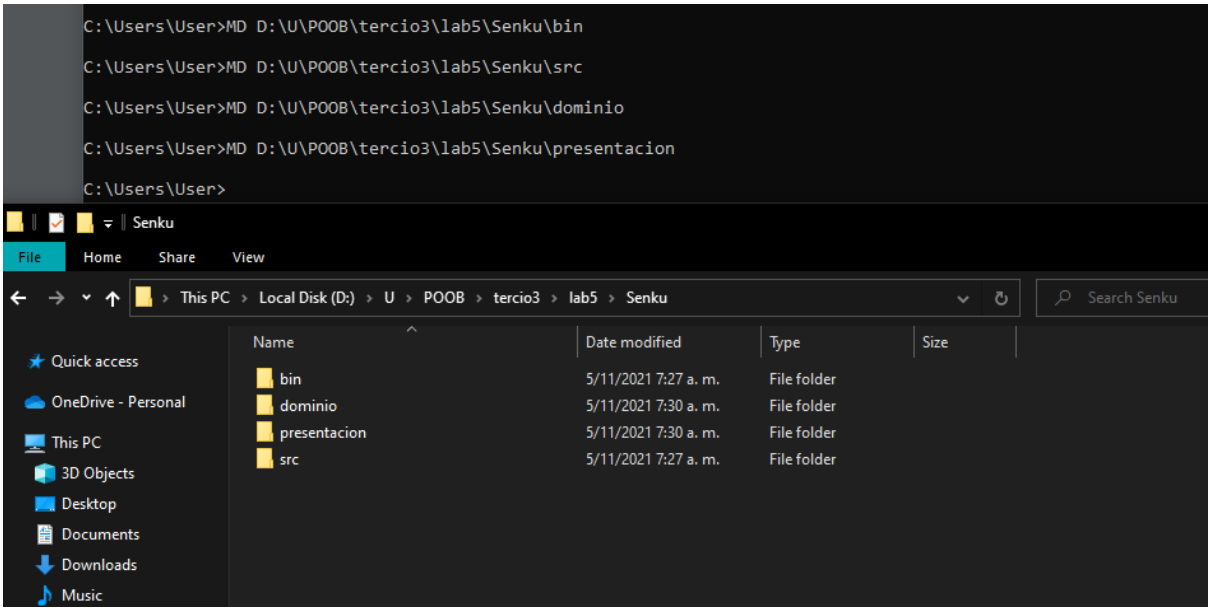
### Directorios

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

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




**Ciclo 0: Ventana vacía - Salir**

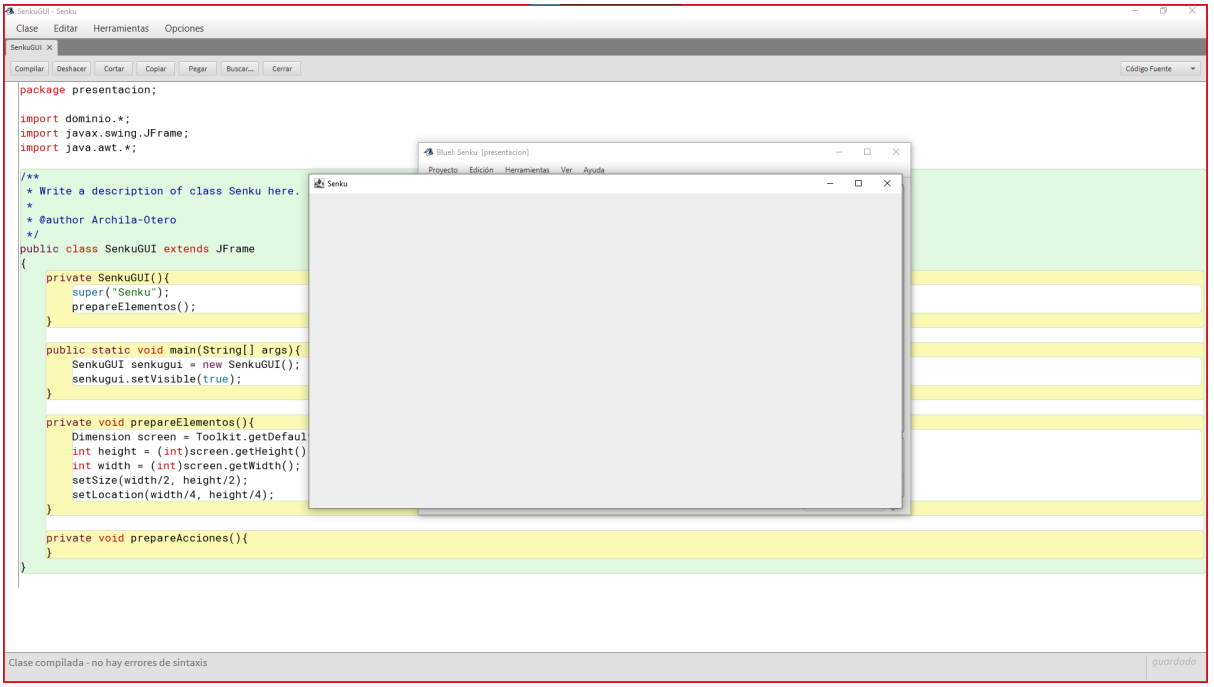



**Ciclo 0: Ventana vacía – Salir**

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

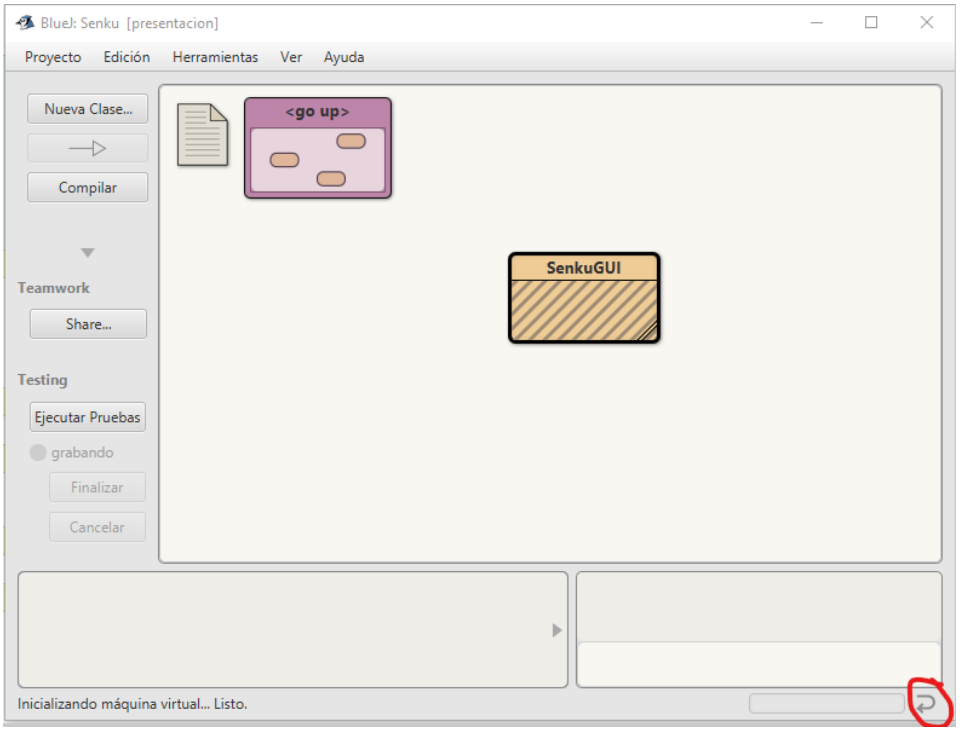
```java
package presentacion;

import dominio.*;
import javax.swing.JFrame;
import java.awt.*;

/**
 * Write a description of class Senku here.
 *
 * @author Archila-Otero
 */
public class SenkuGUI extends JFrame
{
    private SenkuGUI(){
        super("Senku");
        prepareElementos();
        prepareElementos();
    }

    public static void main(String[] args){
        SenkuGUI senkugui = new SenkuGUI();
        senkugui.setVisible(true);
    }
```

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




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

para terminar la ejecución debemos darle la siguiente flecha, por que en el codigo no le decimos cuando terminar



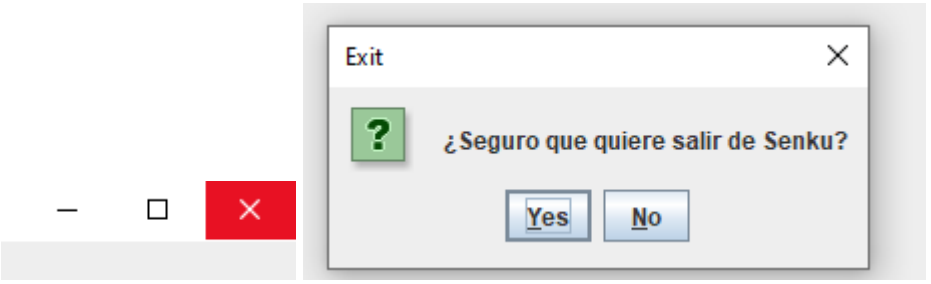
4. Estudien en JFrame el método setDefaultCloseOperation. ¿Para qué sirve? ¿Cómo lo usarían en este caso?

sirve para que el programa termine cuando se cierra el jframe

```
private SenkuGUI(){
    super("Senku");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    prepareElementos();
    prepareElementos();
}
```

5. 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 cierren el programa. Capturen las pantallas.

```
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent w) {
        salga();
    }
});
```



**Ciclo 1: Ventana con menú – Salir**

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

```
private JMenuBar menuBar = new JMenuBar();
private JMenu menu = new JMenu("Menu");
private JMenuItem menuItem1 = new JMenuItem("Nuevo");
private JMenuItem menuItem2 = new JMenuItem("Abrir");
private JMenuItem menuItem3 = new JMenuItem("Salvar");
private JMenuItem menuItem4 = new JMenuItem("Salir");
```

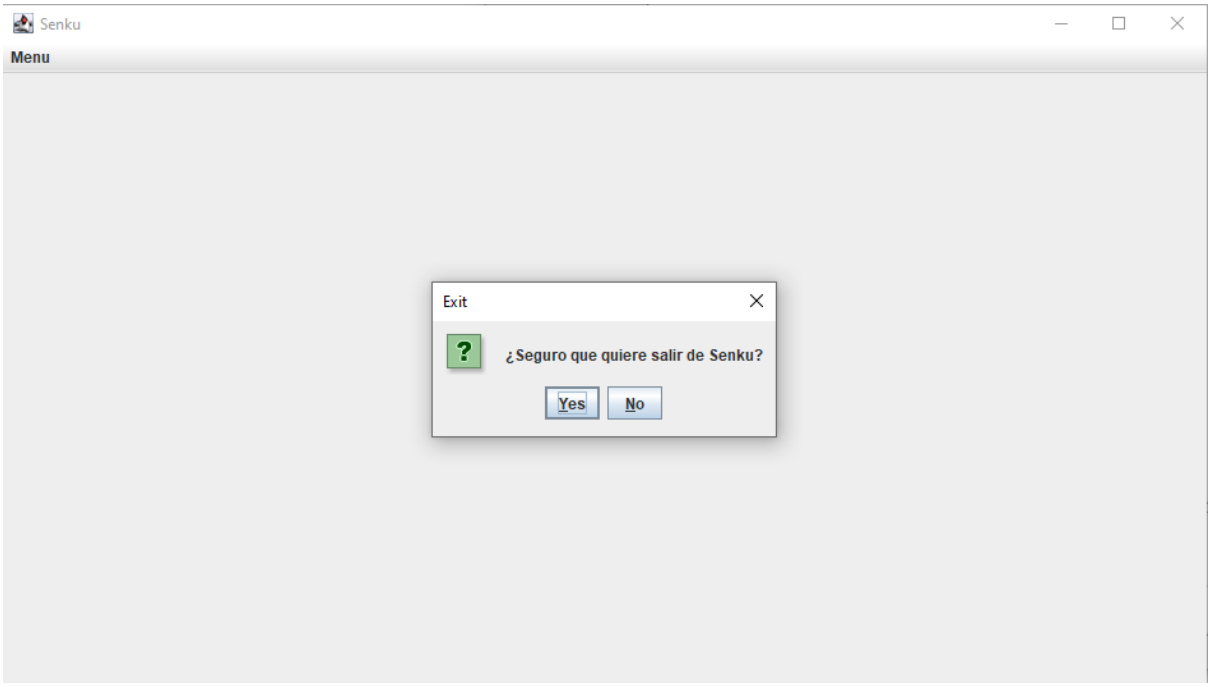
2. Construya la forma del menú propuesto (prepareElementos - prepareElementosMenu) . Ejecuten. Capturen la pantalla.

```
private void prepareElementos(){
    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
    int height = (int)screen.getHeight();
    int width = (int)screen.getWidth();
    setSize(width/2, height/2);
    setLocation(width/4, height/4);
    prepareElementosMenu();
}

private void prepareElementosMenu(){
    setJMenuBar(menuBar);
    menuBar.add(menu);
    menu.add(menuItem1);
    menu.add(menuItem2);
    menu.add(menuItem3);
    menu.add(menuItem4);
}
```

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

```
salir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        salga();
    }
});
```

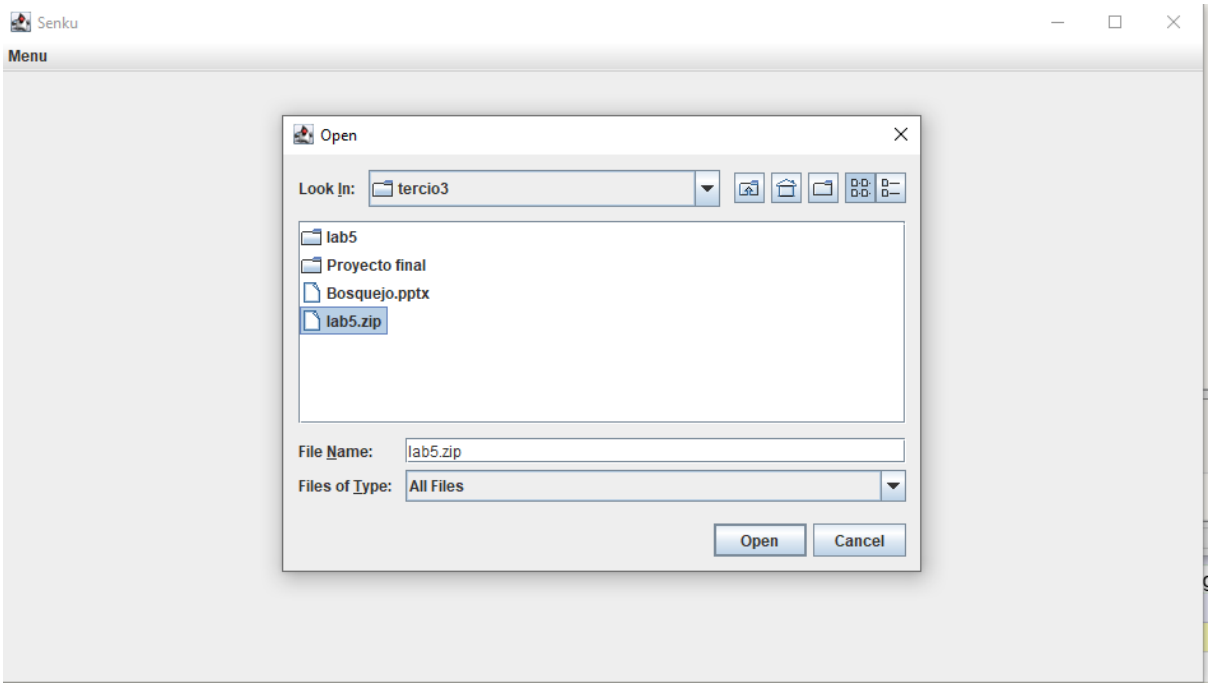

Senku
Menu
Nuevo
Abrir
Salvar
Salir

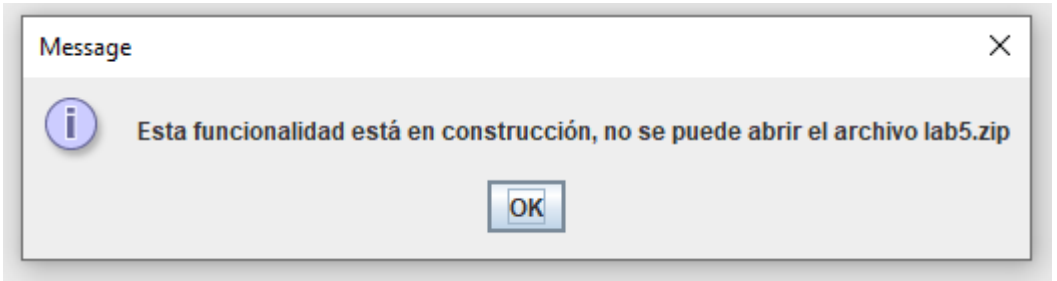

Senku
Menu
Exit
¿Seguro que quiere salir de Senku?
Yes
No

**Ciclo 2: Salvar y abrir**

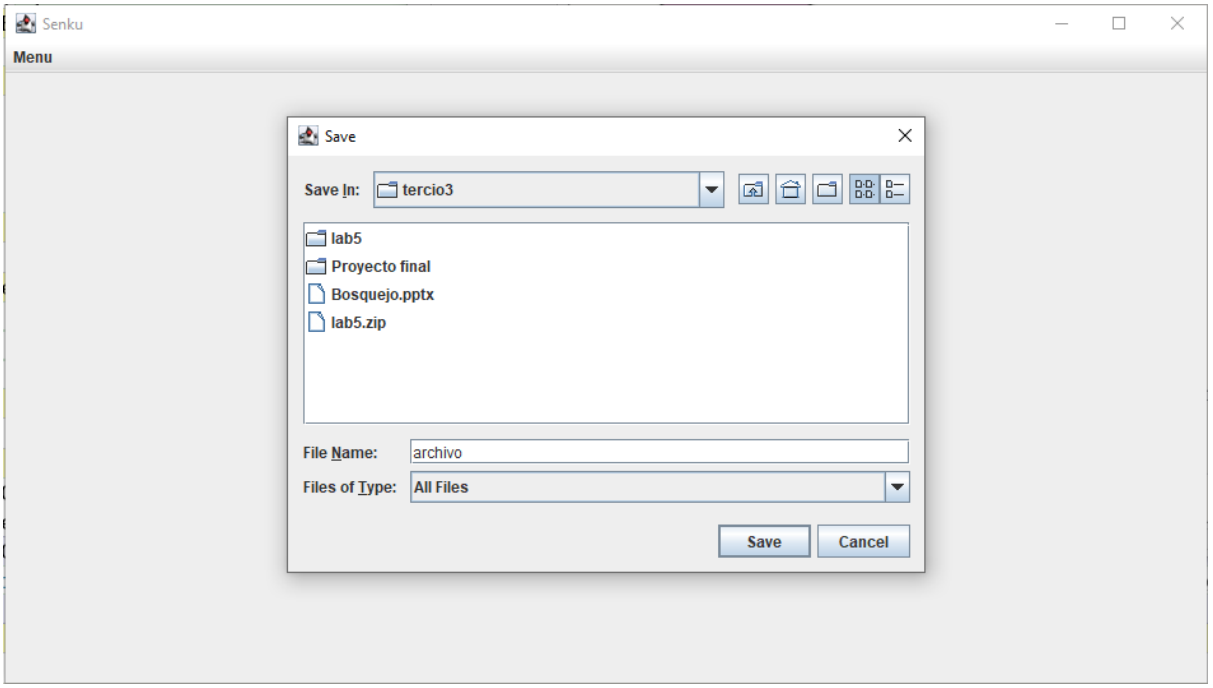
1. Detalle el componente JFileChooser especialmente los métodos : JFileChooser, showOpenDialog, showSaveDialog, getSelectedFile.
JFileChooser: Abre el buscador de archivos.
showOpenDialog: Permite elegir un archivo existente.
showSaveDialog: Permite crear y guardar un nuevo archivo.
getSelectedFile: Devuelve un elemento tipo File con el fichero actual.

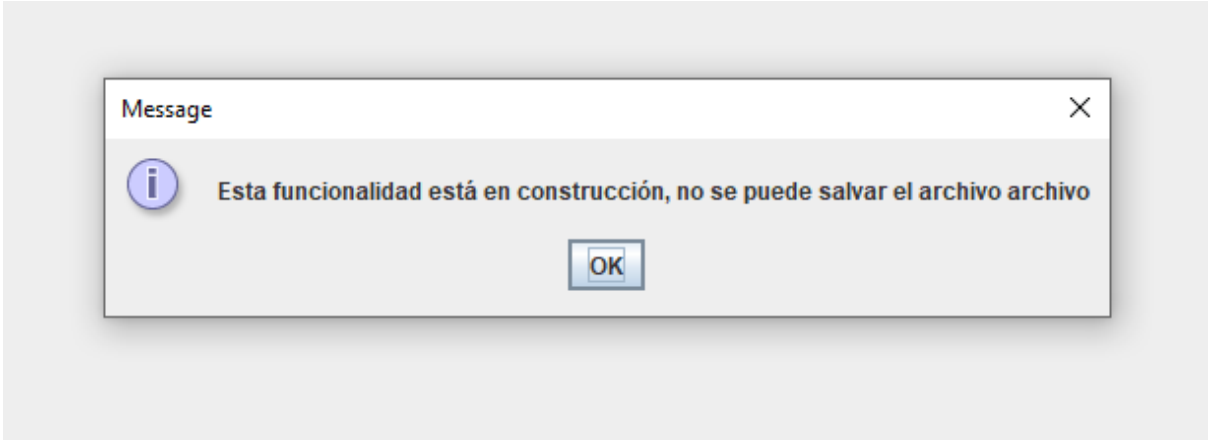
2. Implementen parcialmente los elementos necesarios para salvar y abrir. Al seleccionar los archivos indique que las funcionalidades están en construcción detallando la acción y el nombre del archivo seleccionado.
Al intentar abrir un archivo:



Al intentar salvar un archivo:




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

```
private void prepareAcciones(){
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent w) {
            salga();
        }
    });
    abrir.addActionListener(this);
    salvar.addActionListener(this);
    salir.addActionListener(this);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource()==abrir) abra();
    if (e.getSource()==salvar) salve();
    if (e.getSource()==salir) salga();
}

private void abra(){
    JFileChooser fileChooser = new JFileChooser();
    int returnValue = fileChooser.showOpenDialog(null);
    if (returnValue == JFileChooser.APPROVE_OPTION){
        JOptionPane.showMessageDialog(this, "Esta funcionalidad está en construcción, no se puede abrir el archivo " + fileChooser.getSelectedFile().getName());
    }
}

private void salve(){
    JFileChooser fileChooser = new JFileChooser();
    int returnValue = fileChooser.showSaveDialog(null);
    if (returnValue == JFileChooser.APPROVE_OPTION){
        JOptionPane.showMessageDialog(this, "Esta funcionalidad está en construcción, no se puede salvar el archivo " + fileChooser.getSelectedFile().getName());
    }
}
```

El resto de pantallazos fueron presentados en el anterior punto.

## Ciclo 3: Forma de la ventana principal

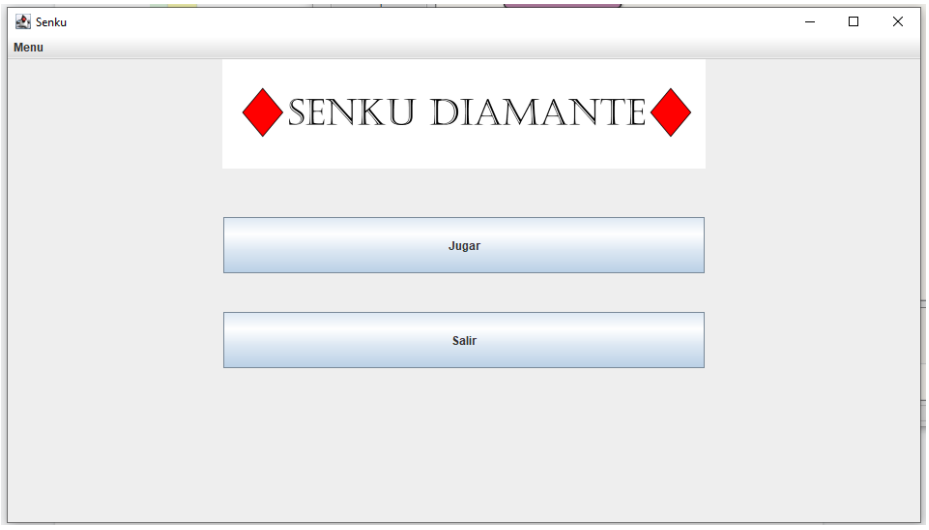
1. Definan como atributos privados todos los componentes visuales necesarios.

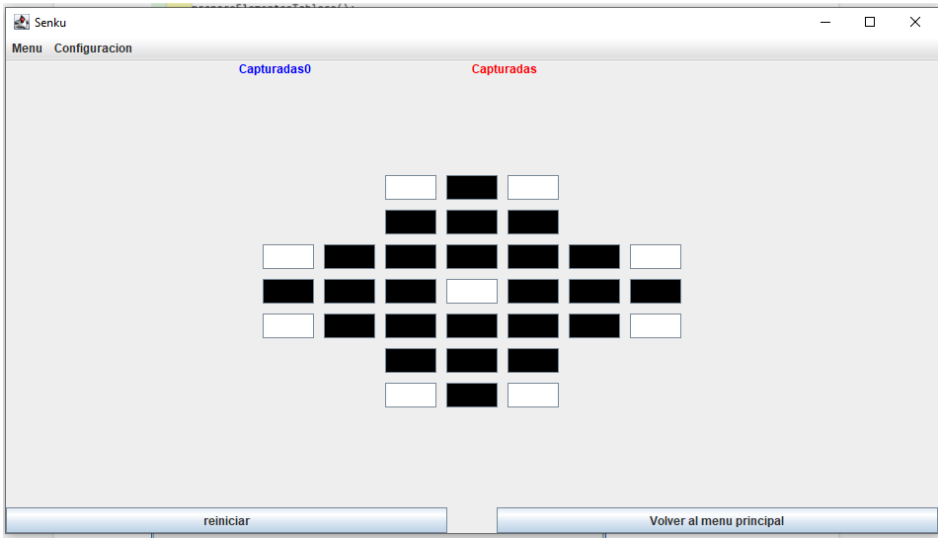
```
private void prepareElementosTablero(){
    tablero = new JPanel();
    tablero.setLayout(new BorderLayout(250, 100));
    JPanel marcadores = new JPanel();
    marcadores.setLayout(new GridLayout(1, 4));
    JPanel blank1 = new JPanel();
    JPanel blank2 = new JPanel();
    marcador1 = new JLabel("Movimientos");
    marcador2 = new JLabel("Capturadas");
    marcador1.setText("Movimientos");
    marcador2.setText("Capturadas");
    marcador1.setForeground(Color.blue);
    marcador2.setForeground(Color.red);
    marcadores.add(blank1);
    marcadores.add(marcador1);
    marcadores.add(marcador2);
    marcadores.add(blank2);
    tablero.add(marcadores, BorderLayout.NORTH);
    JPanel derecha = new JPanel();
    tablero.add(derecha, BorderLayout.EAST);
    JPanel izquierda = new JPanel();
    tablero.add(izquierda, BorderLayout.WEST);
    JPanel abajo = new JPanel();
    abajo.setLayout(new GridLayout(1, 2, 50, 50));
    refrescar = new JButton("Refrescar");
    abajo.add(refrescar);
    volverInicio = new JButton("Volver al menu principal");
    abajo.add(volverInicio);
    tablero.add(abajo, BorderLayout.SOUTH);
    createTablero();
}
```

```
private void createTablero(){
    tablero.setVisible(true);
    int tamano = 7;
    JPanel botonesTablero = new JPanel();
    botonesTablero.setLayout(new GridLayout(tamano, tamano, 10 ,10));
    botones = new JButton[tamano][tamano];
    for (int i = 0 ; i < tamano; i++){
        for (int j = 0 ; j < tamano; j++){
            JButton boton = new JButton();
            boton.setBackground(Color.gray);
            botones[i][j] = boton;
            botonesTablero.add(boton);
        }
    }
    tablero.add(botonesTablero, BorderLayout.CENTER);
    iniciarTablero(tamano);
}

private void iniciarTablero(int tamano){
    int cont = 0;
    int mitad = (tamano/2);
    for(int i = 0;i<mitad+1;i++){
        if (cont == 0){
            botones[i][mitad+1].setBackground(Color.white);
            botones[i][mitad-1].setBackground(Color.white);
            botones[botones.length-i-1][mitad+1].setBackground(Color.white);
            botones[botones.length-i-1][mitad-1].setBackground(Color.white);
            botones[mitad+1][i].setBackground(Color.white);
            botones[mitad-1][i].setBackground(Color.white);
            botones[mitad+1][botones.length-i-1].setBackground(Color.white);
            botones[mitad-1][botones.length-i-1].setBackground(Color.white);
            botones[mitad][mitad].setBackground(Color.white);
        }
        if (cont != mitad){
            botones[i][mitad].setBackground(Color.black);
            botones[botones.length-i-1][mitad].setBackground(Color.black);
        }
        for(int k = 1;k<cont+1;k++){
```

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.

```
private void prepareElementos(){
    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
    this.height = (int)screen.getHeight();
    this.width = (int)screen.getWidth();
    setSize(width/2, height/2);
    setLocation(width/4, height/4);
    prepareElementosMenu();
    prepareElementosHome();
    prepareElementosTablero();
}
```



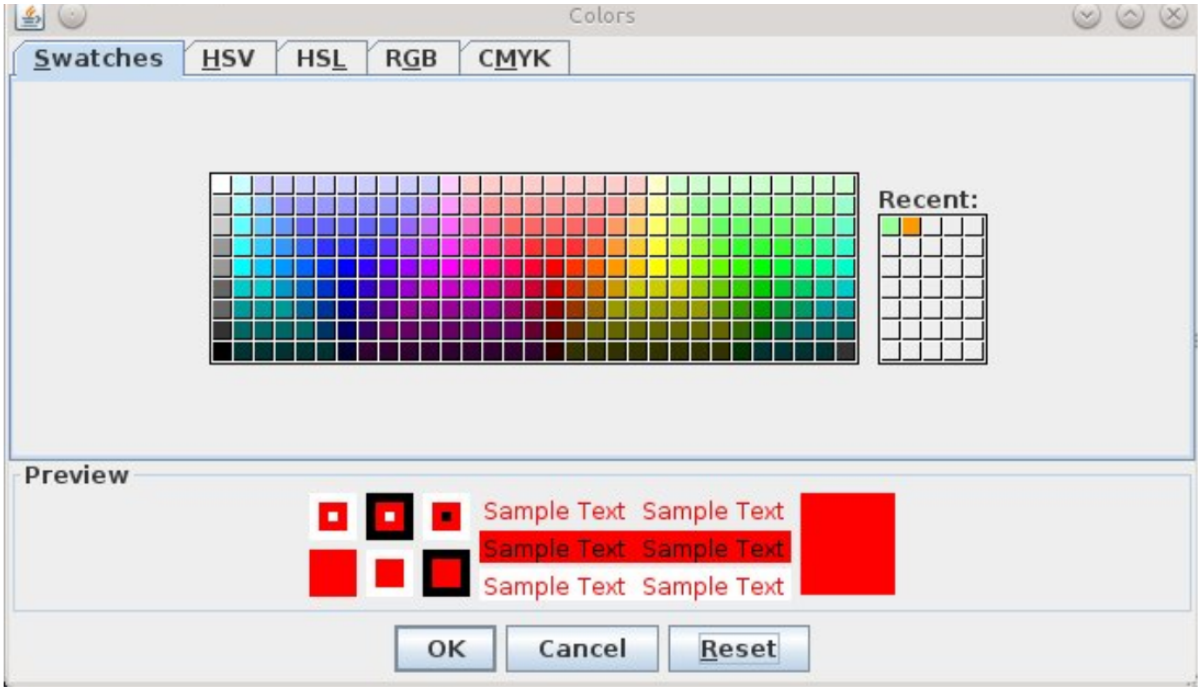




**Ciclo 4: Cambiar color**

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

Se debe crear un botón que nos lleve a una pantalla de control de colores (JColorChooser).

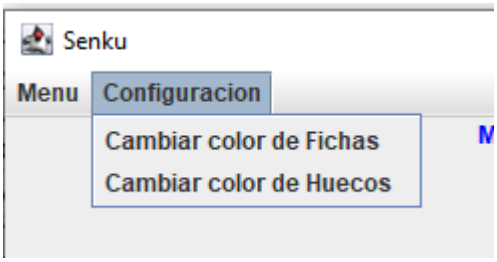
2. Detalle el comportamiento de JColorChooser especialmente el método estático showDialog
Este método abre una pantalla con diferentes tonalidades de colores para que el usuario elija

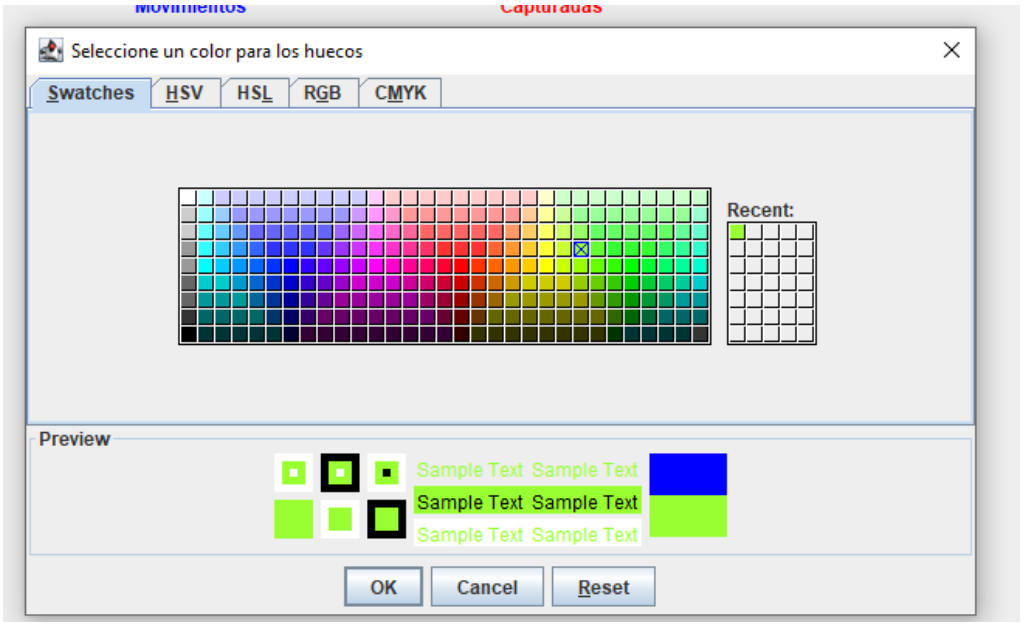


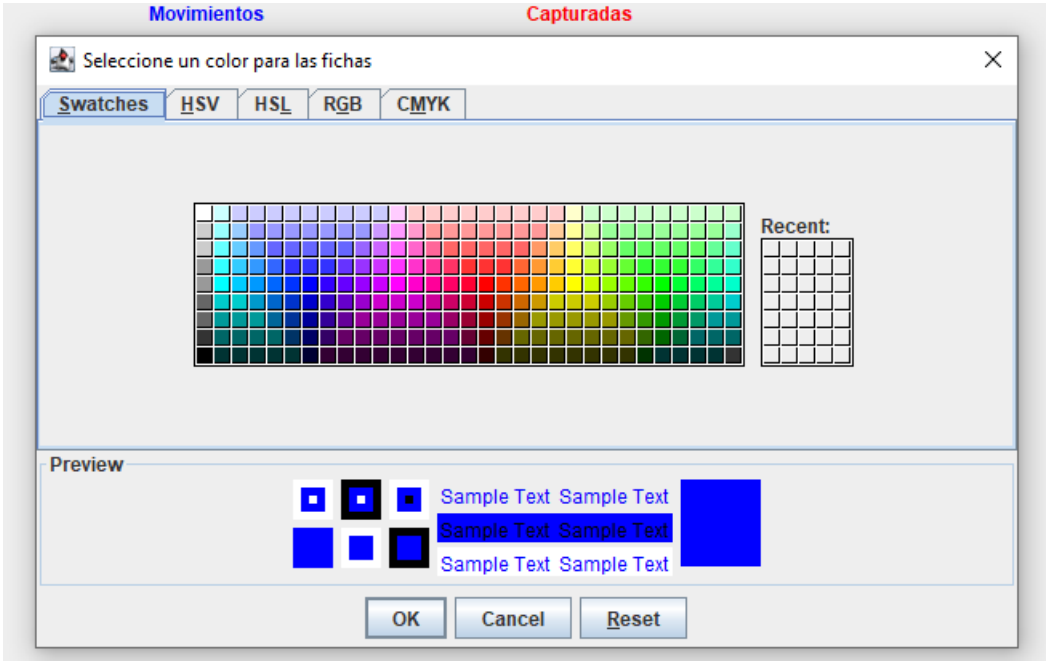

3. Implementen los componentes necesarios para cambiar el color del tablero (inicialmente blanco-negro)

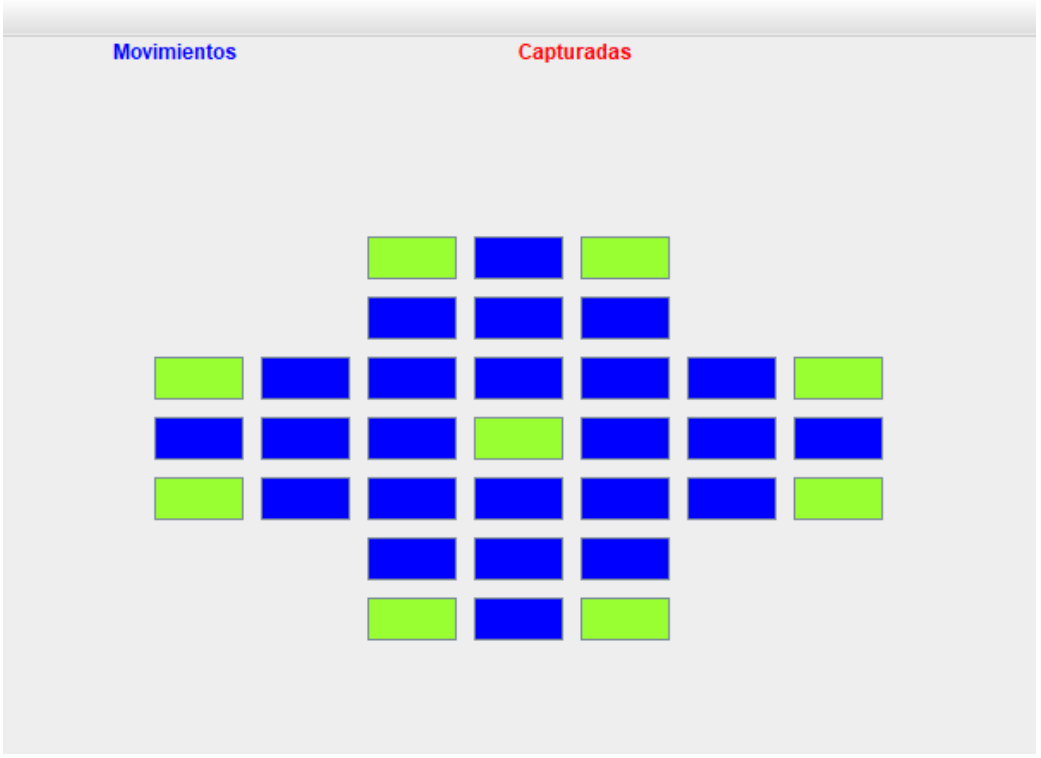
```
private void actualiceColores(Color oldColor, Color newColor){
    for (int i = 0; i < tamano; i ++){
        for (int j = 0; j < tamano; j ++){
            if (botones[i][j].getBackground() == oldColor){
                botones[i][j].setBackground(newColor);
            }
        }
    }
}

private void cambieColorFichas(){
    Color color = JColorChooser.showDialog(null, "Seleccione un Color", Color.BLUE);
    Color oldColor = colorFichas;
    this.colorFichas = color;
    actualiceColores(oldColor, colorFichas);
}

private void cambieColorHuecos(){
    Color color = JColorChooser.showDialog(null, "Seleccione un Color", Color.BLUE);
    Color oldColor = colorHuecos;
    this.colorHuecos = color;
    actualiceColores(oldColor, colorHuecos);
}
```

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


Senku
Menu
Configuracion
Cambiar color de Fichas
Cambiar color de Huecos


Seleccione un color para los huecos
Swatches
HSV
HSL
RGB
CMYK
Recent:
Preview
Sample Text Sample Text
Sample Text Sample Text
Sample Text Sample Text
OK
Cancel
Reset


Movimientos
Capturadas
Seleccione un color para las fichas
Swatches
HSV
HSL
RGB
CMYK
Recent:
Preview
Sample Text Sample Text
Sample Text Sample Text
Sample Text Sample Text
OK
Cancel
Reset

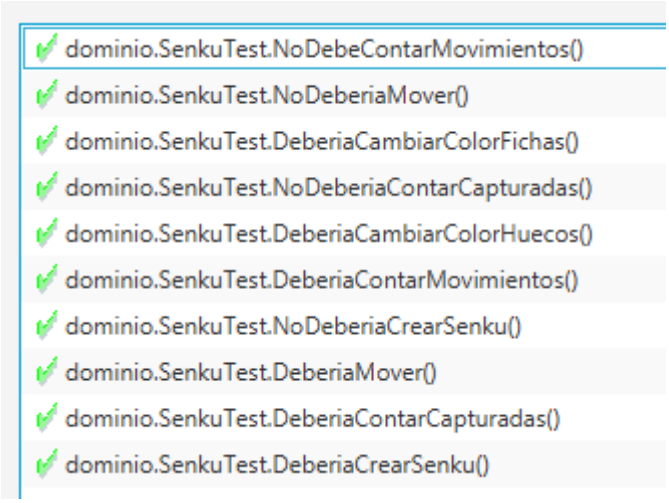



## Ciclo 5: Modelo Senku

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

```
public int[] verificarBotones(int inicioX, int inicioY, int objetivoX, int objetivoY){
    int res[] = new int[2];
    res[0] = -1;
    if ((botones[inicioX][inicioY] == colorFichas) && (botones[objetivoX][objetivoY] == colorHuecos)){
        if (inicioX == objetivoX + 2){
            res[0] = inicioX - 1;
            res[1] = inicioY;
        }
        else if (inicioX == objetivoX - 2){
            res[0] = inicioX + 1;
            res[1] = inicioY;
        }
        else if (inicioY == objetivoY + 2){
            res[0] = inicioX;
            res[1] = inicioY - 1;
        }
        else if (inicioY == objetivoY - 2){
            res[0] = inicioX;
            res[1] = inicioY + 1;
        }
    }
    return res;
}

public void move(int inicioX, int inicioY, int objetivoX, int objetivoY, int[] botonMedio){
    botones[inicioX][inicioY] = Color.white;
    botones[botonMedio[0]][botonMedio[1]] = Color.white;
    botones[objetivoX][objetivoY] = Color.black;
}
```

2. Ejecuten las pruebas y capturen el resultado.




**Ciclo 6: Jugar**

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

```
private void iniciarTablero(){
    senku = new Senku(tamano, colorHuecos, colorFichas);
```

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

```
private void refrescar(){
    marcador1.setText("Movimientos" + senku.movimientos());
    marcador2.setText("Capturadas" + senku.movimientos());
    Color[][] colorTab = senku.getTablero();
    for (int i = 0; i < tamano; i ++){
        for (int j = 0; j < tamano; j ++){
            if(colorTab[i][j] != null){
                botones[i][j].setBackground(colorTab[i][j]);
            }
        }
    }
}
```

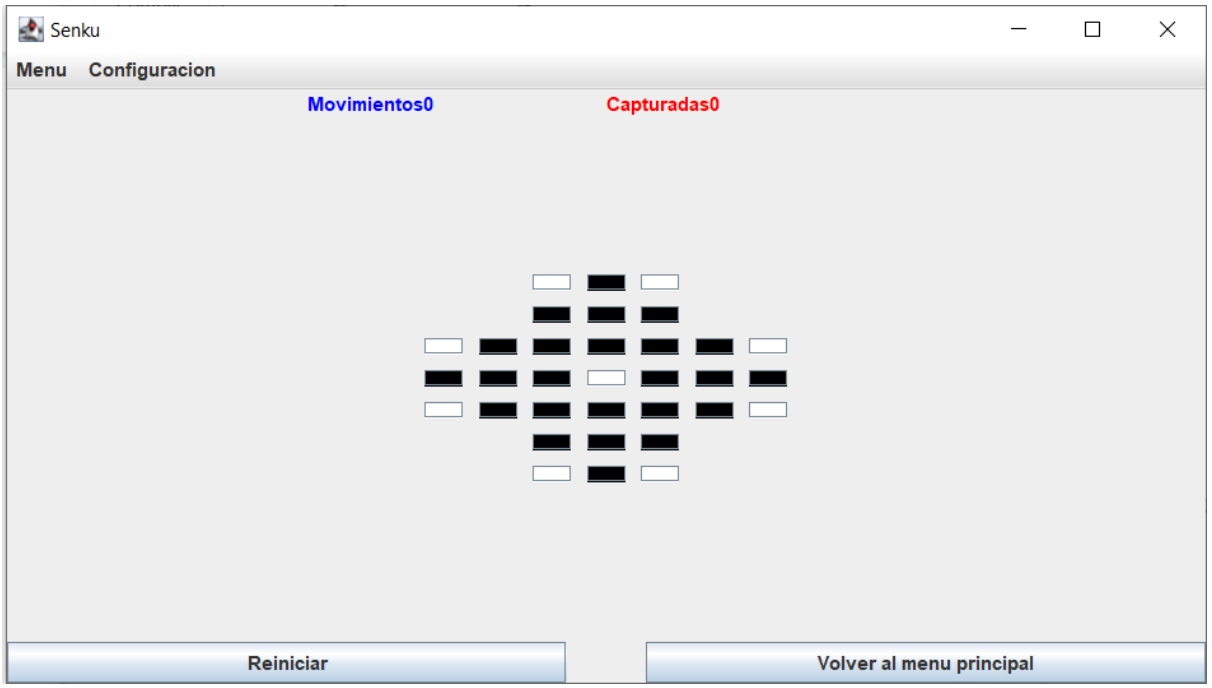
3. Expliquen los elementos necesarios para implementar este caso de uso.

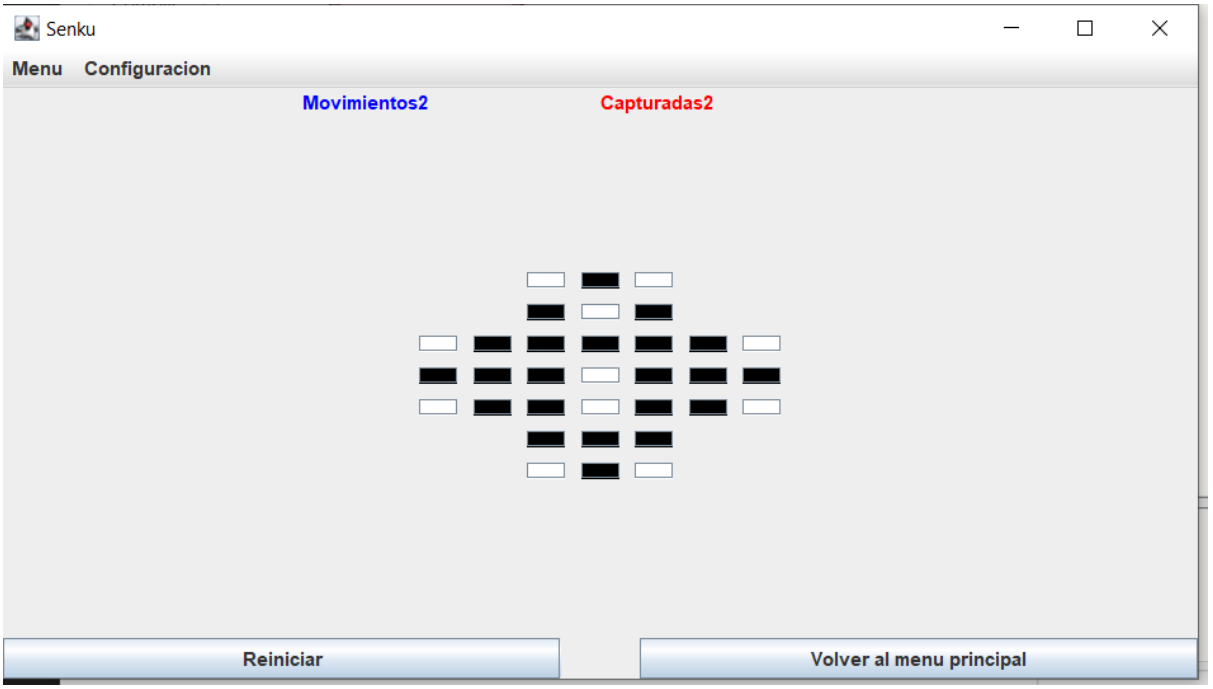
necesitamos en la capa de dominio la capa lógica del tablero, en presentación se actualiza el tablero de los botones

4. Implementen los componentes necesarios para jugar .

```
private void moverFicha(int i, int j){
    if (!inicialElegida){
        inicioX = i;
        inicioY = j;
        inicialElegida = true;
    }
    else if (!finalElegida){
        objetivoX = i;
        objetivoY = j;
        finalElegida = true;
        if (inicialElegida && finalElegida){
            int botonMedio[] = senku.verificarBotones(inicioX, inicioY, objetivoX, objetivoY);
            if (botonMedio[0] != -1){
                if (botones[botonMedio[0]][botonMedio[1]].getBackground() != Color.white){
                    senku.move(inicioX, inicioY, objetivoX, objetivoY, botonMedio);
                }
            }
        }
    }
    if (inicialElegida && finalElegida){
        finalElegida = false;
        inicialElegida = false;
        refrescar();
    }
}
```

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






Ciclo 7: Reiniciar

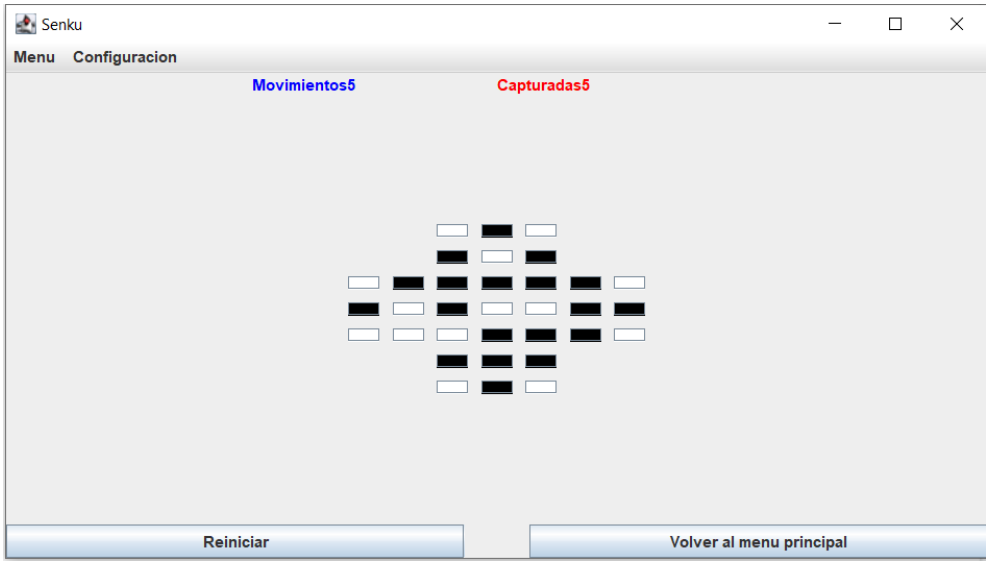
1. Expliquen los elementos a usar para implementar este caso de uso.

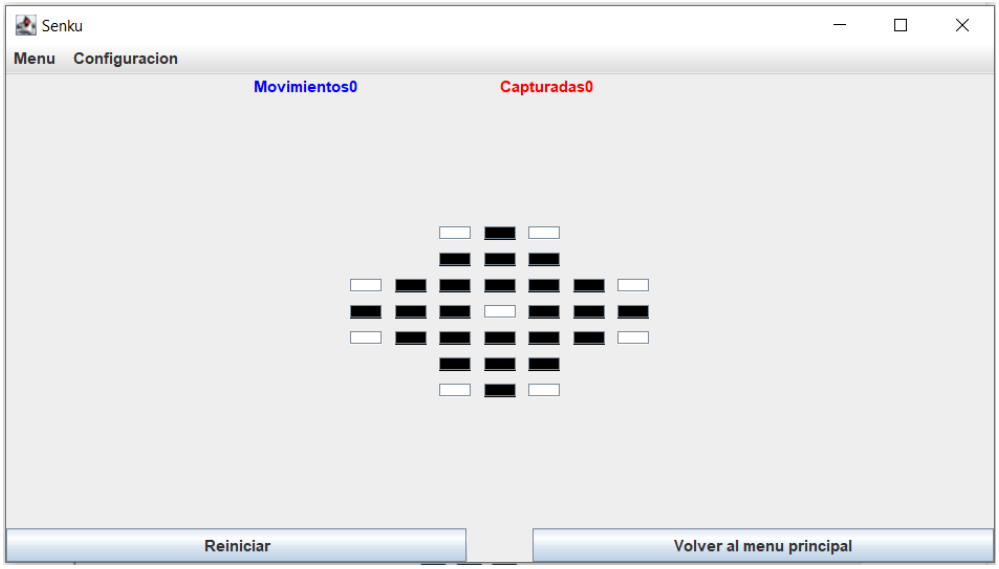
usamos el metodo de iniciarTablero y jugar

2. Implementen los elementos necesarios para reiniciar

```
private void reiniciar(){
    senku = new Senku(tamano, colorHuecos, colorFichas);
    iniciarTablero();
    jugar();
}
```

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




Ciclo 8: Cambiar la configuración del juego

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

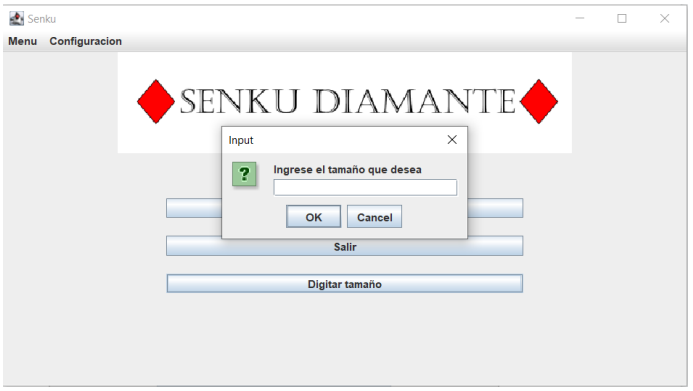
necesitamos un nuevo botón para acceder a un JOptionPane para ahi digitar el número, y que llame al método cambietamano

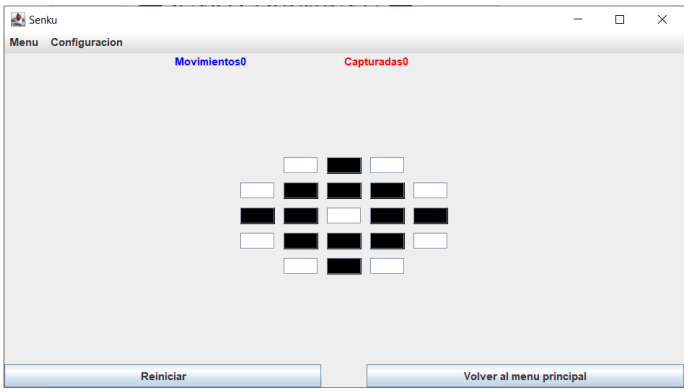
2. Implementen los elementos necesarios para cambiar la configuración del juego: tamaño

```
private void cambietamano(){
    this.tamano = Integer.parseInt(JOptionPane.showInputDialog("Ingrese el tamaño que desea"));
    this.botones = new JButton[tamano][tamano];
    createTablero();
    jugar();
}
```

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


Senku
Menu
Configuracion
SENKU DIAMANTE
Jugar
Salir
Digitar tamaño


Senku
Menu
Configuracion
SENKU DIAMANTE
Input
Ingrese el tamaño que desea
OK
Cancel
Salir
Digitar tamaño


Senku
Menu
Configuracion
Movimientos0
Capturadas0
Reiniciar
Volver al menu principal

RETROSPECTIVA

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

Archila: 12 horas

Otero: 15 horas

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

satisfactorio, se completó casi en su totalidad el laboratorio, por que surgio un problema en el último ciclo que no logramos solucionar

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

porque así el diseño y codificación quedan mucho más limpios y claros, tanto para nosotros, como para quien revise el programa

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

el mayor logro fue contruir con exito los 7 ciclos del laboratorio, el mayor problema fue el manejo de los movimientos del juego  lo resolvimos por medio de la parte logica de dichos movimientos

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

complementar las dudas del otro, realizar más pruebas