

UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO

LABORATORIO 5

PROGRAMACIÓN ORIENTADA A OBJETOS

NOMBRES:

CHRISTIAN ALFONSO ROMERO MARTÍNEZ
ANDERSON FABIAN GARCIA NIETO

PROFESORA:
MARIA IRMA DIAZ ROZO

10/04/25

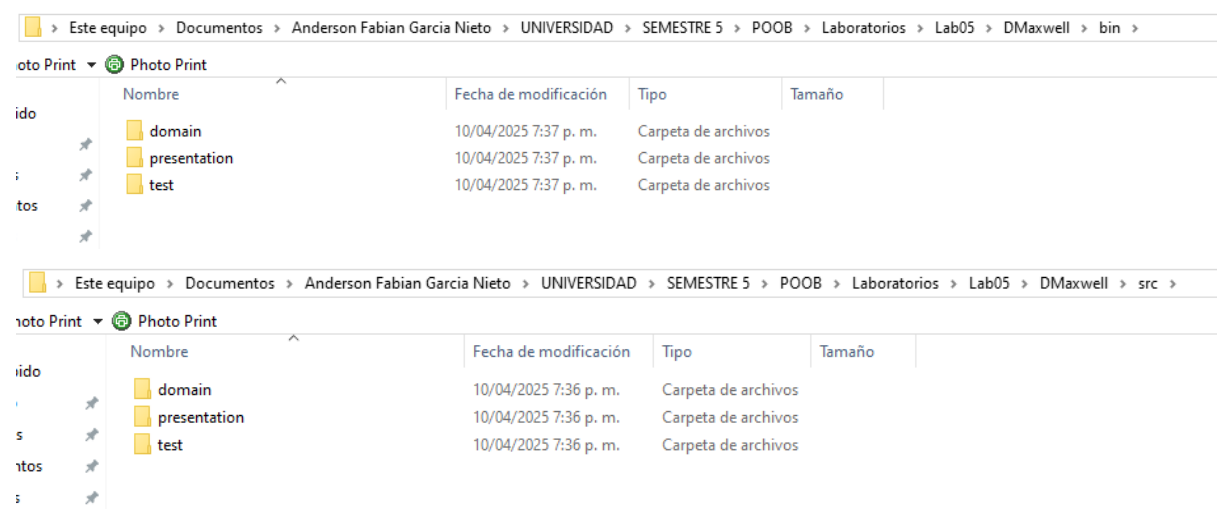
DESARROLLO

Directorios

El objetivo de este punto es construir un primer esquema para el juego DMaxwell. Preparen un directorio llamado DMaxwell con los directorios src y bin y los subdirectorios para presentación, dominio y pruebas de unidad. Capturen una pantalla con la estructura.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5737]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05>mkdir DMaxwell
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05>cd DMaxwell
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>mkdir src
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>mkdir bin
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>cd src
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\src>mkdir presentation
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\src>mkdir domain
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\src>mkdir test
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\src>cd
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\src>cd..
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>cd bin
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\bin>mkdir presentation
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\bin>mkdir domain
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\bin>mkdir test
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell\bin>
```



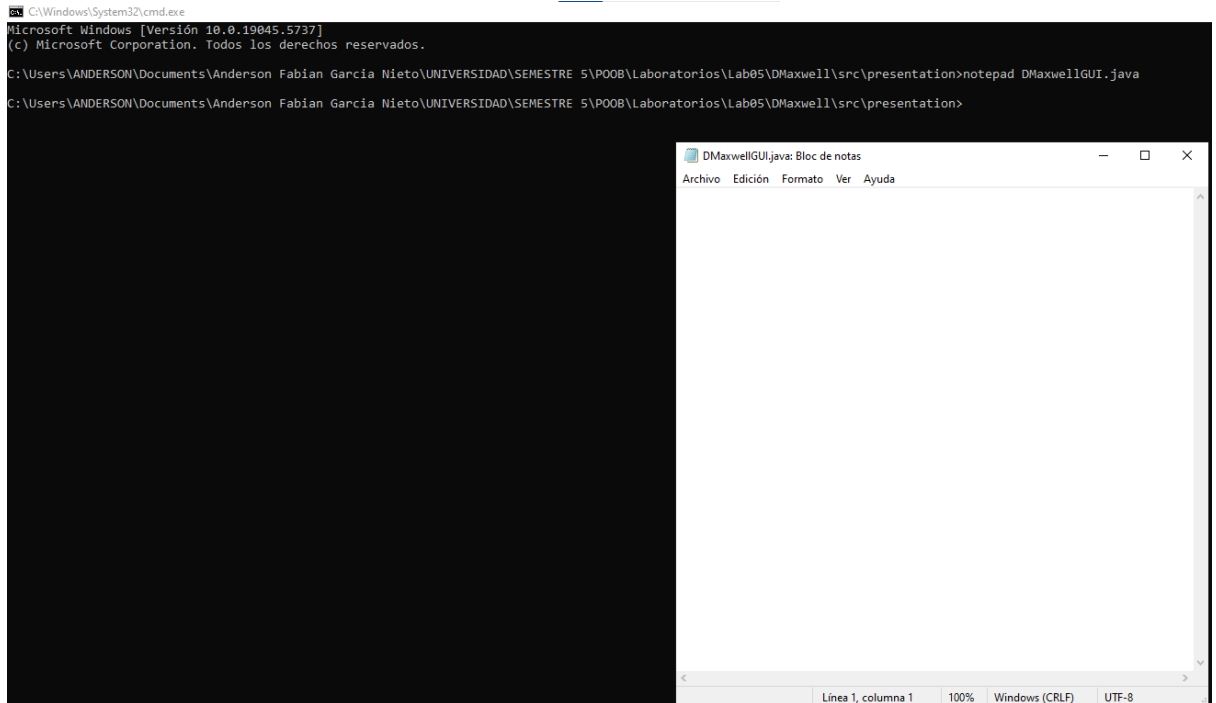
CICLO 0: VENTANA VACÍA-SALIR

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

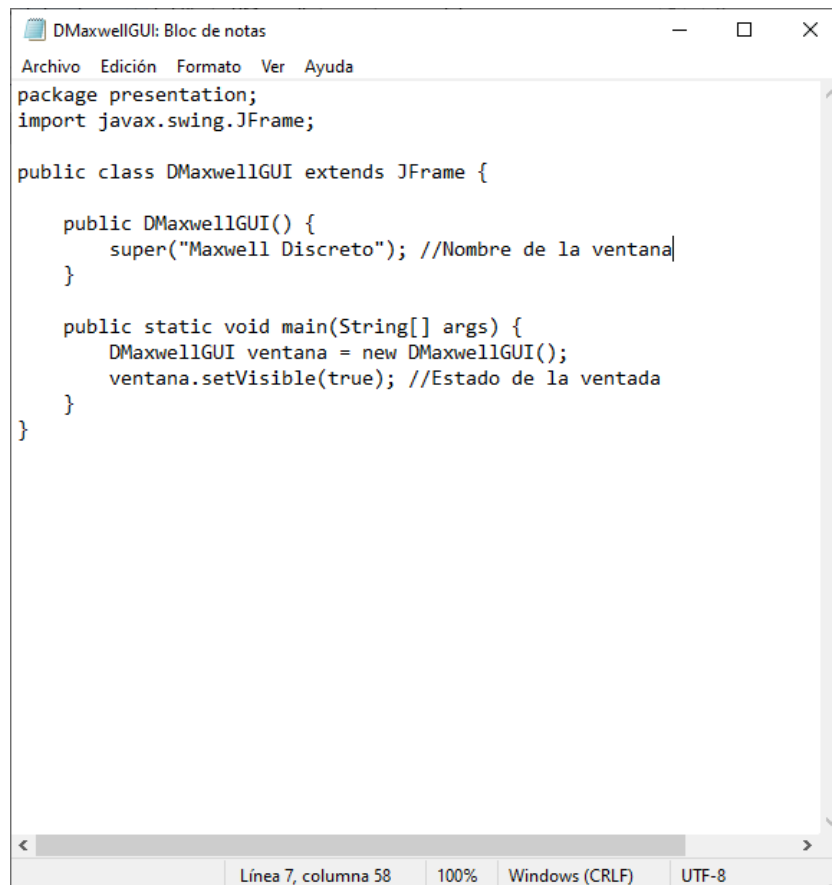
1. Construyan el primer esquema de la ventana de DMaxwell únicamente con el título “Maxwell Discreto”. Para esto cree la clase DMaxwellGUI como un JFrame con su creador (que sólo coloca el título) y el método main que crea un objeto DMaxwellGUI y lo hace visible. Ejecútenlo. Capturen la pantalla.

Para la creación del primer esquema se siguieron los siguientes pasos:

- a. En consola, escribir el siguiente comando dentro de la carpeta src/presentation para crear un archivo .java “**notepad DMaxwellGUI.java**”, y se crea un nuevo archivo.



- b. Dentro del archivo, se escribe el código de la clase y se guarda



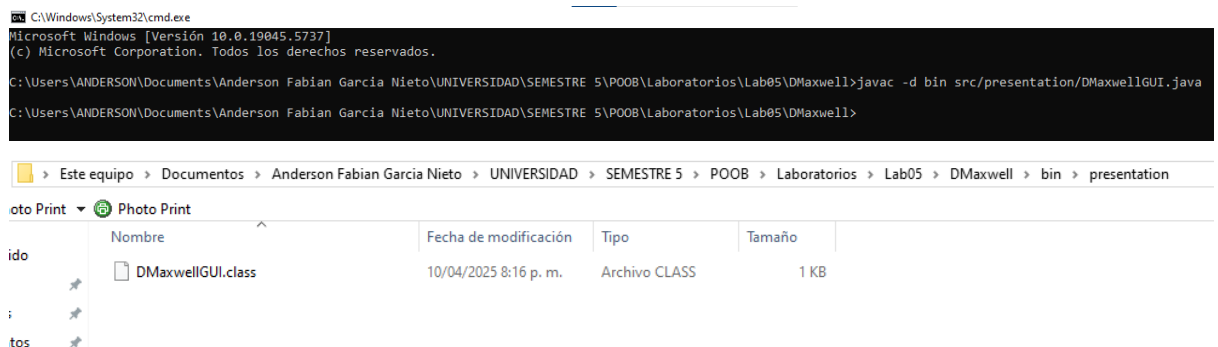
```
package presentation;
import javax.swing.JFrame;

public class DMaxwellGUI extends JFrame {

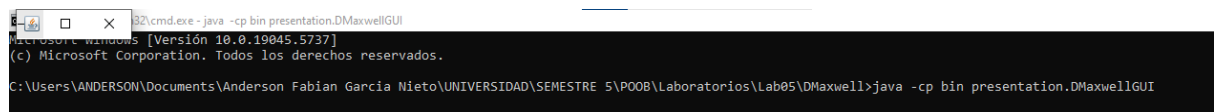
    public DMaxwellGUI() {
        super("Maxwell Discreto"); //Nombre de la ventana
    }

    public static void main(String[] args) {
        DMaxwellGUI ventana = new DMaxwellGUI();
        ventana.setVisible(true); //Estado de la ventada
    }
}
```

- c. Se compila la clase DMaxwellGUI.java desde el directorio raiz con el comando “**javac -d bin src/presentation/DMaxwellGUI.java**”, javac = Compilador de java, -d bin = Donde se guardará el archivo compilado

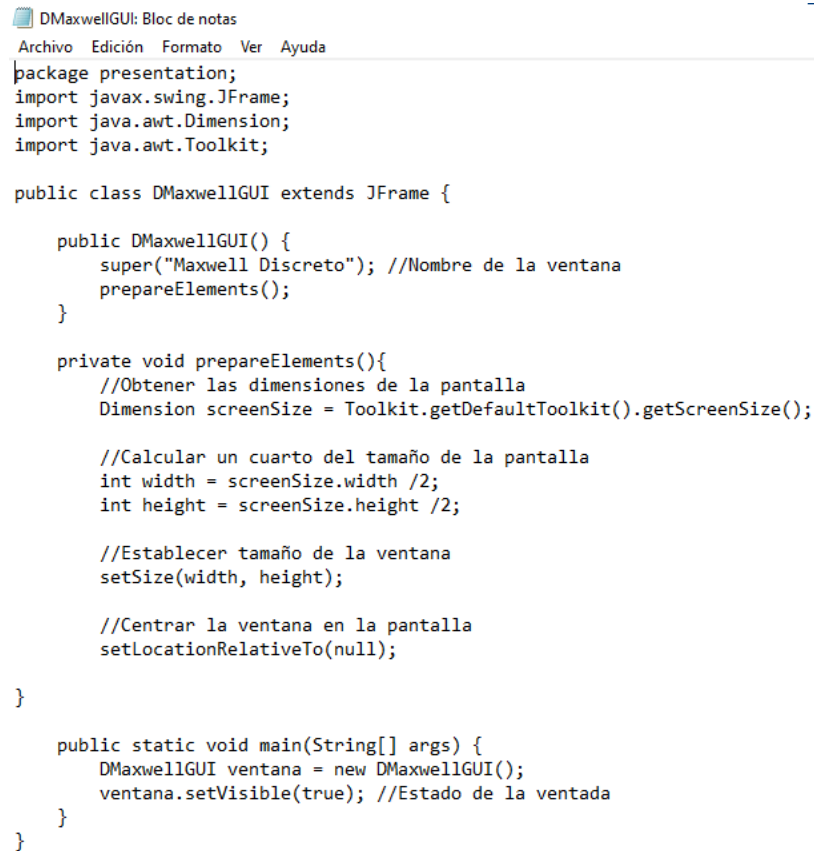


- d. Desde el directorio raiz, se ejecuta el programa a través del siguiente código “**java -cp bin presentation.DMaxwellGUI**”



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.



```
DMaxwellGUI: Bloc de notas
Archivo Edición Formato Ver Ayuda
package presentation;
import javax.swing.JFrame;
import java.awt.Dimension;
import java.awt.Toolkit;

public class DMaxwellGUI extends JFrame {

    public DMaxwellGUI() {
        super("Maxwell Discreto"); //Nombre de la ventana
        prepareElements();
    }

    private void prepareElements(){
        //Obtener las dimensiones de la pantalla
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

        //Calcular un cuarto del tamaño de la pantalla
        int width = screenSize.width /2;
        int height = screenSize.height /2;

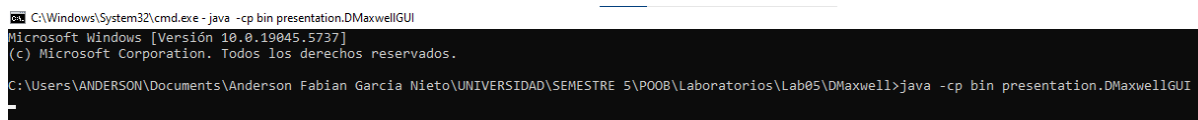
        //Establecer tamaño de la ventana
        setSize(width, height);

        //Centrar la ventana en la pantalla
        setLocationRelativeTo(null);
    }

    public static void main(String[] args) {
        DMaxwellGUI ventana = new DMaxwellGUI();
        ventana.setVisible(true); //Estado de la ventada
    }
}
```

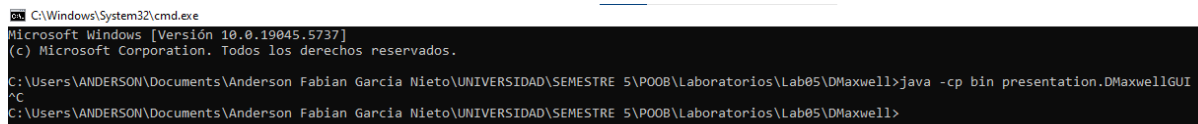
3. Traten de cerrar la ventana. ¿Termina la ejecución? ¿Qué deben hacer en consola para terminar la ejecución?

Al momento de cerrar la ventana, se cierra la GUI (Graphical User Interface), pero la consola sigue activa (No sale la siguiente línea para colocar el nuevo comando)



```
C:\Windows\System32\cmd.exe - java -cp bin presentation.DMaxwellGUI
Microsoft Windows [Versión 10.0.19045.5737]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\ANDERSON\Documents\Anderson Fabian García Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>java -cp bin presentation.DMaxwellGUI
```

Para terminar la ejecución lo que se hizo fue presionar Ctrl + C, haciendo que el proceso finalice manualmente:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5737]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\ANDERSON\Documents\Anderson Fabian García Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>java -cp bin presentation.DMaxwellGUI
^C
C:\Users\ANDERSON\Documents\Anderson Fabian García Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>
```

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?

public void setDefaultCloseOperation(int operación)

Establece la operación que se realizará por defecto cuando el usuario cierre este marco. Debe especificar una de las siguientes opciones:

DO_NOTHING_ON_CLOSE (definido en WindowConstants): No hacer nada; requerir que el programa maneje la operación en el windowClosing método de un WindowListener objeto registrado.

HIDE_ON_CLOSE (definido en WindowConstants): Oculta automáticamente el marco después de invocar cualquier WindowListener objeto registrado.

DISPOSE_ON_CLOSE (definido en WindowConstants): Oculta y elimina automáticamente el marco después de invocar cualquier WindowListener objeto registrado.

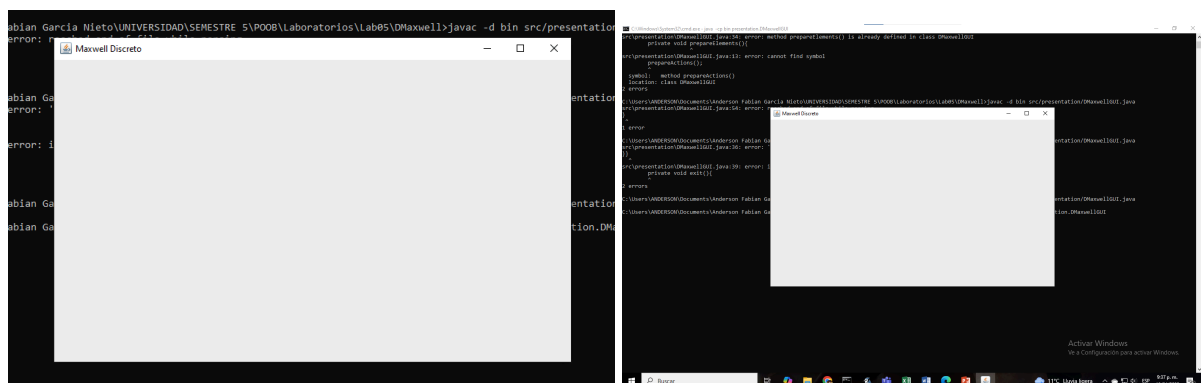
EXIT_ON_CLOSE (definido en JFrame): Salir de la aplicación usando el System exit método. Úselo solo en aplicaciones.

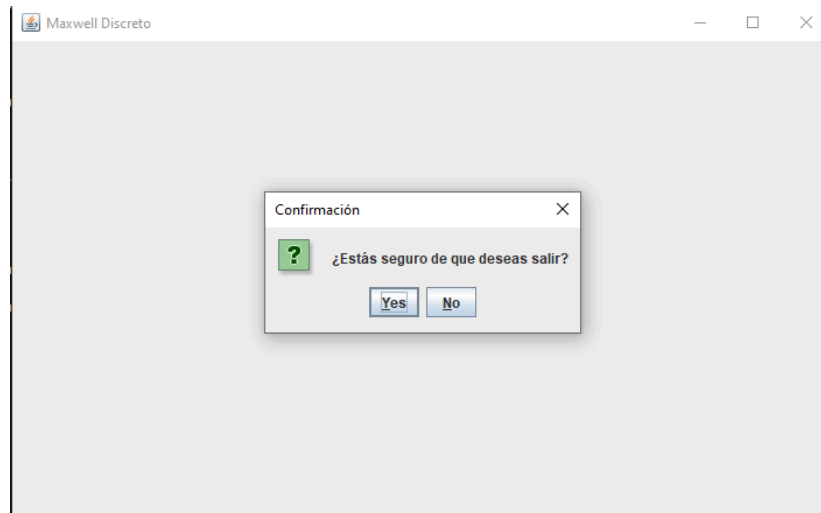
-Si queremos confirmar el cierre de la aplicación, se usa el método, enviando el parámetro (JFrame.DO_NOTHING_ON_CLOSE) y también se debe agregar un Listener para manejar el evento del cierre con un JOptionPane.

-Si queremos cerrar simplemente la aplicación, deberíamos usar el método, pero con argumento (JFrame.EXIT_ON_CLOSE), de la siguiente manera:

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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.





Se termina de ejecutar el programa

```
C:\Windows\System32\cmd.exe
src\presentation\DMaxwellGUI.java:34: error: method prepareElements() is already defined in class DMaxwellGUI
    private void prepareElements(){
                ^
src\presentation\DMaxwellGUI.java:13: error: cannot find symbol
    prepareActions();
    ^
    symbol:   method prepareActions()
    location: class DMaxwellGUI
2 errors

C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>javac -d bin src\presentation\DMaxwellGUI.java
src\presentation\DMaxwellGUI.java:54: error: reached end of file while parsing
}
^
1 error

C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>javac -d bin src\presentation\DMaxwellGUI.java
src\presentation\DMaxwellGUI.java:36: error: ')' or ',' expected
    }}
    ^
src\presentation\DMaxwellGUI.java:39: error: illegal start of expression
    private void exit(){
    ^
2 errors

C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>javac -d bin src\presentation\DMaxwellGUI.java
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>java -cp bin presentation.DMaxwellGUI
C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\DMaxwell>
```

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.

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

Para crear un menú en una aplicación de Java se necesita los siguientes componentes:

-JMenuBar: Es la barra de menú principal que se ubica en la parte superior de la ventana, esta puede contener uno o varios menús desplegables.

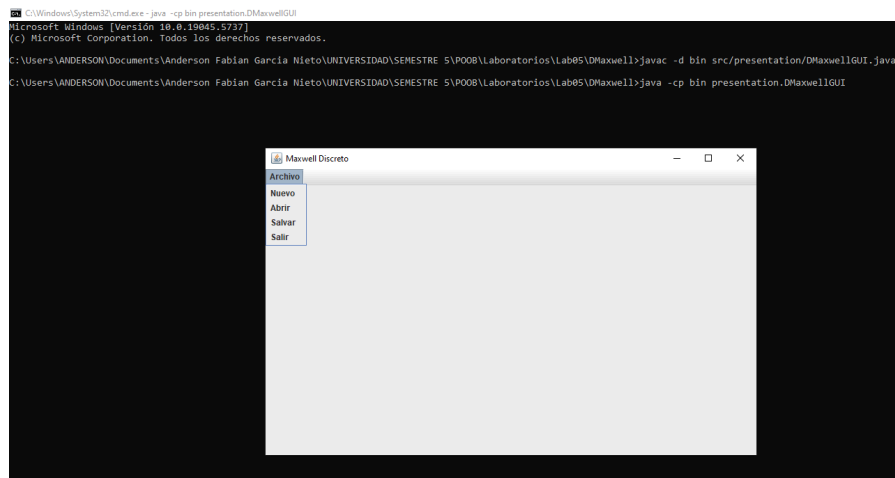
-JMenu: Representa cada uno de los menús desplegables dentro de la barra.

-JMenuItem: Son las opciones individuales del menú que el usuario puede seleccionar.

-JSeparator: Es un componente visual que se usa para dividir o separar grupos de opciones dentro del menú para mejorar la organización.

Los que son atributos son los 3 primeros (JMenuBar, JMenu, JMenuItem), debido a que estos se pueden personalizar, y se pueden acceder después de creados, para su modificación, en cambio JSeparator apenas es creado, no es necesario acceder a este.

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




```

DMaxwellGUI: Bloc de notas
Archivo Edición Formato Ver Ayuda
}

private void prepareElements() {

    setTitle("Maxwell Discreto");

    // Obtener las dimensiones de la pantalla
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

    // Calcular un cuarto del tamaño de la pantalla
    int width = screenSize.width / 2;
    int height = screenSize.height / 2;

    // Establecer tamaño de la ventana
    setSize(width, height);

    // Centrar la ventana en la pantalla
    setLocationRelativeTo(null);

    // Crear los elementos del menu
    prepareElementsMenu();
}

private void prepareElementsMenu(){

    menuBar = new JMenuBar();
    menuArchivo = new JMenu("Archivo");

    menuNuevo = new JMenuItem("Nuevo");
    menuAbrir = new JMenuItem("Abrir");
    menuSalvar = new JMenuItem("Salvar");
    menuSalir = new JMenuItem("Salir");

    menuArchivo.add(menuNuevo);
    menuArchivo.add(menuAbrir);
    menuArchivo.add(menuSalvar);
    menuArchivo.add(menuSalir);

    menuBar.add(menuArchivo);
    setJMenuBar(menuBar);
}

```

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

```

private void prepareActions() {
    // Oyente NO del menú: Ventana de cierre
    addWindowListener(new WindowAdapter() { // Listener prepareActions
        @Override
        public void windowClosing(WindowEvent e) {
            exitApplication();
        }
    });

    // Oyente NO del menú: Redimensionamiento de la ventana
    addComponentListener(new java.awt.event.ComponentAdapter() { // Listener prepareActions
        public void componentResized(java.awt.event.ComponentEvent evt) {
            refresh();
        }
    });

    // Oyente NO del menú: Botón Start
    startButton.addActionListener(e -> toggleStart()); // Listener prepareActions

    // Oyente NO del menú: Flecha Arriba
    arrowButtons[0].addActionListener(e -> moveAllParticles(DMaxwell.UP)); // Listener prepareActions

    // Oyente NO del menú: Flecha Izquierda
    arrowButtons[1].addActionListener(e -> moveAllParticles(DMaxwell.LEFT)); // Listener prepareActions

    // Oyente NO del menú: Flecha Derecha
    arrowButtons[2].addActionListener(e -> moveAllParticles(DMaxwell.RIGHT)); // Listener prepareActions

    // Oyente NO del menú: Flecha Abajo
    arrowButtons[3].addActionListener(e -> moveAllParticles(DMaxwell.DOWN)); // Listener prepareActions
}

```

```

private void prepareActionsMenu() {
    // Oyente del menú: Nuevo
    newMenuItem.addActionListener(e -> resetBoard()); // Listener prepareActionsMenu

    // Oyente del menú: Reiniciar
    resetMenuItem.addActionListener(e -> resetGame()); // Listener prepareActionsMenu

    // Oyente del menú: Salir
    exitMenuItem.addActionListener(e -> exitApplication()); // Listener prepareActionsMenu

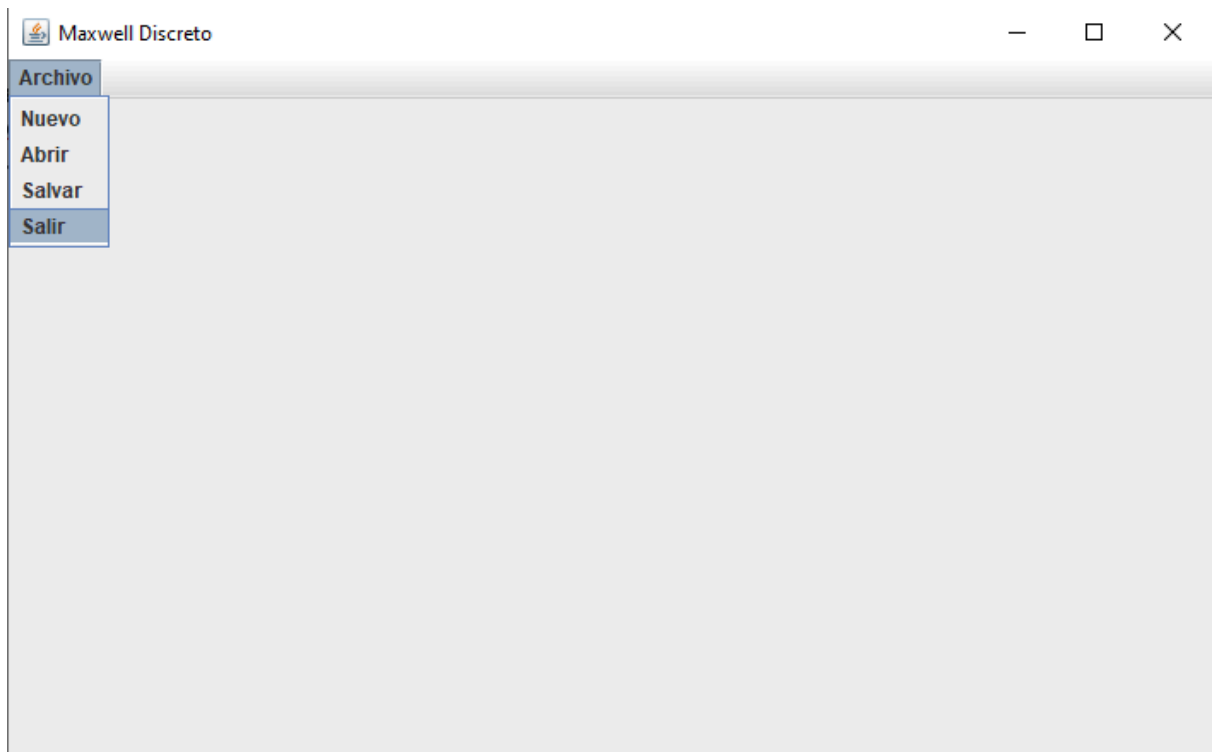
    // Oyente del menú: Cambiar Color Partículas
    changeColorMenuItem.addActionListener(e -> changeParticleColor()); // Listener prepareActionsMenu

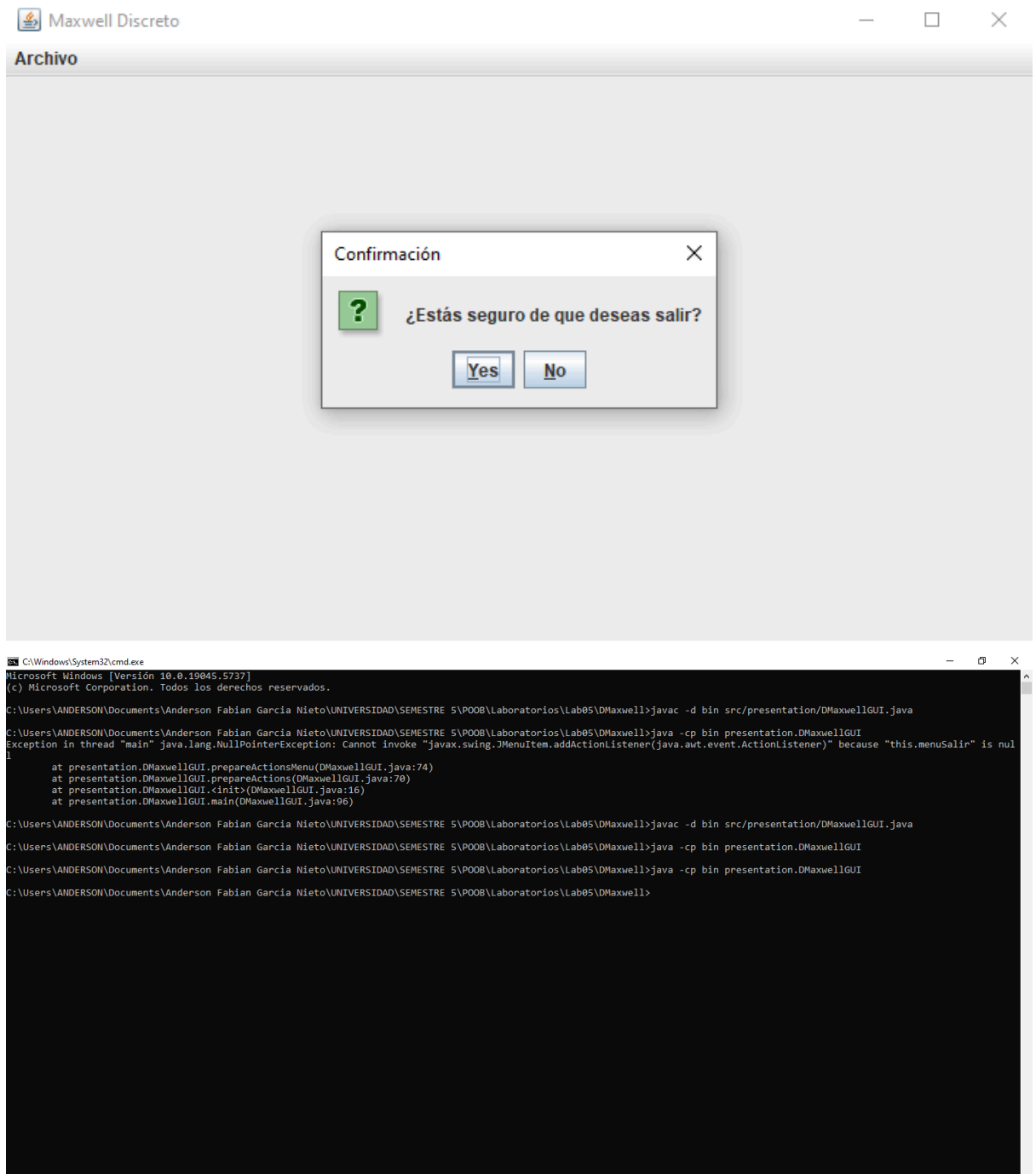
    // Oyente del menú: Abrir
    openMenuItem.addActionListener(new ActionListener() { // Listener prepareActionsMenu
        public void actionPerformed(ActionEvent e) {
            JFileChooser chooser = new JFileChooser();
            int resultado = chooser.showOpenDialog(DMaxwellGUI.this);

            if (resultado == JFileChooser.APPROVE_OPTION) {
                File archivo = chooser.getSelectedFile();
                JOptionPane.showMessageDialog(
                    DMaxwellGUI.this,
                    "Función 'Abrir' en construcción.\nArchivo seleccionado: " + archivo.getName(),
                    title:"Aviso",
                    JOptionPane.INFORMATION_MESSAGE
                );
            }
        }
    });

    // Oyente del menú: Salvar
    saveMenuItem.addActionListener(new ActionListener() { // Listener prepareActionsMenu
        public void actionPerformed(ActionEvent e) {
            JFileChooser chooser = new JFileChooser();

```





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: Inicializa el cuadro de diálogo para seleccionar archivos.

Ejemplo:

JFileChooser chooser = new JFileChooser("C:/Users/Usuario/MisArchivos");

-showOpenDialog: Muestra un diálogo para abrir un archivo.

-showSaveDialog: Muestra un diálogo para guardar un archivo.

-getSelectedFile: Devuelve el archivo seleccionado por el usuario como un objeto File.

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.

```
//Accion para abrir
menuAbrir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFileChooser chooser = new JFileChooser();
        int resultado = chooser.showOpenDialog(DMaxwellGUI.this);

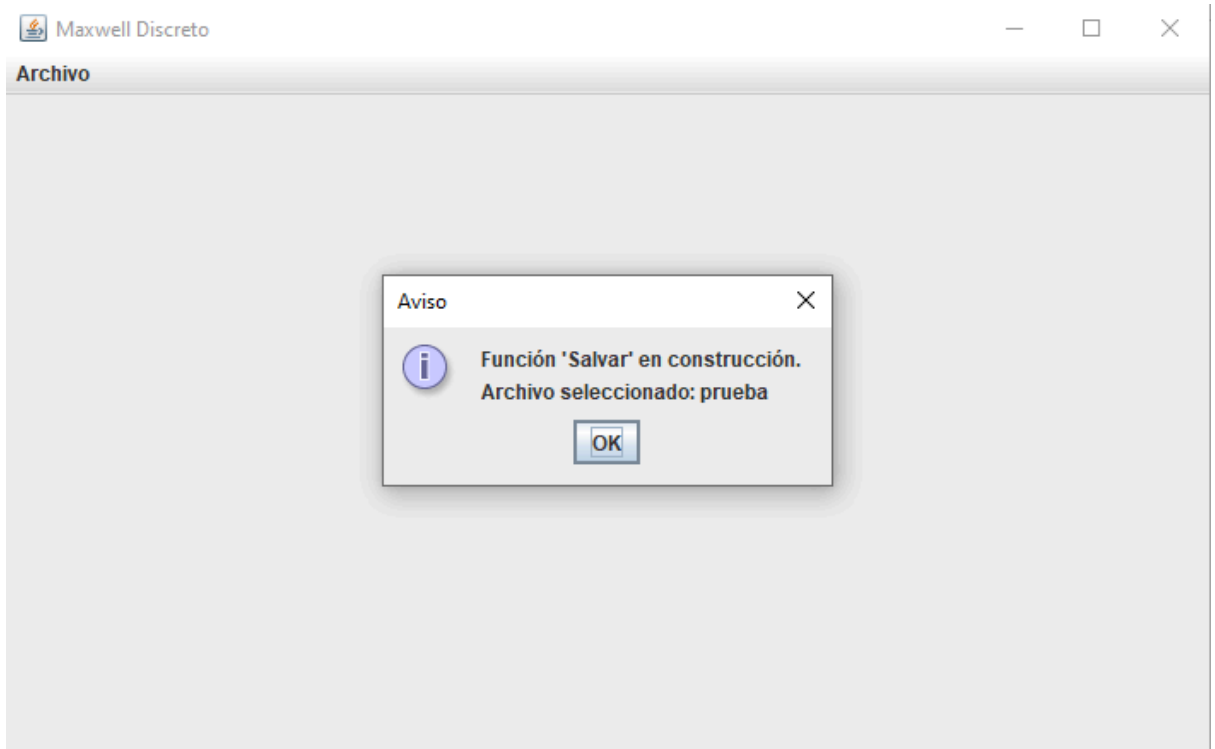
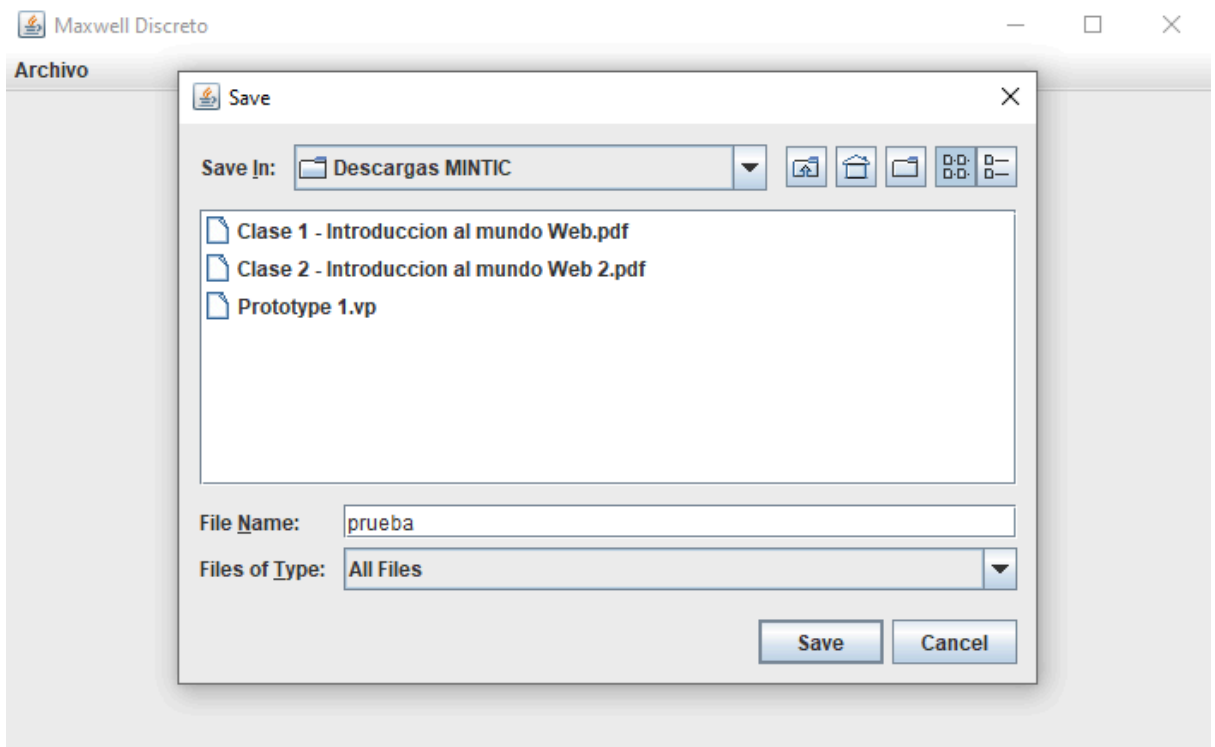
        if (resultado == JFileChooser.APPROVE_OPTION) {
            File archivo = chooser.getSelectedFile();
            JOptionPane.showMessageDialog(
                DMaxwellGUI.this,
                "Función 'Abrir' en construcción.\nArchivo seleccionado: " + archivo.getName(),
                "Aviso",
                JOptionPane.INFORMATION_MESSAGE
            );
        }
    }
});

// Acción para "Salvar"
menuSalvar.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFileChooser chooser = new JFileChooser();
        int resultado = chooser.showSaveDialog(DMaxwellGUI.this);

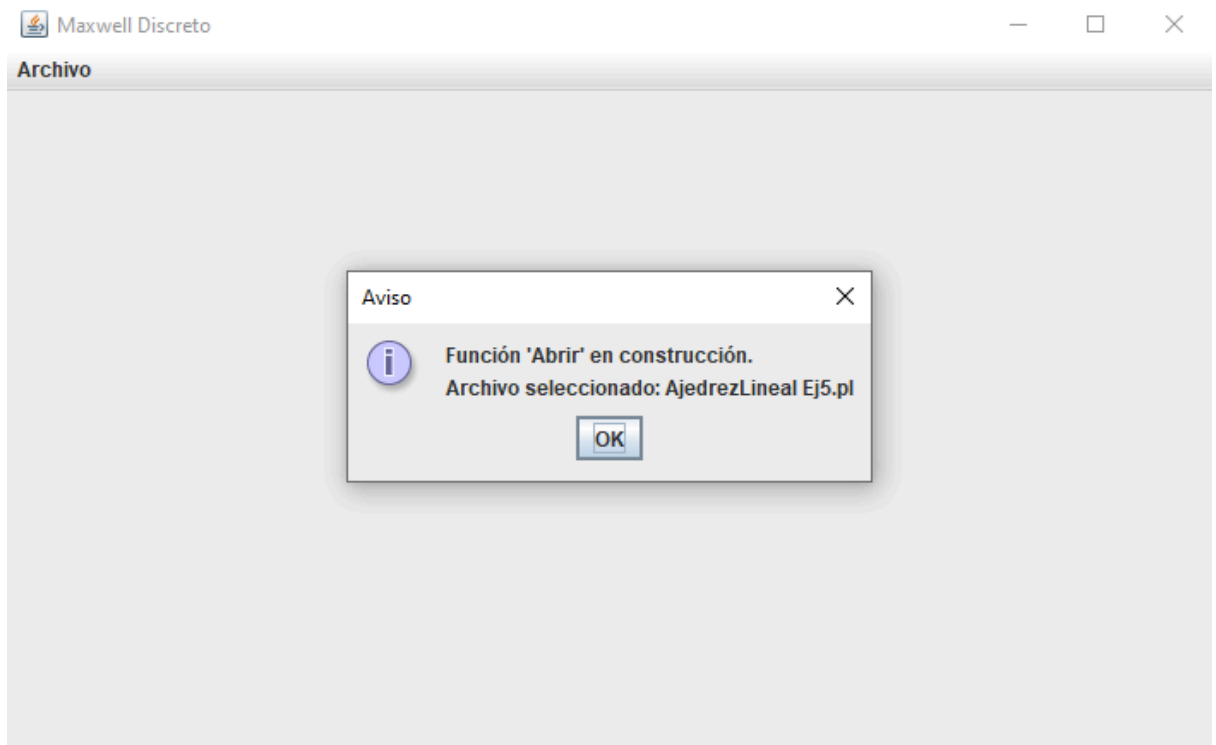
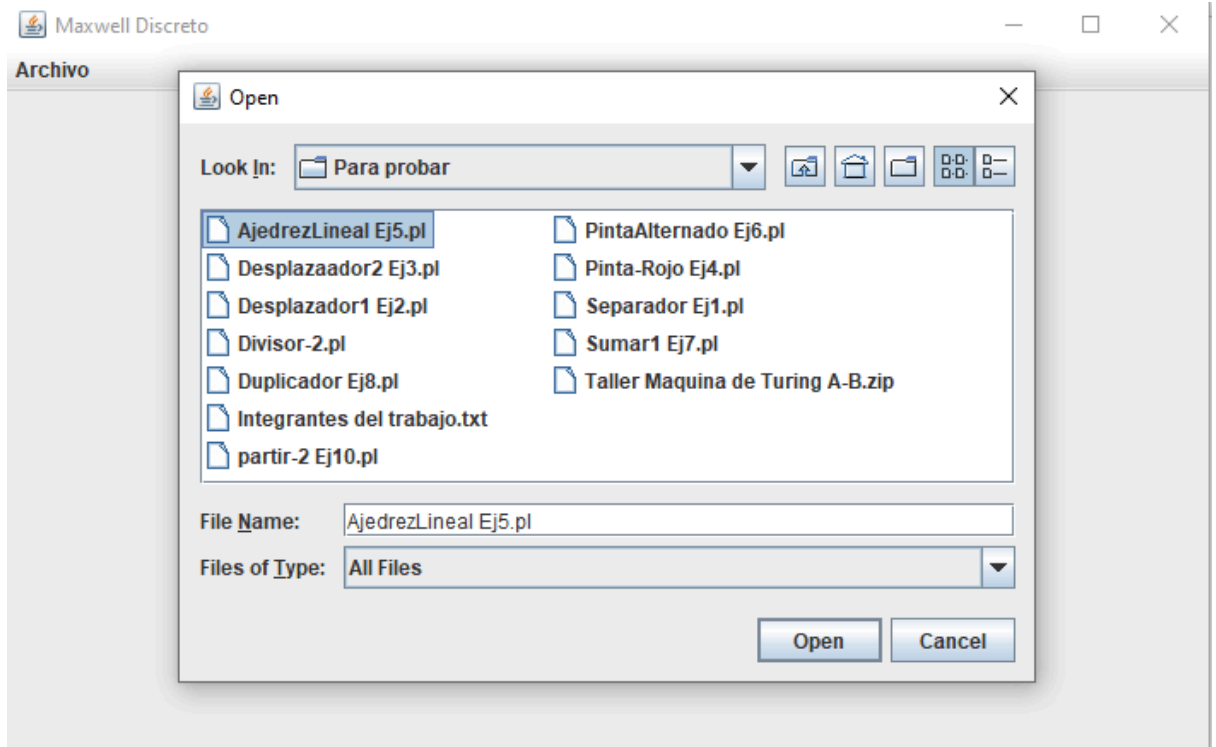
        if (resultado == JFileChooser.APPROVE_OPTION) {
            File archivo = chooser.getSelectedFile();
            JOptionPane.showMessageDialog(
                DMaxwellGUI.this,
                "Función 'Salvar' en construcción.\nArchivo seleccionado: " + archivo.getName(),
                "Aviso",
                JOptionPane.INFORMATION_MESSAGE
            );
        }
    }
});
```

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

Funcionalidad de Salvar



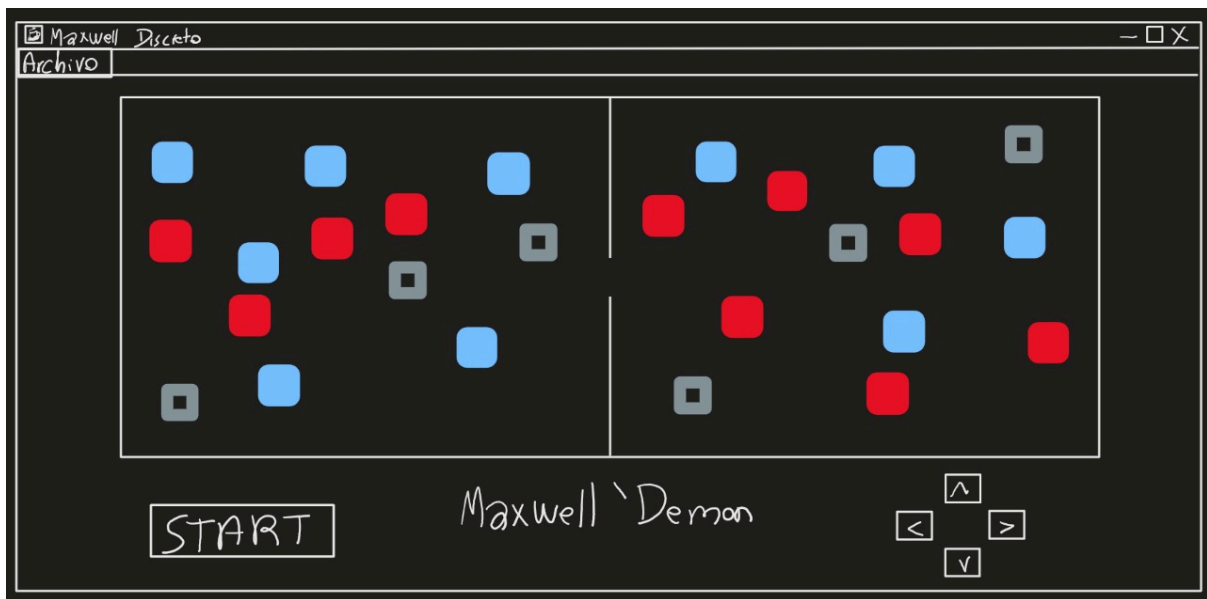
Funcionalidad de Abrir



CICLO 3

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. (Incluyan la imagen)



2. Continúen con la implementación definiendo los atributos necesarios y extendiendo el método `prepareElements()`.

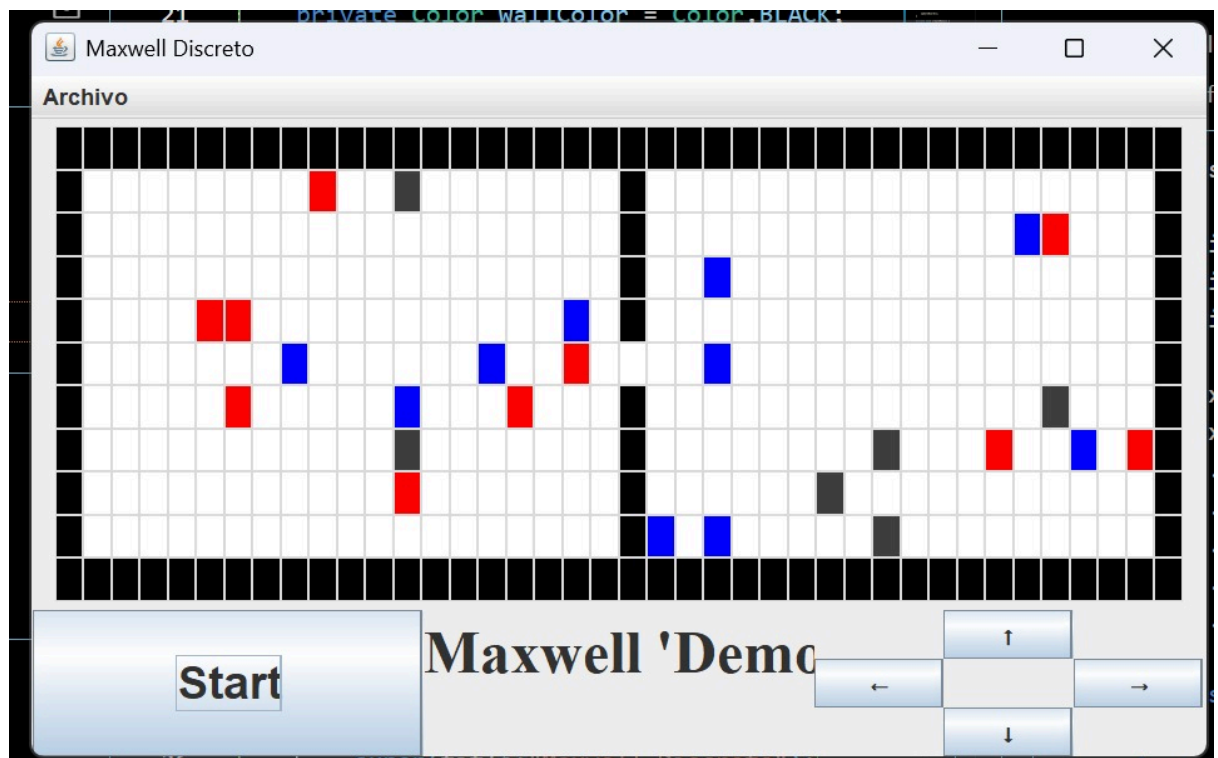
Para la zona del tablero definen un método `prepareElementsBoard()` y un método `refresh()` que actualiza la vista del tablero considerando, por ahora, el tablero inicial por omisión. Este método lo vamos a implementar realmente en otros ciclos.

```
private void prepareElements() {
    maxwell = new DMaxwell(rows, cols, particles, holes); //Crea un objeto de DMaxwell
    setupFrame();
    setupMainPanel();
    prepareElementsMenu();
    createMenu();
    createButtonsAndArrows();
    prepareActions();
    prepareActionsMenu();
    setVisible(b:true);
}
```

```
private void prepareElementsMenu() {
    boardPanel = new JPanel(new GridLayout(rows, cols));
    mainPanel.add(boardPanel, BorderLayout.CENTER);

    createHoles();
    createParticles();
    refresh(); // Se usa refreah para actualizar el tablero al inicio
}
```

3. Ejecuten y capturen la pantalla.



CICLO 4 CAMBIAR COLORES

El objetivo es implementar este caso de uso.

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

Para implementar el caso de uso "Cambiar los colores", se utilizan diversos elementos correspondientes a los componentes de la arquitectura Modelo-Vista-Controlador (MVC). A continuación se describen los elementos involucrados en cada uno de estos componentes.

Vista (Interfaz de Usuario):

En la interfaz gráfica se encuentra un elemento llamado `changeColorMenuItem`, que corresponde a una opción dentro del menú "Configuración". Este ítem desencadena el proceso de cambio de color cuando el usuario lo selecciona. Posteriormente, se utiliza un `JOptionPane` para mostrar un cuadro de diálogo donde el usuario puede elegir si desea modificar las partículas rojas o las azules. Una vez hecho esto, se despliega un componente `JColorChooser`, que permite seleccionar el nuevo color desde una paleta gráfica. Finalmente, el método `refresh()` de la clase `DMaxwellGUI` se encarga de actualizar el `boardPanel`, asegurando que el nuevo color aplicado a las partículas se refleje correctamente en la interfaz.

Controlador (Lógica de Interacción):

La lógica de interacción inicia con el ActionListener asociado al `changeColorMenuItem`, el cual responde al evento de clic generado por el usuario. Este listener llama al método `changeParticleColor()`, que contiene el flujo principal de este caso de uso. En dicho método, primero se muestra un `JOptionPane` para que el usuario elija el tipo de partícula a modificar (rojas o azules). Después, se despliega un `JColorChooser` para que el usuario seleccione el nuevo color. Con esta información, el controlador procede a iterar sobre la lista de partículas y modifica el color de aquellas que coincidan con el tipo seleccionado. Finalmente, se llama al método `refresh()` para redibujar el tablero y reflejar los cambios en la vista.

Modelo (Datos y Estado):

El modelo está compuesto principalmente por una lista de objetos `Particle`, que representan todas las partículas del sistema. Cada objeto `Particle` contiene un atributo `color`, el cual determina su apariencia en la interfaz. Además, la clase `Particle` dispone del método `setColor(Color c)`, que permite modificar dicho atributo. Una vez que el modelo ha sido actualizado por el controlador, se procede a actualizar la vista para que los cambios sean visibles al usuario.

Flujo del Caso de Uso:

El flujo comienza cuando el usuario selecciona la opción "Cambiar Color Partículas" desde el menú. A continuación, el controlador muestra los cuadros de diálogo para elegir el tipo de partícula y el nuevo color. Después, se actualizan las propiedades de color en los objetos correspondientes dentro de la lista `particles`. Por último, se invoca el método `refresh()` para que la vista muestre los cambios aplicados.

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

`JColorChooser` es un componente de la biblioteca Swing de Java que permite a los usuarios seleccionar colores mediante una interfaz gráfica intuitiva. Este componente ofrece varias formas de selección, incluyendo paletas visuales, deslizadores para los modelos de color RGB y HSB, campos numéricos y la posibilidad de definir muestrarios personalizados. Es ampliamente utilizado en aplicaciones gráficas que requieren que el usuario personalice colores de elementos visuales, como botones, fondos o formas.

Uno de los métodos más utilizados de este componente es el método estático `showDialog()`. Este método abre un cuadro de diálogo modal que contiene una instancia de `JColorChooser`, permitiendo la selección interactiva de un color. La modalidad del diálogo implica que la ventana padre queda bloqueada hasta que el usuario toma una decisión, ya sea seleccionando un color o cancelando la operación. Esta característica lo hace similar a componentes como `JOptionPane`.

El método `showDialog()` recibe tres parámetros: el componente padre (`parent`), un título para la ventana (`title`) y un color inicial (`initialColor`) que aparece preseleccionado al abrir el diálogo. Si el parámetro `parent` es `null`, el cuadro de diálogo se centra en la pantalla. En cuanto al color inicial, si se pasa un valor `null`, el diálogo utilizará el color blanco (`Color.WHITE`) por defecto.

El valor de retorno del método es un objeto de tipo `Color`, que representa la elección del usuario. Si el usuario hace clic en "Aceptar", se devuelve el color seleccionado. En cambio, si cancela o cierra el cuadro de diálogo sin confirmar, el método devuelve `null`. Por ello, es recomendable verificar si el resultado no es nulo antes de utilizarlo en la aplicación, usando una instrucción como `if (color != null)`.

Dentro del cuadro de diálogo generado por `showDialog()`, se incluyen múltiples pestañas para distintos modos de selección de color, una previsualización en tiempo real del color elegido y botones de confirmación y cancelación.

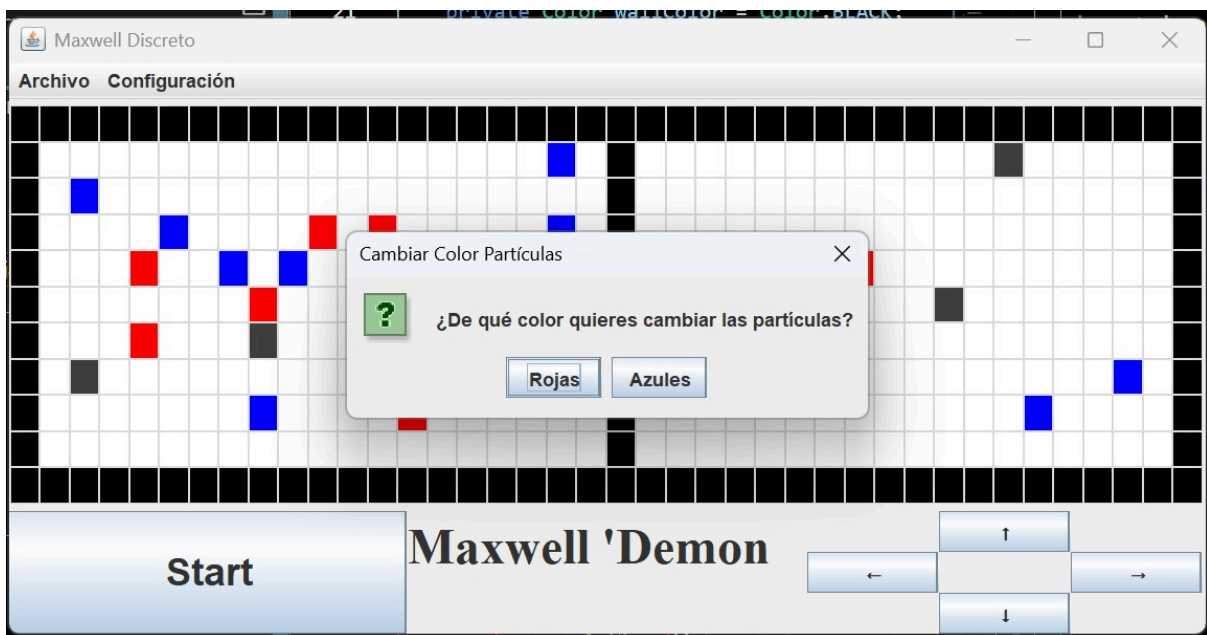
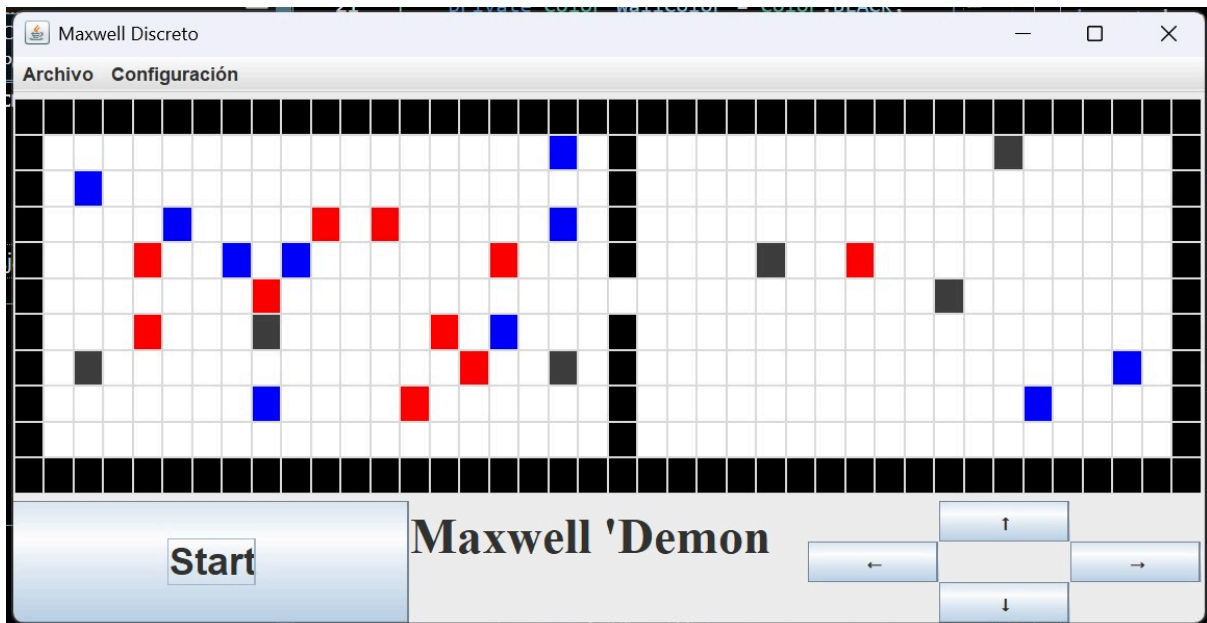
3. Implementen los componentes necesarios para cambiar el color de las fichas.

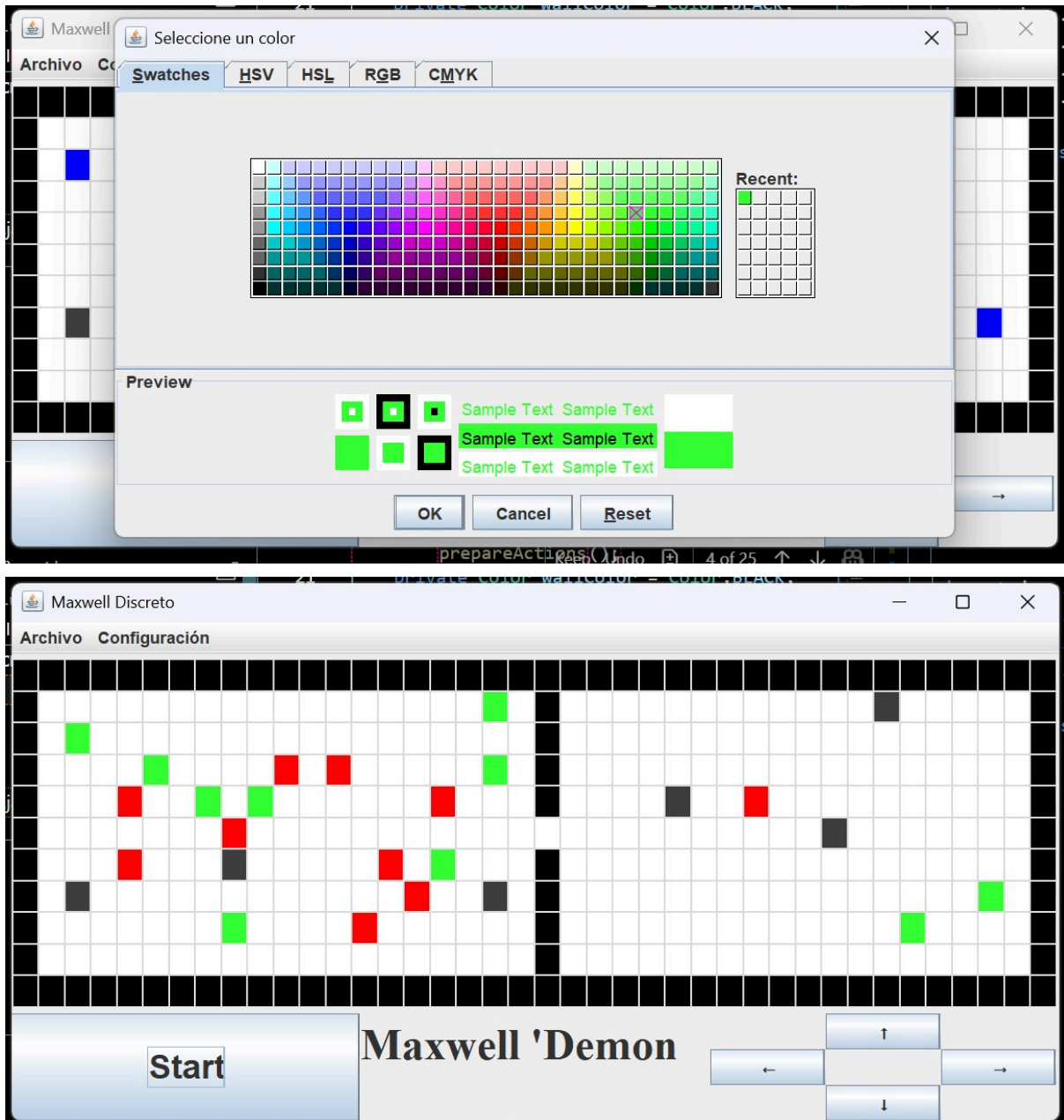
```
private void changeParticleColor() {
    String[] options = {RED_PARTICLES, BLUE_PARTICLES};
    int choice = JOptionPane.showOptionDialog(
        this,
        PARTICLE_COLOR_SELECTION,
        title:"Cambiar Color Partículas",
        JOptionPane.DEFAULT_OPTION,
        JOptionPane.QUESTION_MESSAGE,
        icon:null,
        options,
        options[0]
    );

    Color newColor = JColorChooser.showDialog(this, COLOR_CHOOSER_TITLE, Color.WHITE);

    if (newColor != null) {
        if (choice == 0) { // Cambiar color de las partículas rojas
            for (Particle particle : particles) {
                if (particle.getColor().equals(new Color(r:255, g:0, b:0))) {
                    particle.setColor(newColor);
                }
            }
        } else if (choice == 1) { // Cambiar color de las partículas azules
            for (Particle particle : particles) {
                if (particle.getColor().equals(new Color(r:0, g:0, b:255))) {
                    particle.setColor(newColor);
                }
            }
        }
    }
    refresh();
}
```

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












CICLO 5

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

Este punto se encuentra desarrollador en el archivo .astah

2. Ejecuten las pruebas y capturen el resultado

Para poder ejecutar las pruebas, se necesitó crear la carpeta lib, donde se encuentran todos los archivos necesarios para que las pruebas compilen, los archivos usados fueron:

	apiguardian-api-1.1.2	22/03/2025 10:10 p. m.	Executable Jar File	7 KB
	junit-4.8	22/03/2025 10:05 p. m.	Executable Jar File	232 KB
	junit-jupiter-api-5.9.2	22/03/2025 10:10 p. m.	Executable Jar File	203 KB
	junit-jupiter-engine-5.9.2	22/03/2025 10:10 p. m.	Executable Jar File	241 KB
	junit-platform-commons-1.9.2	22/03/2025 10:10 p. m.	Executable Jar File	101 KB
	junit-platform-console-standalone-1.9.2	22/03/2025 10:10 p. m.	Executable Jar File	2.554 KB
	junit-platform-engine-1.9.2	22/03/2025 10:10 p. m.	Executable Jar File	185 KB

Luego de eso se compilo el paquete de dominio y el paquete de presentación, para poder compilar el paquete de pruebas a traves de este comando **javac -cp "lib/*" -d bin src/domain/*.java src/presentation/*.java src/test/*.java**

Ya con el paquete de pruebas compilado, se usó el siguiente comando para ejecutarlas a través de la consola **java -cp "lib/*;bin;test"**

org.junit.platform.console.ConsoleLauncher --scan-classpath
mostrando como resultado lo siguiente:

```

at org.junit.platform.console.ConsoleLauncher.execute(ConsoleLauncher.java:46)
at org.junit.platform.console.ConsoleLauncher.main(ConsoleLauncher.java:42)

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

+ [36m+ [0m
+ [36m+--- [0m + [36mJUnit Jupiter+ [0m + [32m[OK]+ [0m
+ [36m+--- [0m + [36mJUnit Vintage+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [36mHoleTest+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mgetCol_shouldReturnCorrectCol+ [0m + [32m[OK]+ [0m
+ [36m] | '--- [0m + [34mgetRow_shouldReturnCorrectRow+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [36mSquareTest+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34msetColor_shouldChangeColor+ [0m + [32m[OK]+ [0m
+ [36m] | '--- [0m + [34mgetColor_shouldReturnCorrectColor+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [36mMaxwellTest+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldNotMoveParticleIntoWall+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldReturnCorrectPercentage+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldReturnCorrectInformation+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldNotMoveParticleIfOccupied+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldMoveParticleLeft+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldMoveParticleUp+ [0m + [32m[OK]+ [0m
+ [36m] | +--- [0m + [34mshouldMoveParticleDown+ [0m + [32m[OK]+ [0m
+ [36m] | '--- [0m + [34mshouldMoveParticleRight+ [0m + [32m[OK]+ [0m
+ [36m] '--- [0m + [36mParticleTest+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [34msetColor_shouldChangeColor+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [34msetCol_shouldChangeCol+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [34msetRow_shouldChangeRow+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [34mgetCol_shouldReturnCorrectCol+ [0m + [32m[OK]+ [0m
+ [36m] +--- [0m + [34mgetColor_shouldReturnCorrectColor+ [0m + [32m[OK]+ [0m
+ [36m] '--- [0m + [34mgetRow_shouldReturnCorrectRow+ [0m + [32m[OK]+ [0m
+ [36m] '--- [0m + [36mJUnit Platform Suite+ [0m + [32m[OK]+ [0m

Test run finished after 426 ms
[ 7 containers found ]
[ 0 containers skipped ]
[ 7 containers started ]
[ 0 containers aborted ]
[ 7 containers successful ]
[ 0 containers failed ]
[ 18 tests found ]
[ 0 tests skipped ]
[ 18 tests started ]
[ 0 tests aborted ]
[ 18 tests successful ]
[ 0 tests failed ]

C:\Users\ANDERSON\Documents\Anderson Fabian Garcia Nieto\UNIVERSIDAD\SEMESTRE 5\POOB\Laboratorios\Lab05\Probando\LAB05version04>

```

Verificando que las pruebas de unidad funcionan correctamente

CICLO 6 JUGAR

El objetivo es implementar el caso de uso jugar.

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

```
import domain.DMaxwell; // se importa el modelo del DMaxwell
```

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

```
private void refresh() {
    boardPanel.removeAll();
    boardPanel.setLayout(new GridLayout(rows, cols));

    for (int row = 0; row < rows; row++) {
        for (int col = 0; col < cols; col++) {
            Square cell = null;

            if (isWall(row, col)) {
                cell = new Square(wallColor);
            } else if (isHole(row, col)) {
                cell = getHoleAt(row, col);
            } else if (isParticle(row, col)) {
                cell = getParticleAt(row, col);
            } else {
                cell = new Square(emptyColor);
            }

            if (cell != null) {
                cell.setBorder(new LineBorder(gridColor));
                boardPanel.add(cell);
            }
        }
    }

    boardPanel.revalidate();
    boardPanel.repaint();
}
```

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

- a. Para implementar este caso de uso necesitamos varios elementos que trabajen en conjunto para que la simulación funcione correctamente. Primero, hay que darle funcionalidad al botón Start, que se encargará de iniciar la simulación y habilitar los controles para que el usuario pueda mover las partículas. Luego, debemos programar los botones de dirección, que permitirán al usuario mover las partículas dentro del contenedor. Cada vez que se haga un movimiento, el sistema tiene que verificar si hubo alguna colisión, ya sea con los bordes del contenedor o con algún hole (agujero). Si ocurre una colisión, se actualiza el estado del sistema, por ejemplo, cambiando la posición de las partículas o el número de partículas capturadas. Por último, es importante refrescar la interfaz gráfica para que el usuario vea en tiempo real todos los cambios que se han producido en la simulación.

4. Implementen los componentes necesarios para jugar .

a. ¿Cuántos oyentes necesitan?

5 para este caso de uso pero en el proyecto usamos un total de 8 oyentes

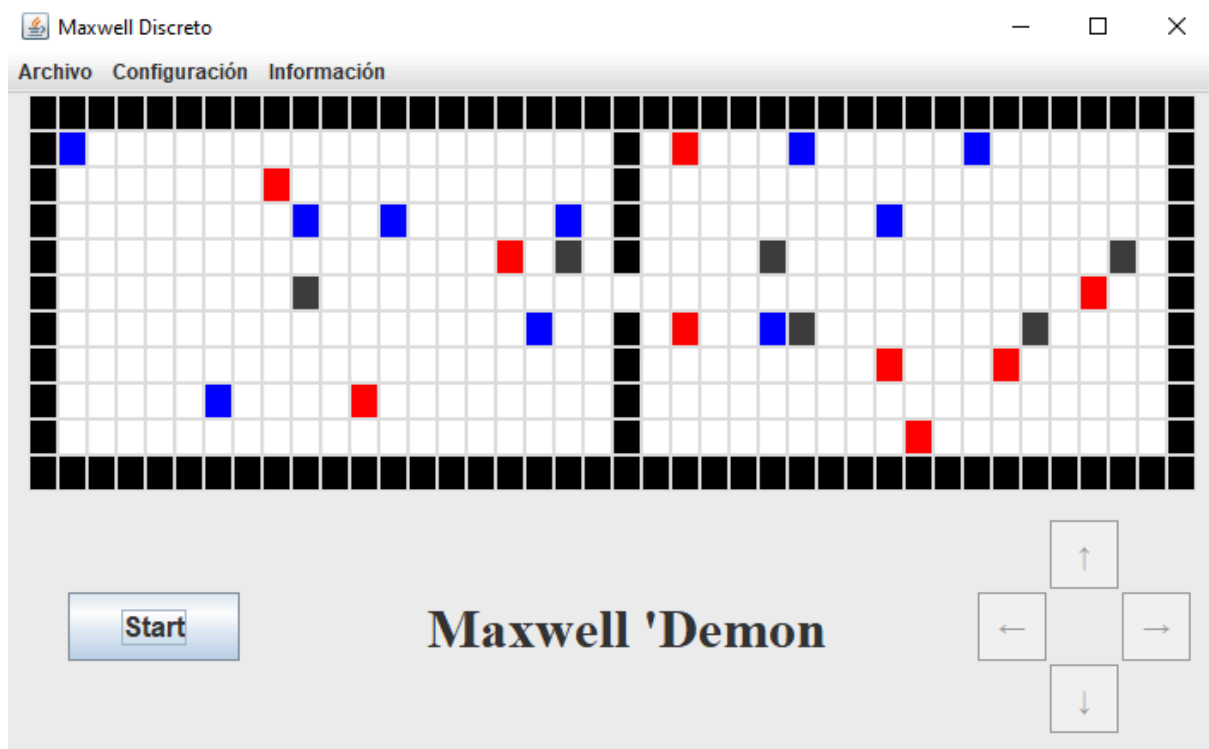
i. ¿Por qué?

El botón start es un oyente encargado de iniciar el juego

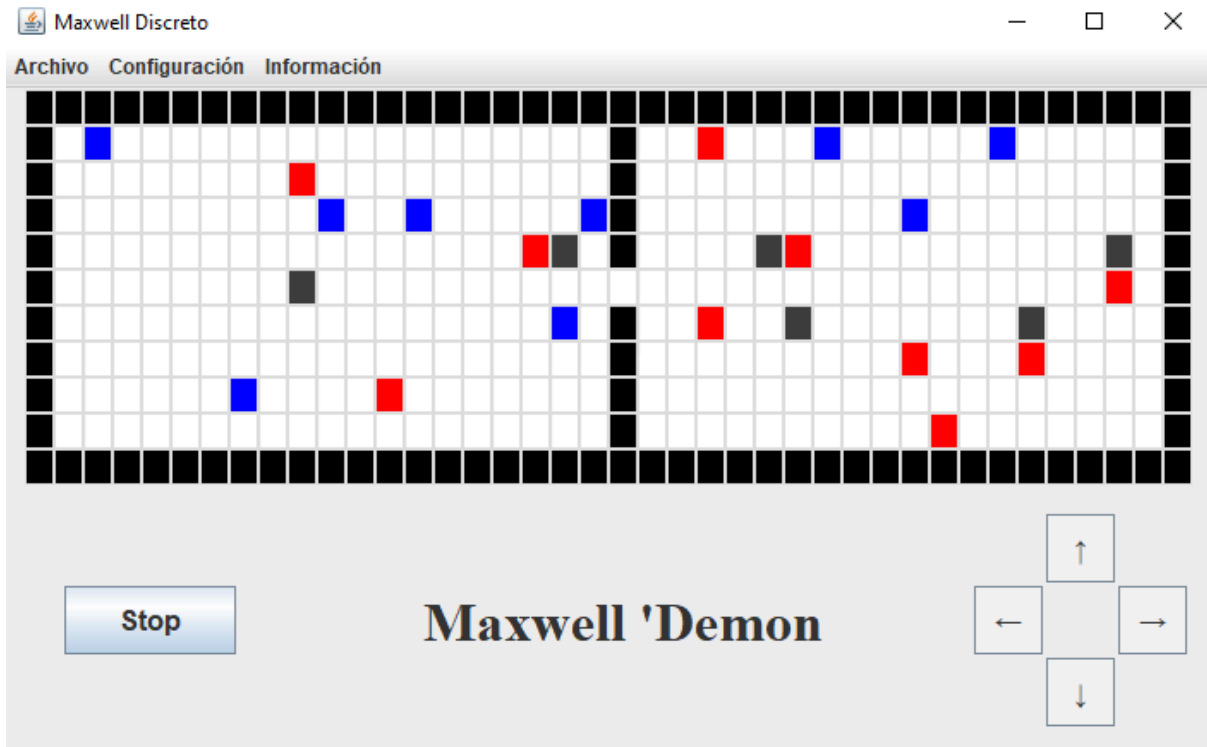
Los 4 botones de las flechas son los encargados de darle dirección a donde debe moverse las partículas

Por último que no son necesarias para el caso de uso, el la de cerrar y los menos para un total de 8

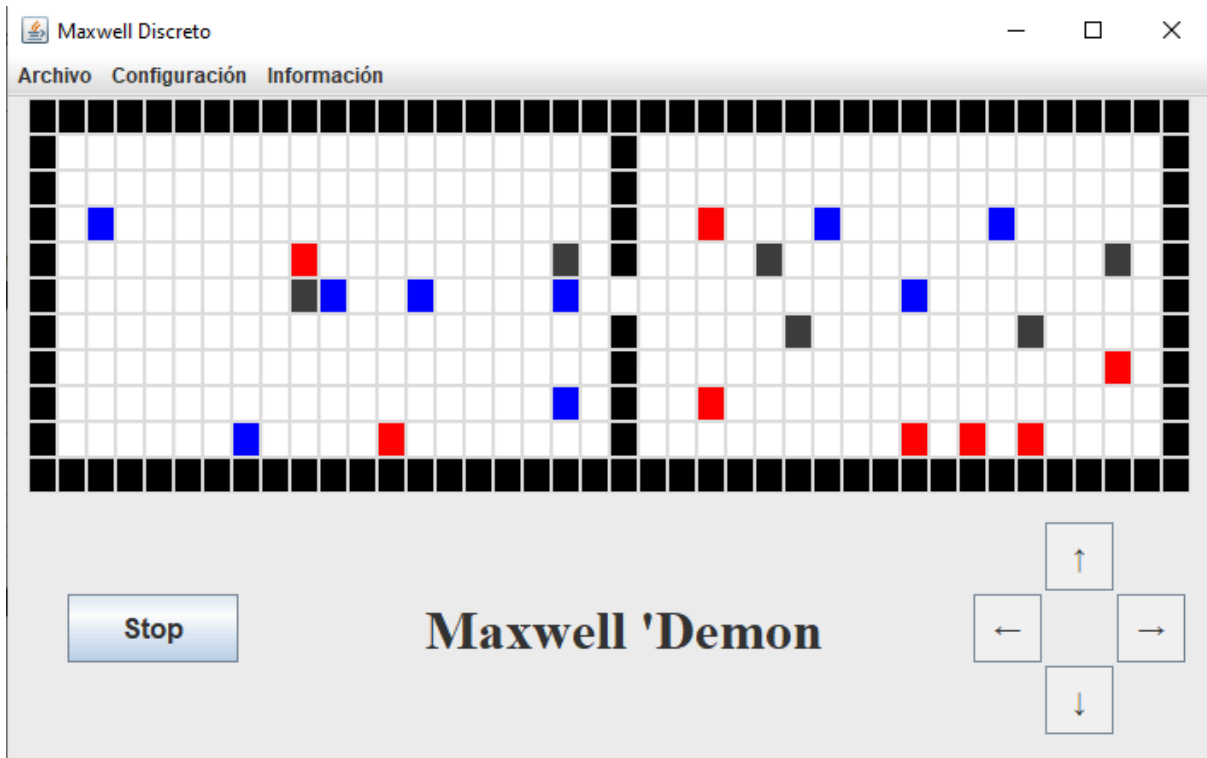
5. Ejecuten el caso de uso y capture las pantallas más significativas.



Click en start y click en la flecha a la derecha.



Más movimientos



Ciclo 7: REINICIAR

El objetivo es implementar este caso de uso.

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

Para implementar este caso de uso, se necesitan varios elementos que permitan reiniciar correctamente el estado del juego. Cuando el usuario selecciona la opción **"Reiniciar"** desde el menú Archivo, lo primero que debe hacer el sistema es detener la simulación si está en ejecución, para evitar que siga corriendo en segundo plano. Luego, se deben limpiar todas las estructuras de datos, como listas o arreglos que almacenen partículas, hoyos, demonios, etc., asegurándonos de que no quede información del estado anterior. Después de eso, se debe recrear el tablero con nuevos elementos, es decir, generar nuevamente las partículas, posiciones iniciales y cualquier otro componente necesario para empezar desde cero. Finalmente, se actualiza la interfaz gráfica para que el usuario vea el tablero reiniciado y listo para una nueva simulación.

2. Implementen los elementos necesarios para reiniciar

```
private void resetGame() {
    // Detener el juego si está corriendo
    running = false;
    setArrowButtonsEnabled(false);

    // Limpiar y recrear el tablero
    particles.clear();
    holes.clear();

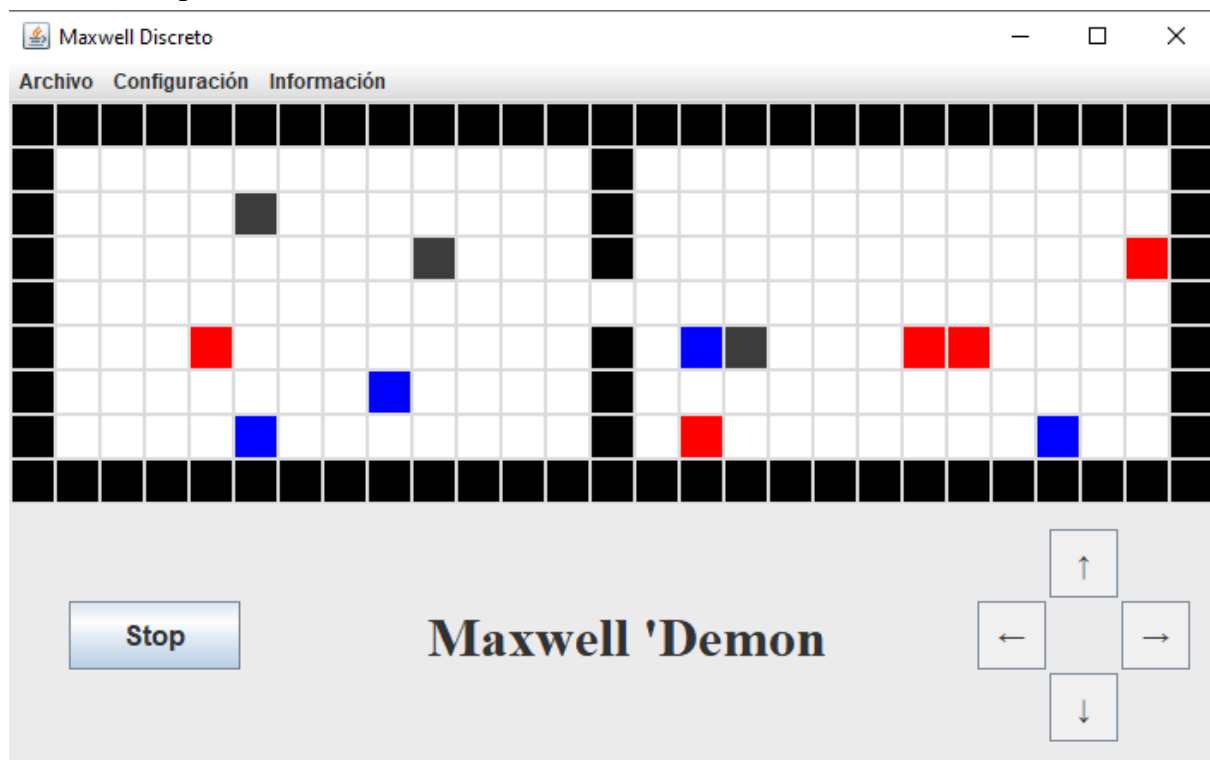
    // Recrear el tablero con nuevas partículas y
    createBoard();

    // Actualizar la visualización
    refresh();
}

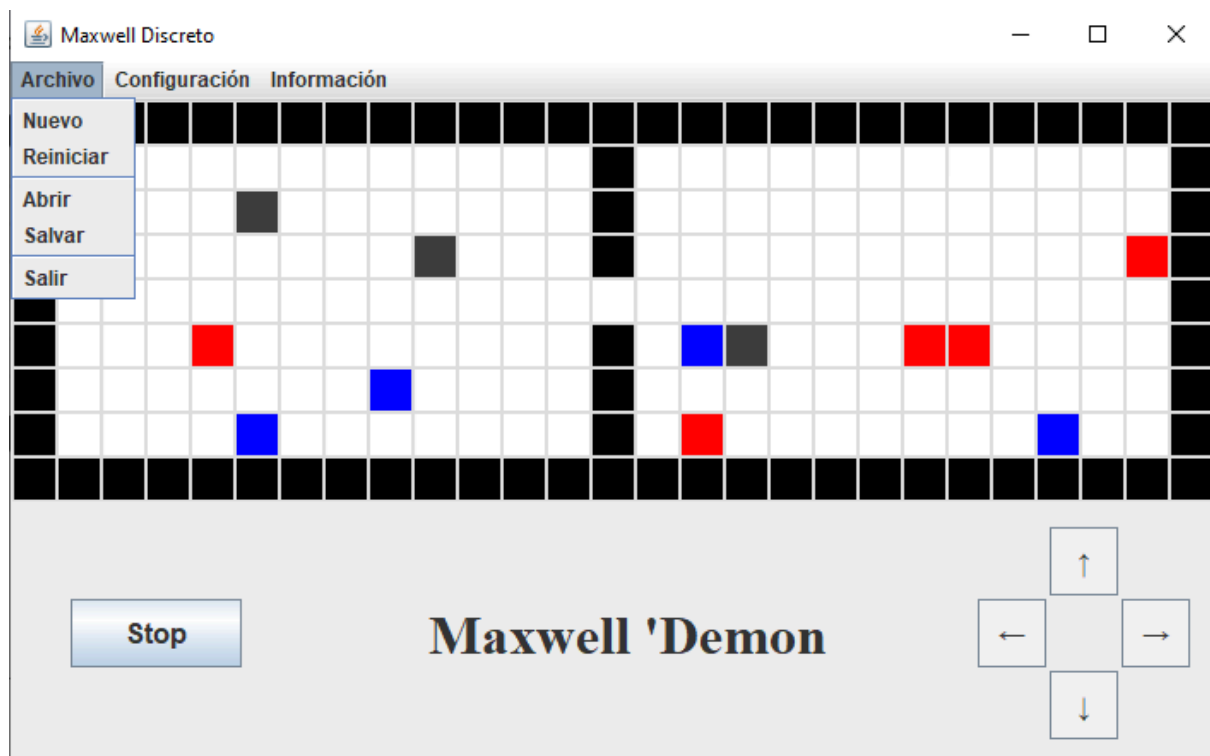
exitMenuItem = new JMenuItem("Salir");
JMenuItem resetMenuItem = new JMenuItem("Reiniciar"); // Nuevo ítem de menú
changeColorMenuItem = new JMenuItem("Cambiar Color Partículas");
fileMenu.add(resetMenuItem);
fileMenu.add(changeColorMenuItem);
fileMenu.add(exitMenuItem);
```

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

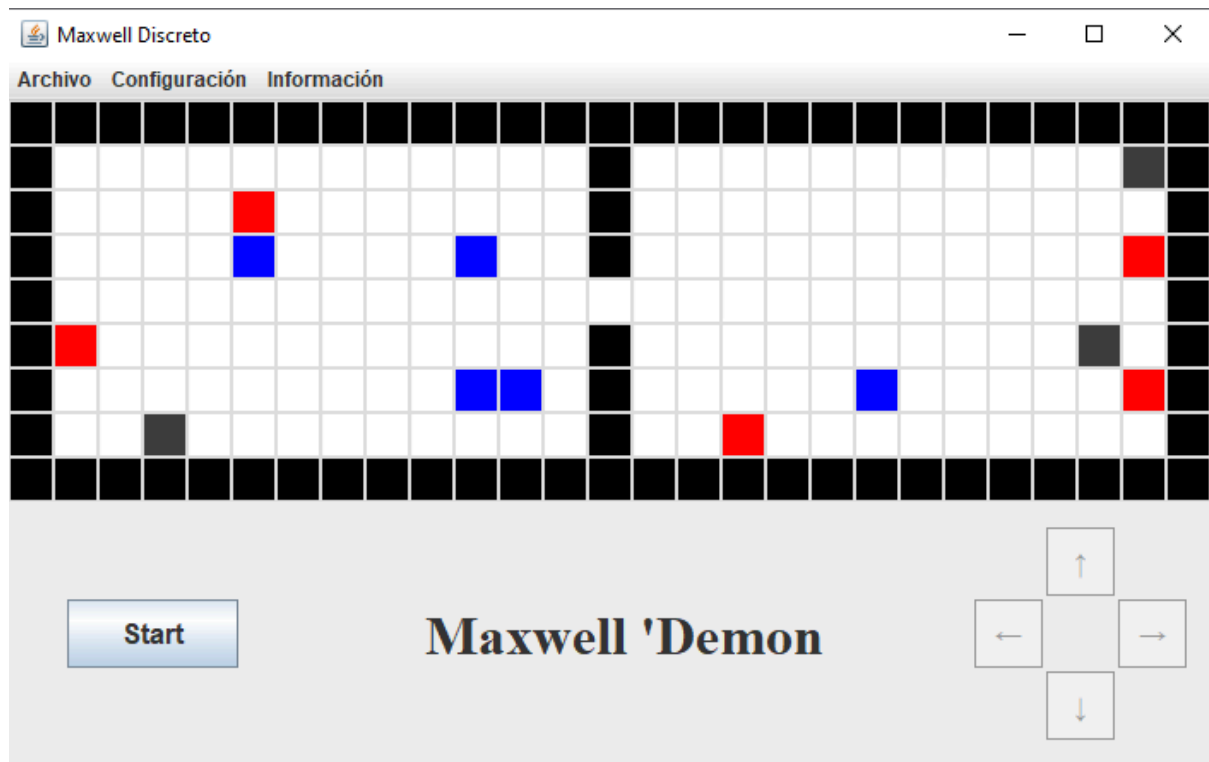
Se tiene una partida, se hacen varios movimientos



Click en reiniciar



El tablero vuelve a la posición original de las fichas



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

Para ajustar el tamaño de la interfaz, utilizamos varios elementos clave. Primero, un `ComponentListener` detecta cambios en el tamaño de la ventana y llama al método `refresh()` para calcular y actualizar las dimensiones del tablero. Establecemos un tamaño mínimo de ventana con `setMinimumSize()` para evitar que los elementos se oculten al reducir demasiado la ventana. Usamos `Layout Managers` como `BorderLayout` para organizar el panel principal, `GridLayout` para distribuir las celdas del tablero y `GridBagLayout` para centrar botones como el de "Start". Además, calculamos dinámicamente el tamaño de las celdas del tablero en función del tamaño del panel, asegurando un mínimo de 10 píxeles por celda. Finalmente, aplicamos `PreferredSize` a los componentes y configuramos bordes y márgenes con `BorderFactory` para mantener un diseño consistente y adaptativo. Estos elementos trabajan juntos para garantizar que la interfaz sea funcional y visualmente agradable en cualquier tamaño de ventana.

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

```

JPanel leftPanel = new JPanel(new GridBagLayout());
leftPanel.setPreferredSize(new Dimension(150, 0)); // Dar u

JButton startButton = new JButton("Start");
startButton.setFont(new Font("Arial", Font.BOLD, 16));
startButton.setPreferredSize(new Dimension(100, 40));
startButton.addActionListener(e -> toggleStart());

// Usar GridBagConstraints para centrar el botón
GridBagConstraints gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.anchor = GridBagConstraints.CENTER;
leftPanel.add(startButton, gbc);

```

```

private void refresh() {
    boardPanel.removeAll();
    boardPanel.setLayout(new GridLayout(rows, cols));

    // Calcular el tamaño de cada celda basado en el tamaño d
    Dimension boardSize = boardPanel.getSize();
    int cellWidth = Math.max(boardSize.width / cols, 10);
    int cellHeight = Math.max(boardSize.height / rows, 10);

    for (int row = 0; row < rows; row++) {
        for (int col = 0; col < cols; col++) {
            Square cell = null;

            if (isHole(row, col)) {
                cell = new Hole(row, col);
            } else if (isWall(row, col)) {
                cell = new Square(wallColor);
            } else if (isParticle(row, col)) {
                cell = getParticleAt(row, col);
            } else {
                cell = new Square(emptyColor);
            }

            if (cell != null) {
                cell.setBorder(new LineBorder(gridColor));
                cell.setPreferredSize(new Dimension(cellWidth
                boardPanel.add(cell);
            }
        }
    }
}

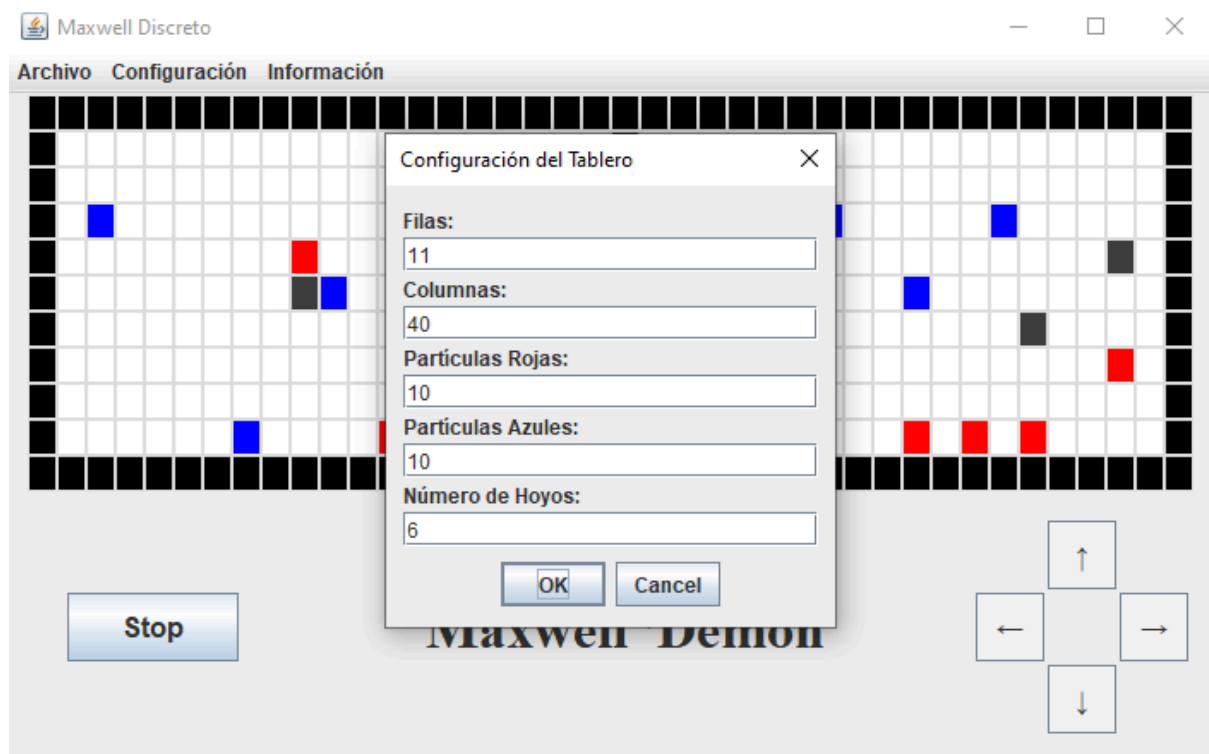
```

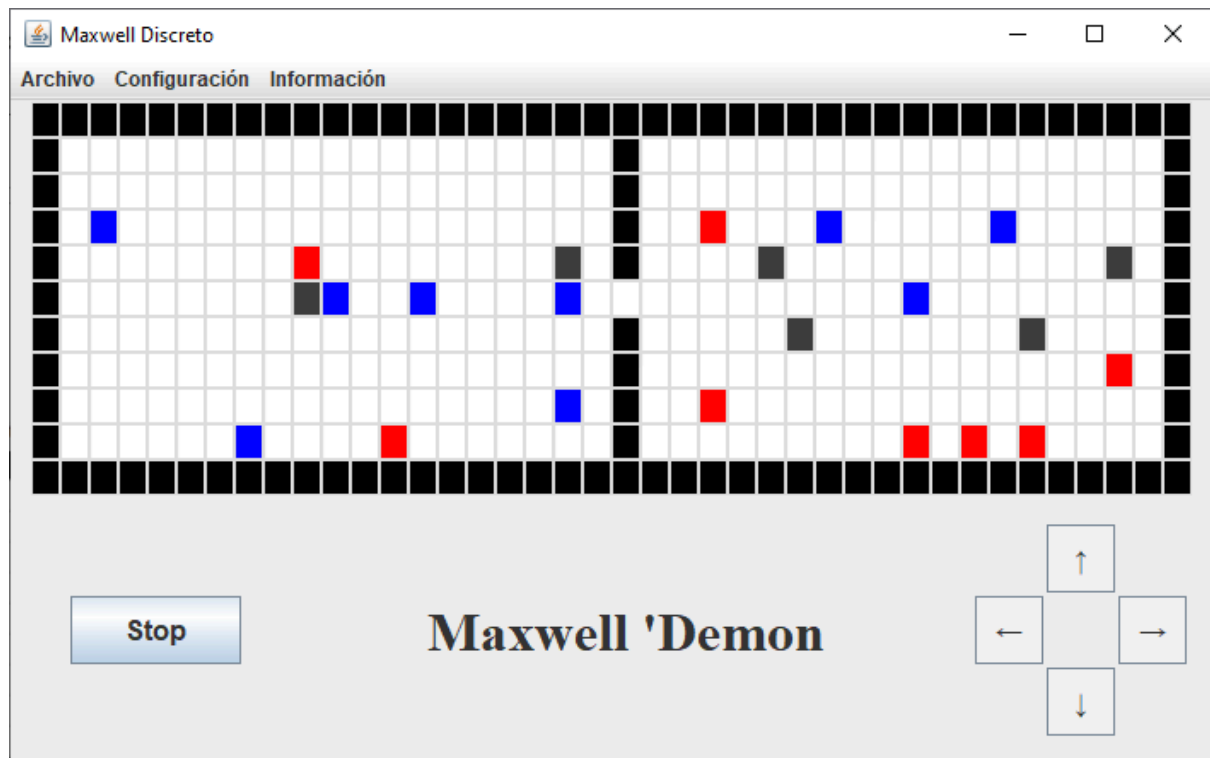
```
private void setupFrame() {
    setTitle(WINDOW_TITLE);
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            exitApplication();
        }
    });

    // Agregar ComponentListener para manejar el redimensionamiento
    addComponentListener(new java.awt.event.ComponentAdapter() {
        public void componentResized(java.awt.event.ComponentEvent evt) {
            refresh();
        }
    });

    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    setMinimumSize(new Dimension(600, 400)); // Establecer tamaño mínimo
    setLocationRelativeTo(null);
}
```

3. Ejecuten el caso de uso y capture las pantallas más significativas.
Dimensiones del tablero inicialmente





Click en configuración y en Modificar Tablero

Configuración del Tablero

Filas:

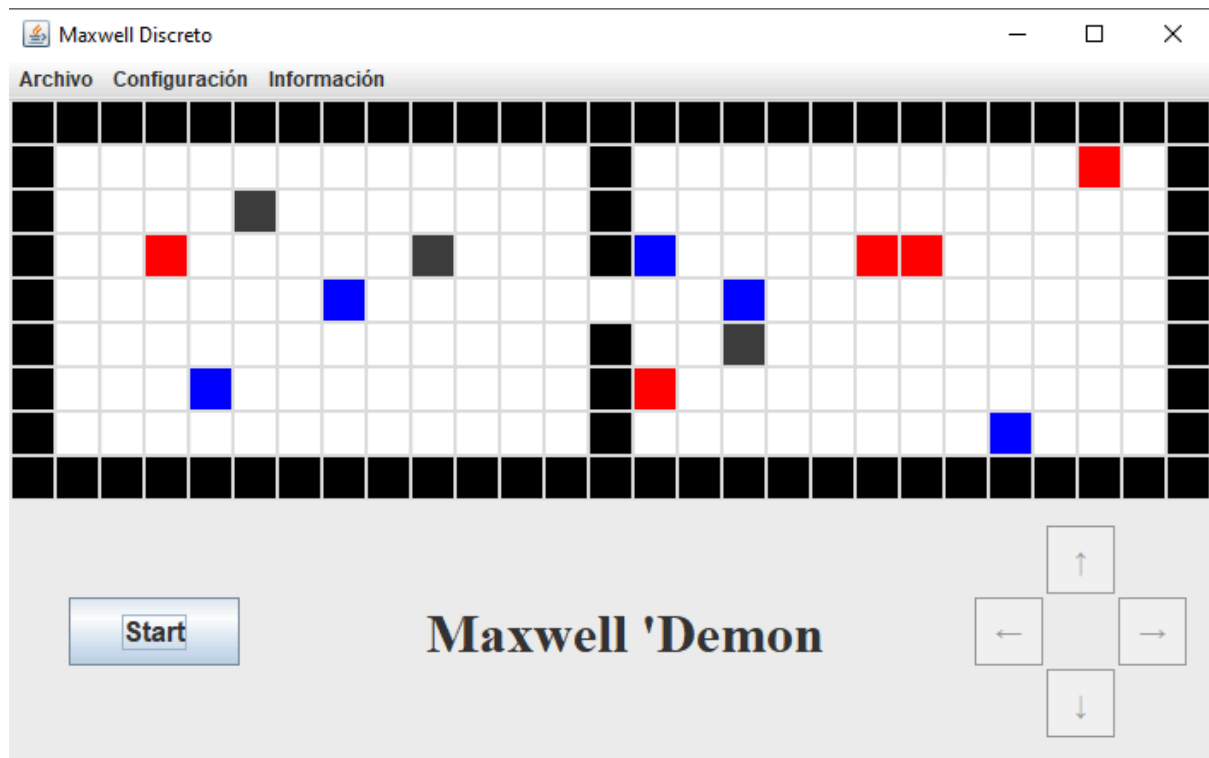
Columnas:

Partículas Rojas:

Partículas Azules:

Número de Hoyos:

OK Cancel



RETROSPECTIVA

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

Christian Romero: 10 horas

Anderson Garcia: 10 horas

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

El laboratorio se encuentra completo y funcional con cada uno de los requerimientos, esto se debe al compromiso y dedicación, además del trabajo en grupo y comunicación proactiva

3. **Considerando las prácticas XP del laboratorio. ¿Cuál fue la más útil? ¿Por qué?**

Una de las prácticas más valiosas fue la implementación de pruebas y la programación en pareja (pair programming). Esto se debe a que, al momento de escribir código, es común cometer errores que no siempre son evidentes para quien los introduce, ya que cuesta mantener una visión objetiva sobre la lógica implementada. Contar con otra persona que revise el código no solo permite detectar fallos más fácilmente, sino que también enriquece el proceso con diferentes perspectivas y conocimientos.

4. **¿Cuál consideran fue el mayor logro? ¿Por qué?**

Consideramos que el mayor logro fue sacar el laboratorio en la mitad del tiempo que los anteriores, esto debido a que fue un gran aporte de ambos, aparte de mantenernos concentrados y con la menor cantidad de distracciones a la hora de realizar el laboratorio.

También se podría decir que la entrega anticipada del laboratorio y no el último día de plazo, debido a que tratamos de aprovechar las vacaciones para adelantar el laboratorio y no tener que correr a lo último.

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

Nuestro mayor problema técnico fue implementar el movimiento correcto de las partículas, pues estas estaban condicionadas al entorno, reglas de juego y a la ubicación de las otras partículas para asegurar un funcionamiento adecuado. específicamente al momento de mover las partículas y quedar en el borde, se solapaba, perdiendo la visibilidad de las partículas. Para resolverlo tuvimos que indagar validaciones implementadas en juegos como tetris entre otros juegos que se enfrentan a problemáticas similares, lo anterior por medio de git, pues nos dio ideas para codificar la lógica que nos llevó al correcto y esperado funcionamiento del juego

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

Como equipo nos destacamos en el trabajo colaborativo, nos permitió terminar todo de forma adecuada, sin embargo podemos mejorar, por lo que nos comprometemos a implementar herramientas que nos permitan tener acceso al código en tiempo real.

7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.

JFileChooser (Java Platform SE 8). (2025, April 5).
<https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html>

How do I refresh a GUI in Java? (n.d.). Stack Overflow.
https://stackoverflow.com/questions/12865803/how-do-i-refresh-a-gui-in-java?utm_source

How to Reset JPanel On Click? (Swing / AWT / SWT forum at Coderanch). (n.d.). https://coderanch.com/t/752570/java/Reset-JPanel-Click?utm_source

Components disappear on resize. . . (2003, October 1). Oracle Forums.
<https://forums.oracle.com/ords/apexds/post/components-disappear-on-resize-2046>