

PROGRAMACIÓN ORIENTADA A OBJETOS

Interfaz

Laboratorio 5/6

Integrantes:

Daniel Useche

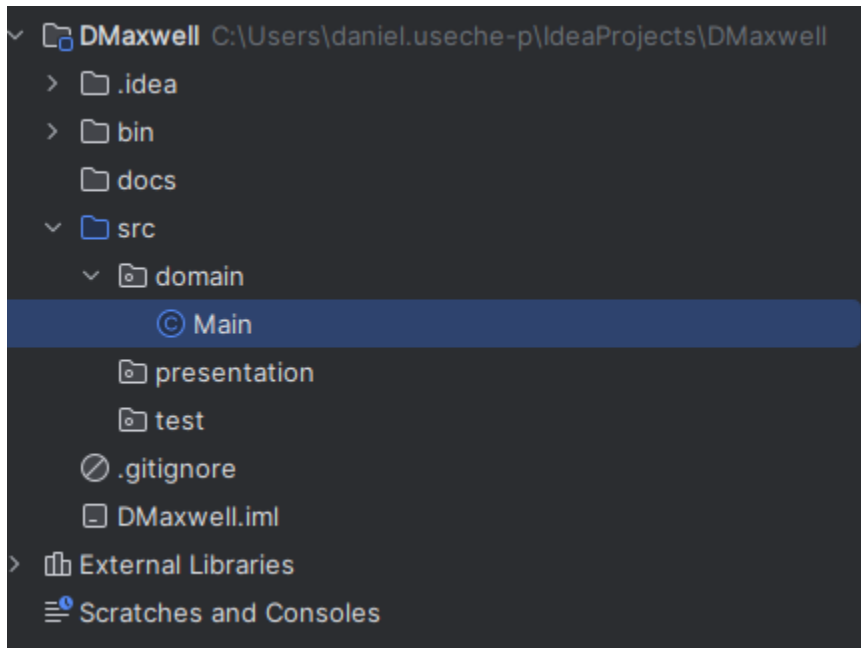
Daniel Patiño

DESARROLLO

Directorios

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

3. Preparen un directorio llamado DMaxwell con los directorios src y bin y los subdirectorios para presentación, dominio y pruebas de unidad. Capturen un pantalla con la estructura.

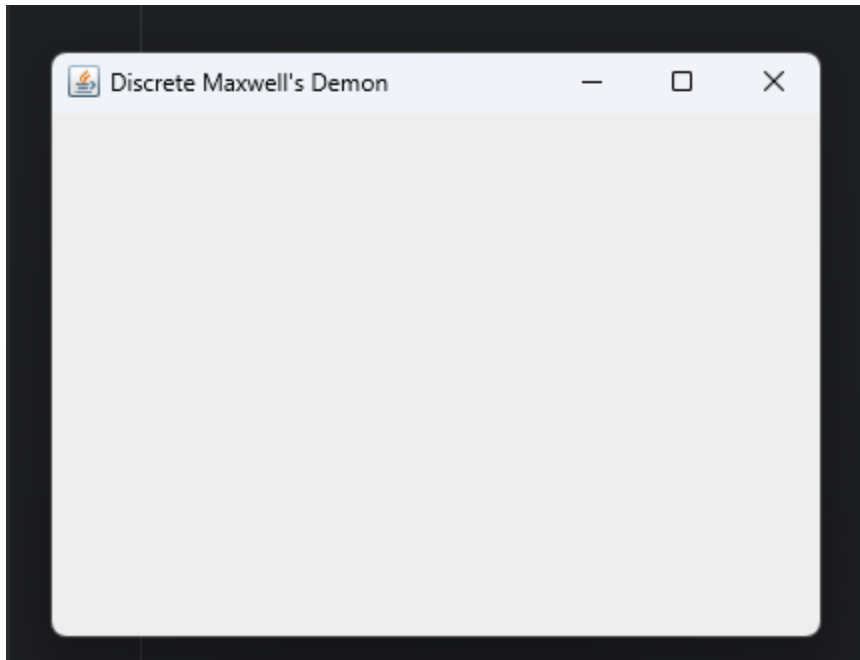


Ciclo 0: Ventana vacía – Salir

[En *.java y lab05.doc]

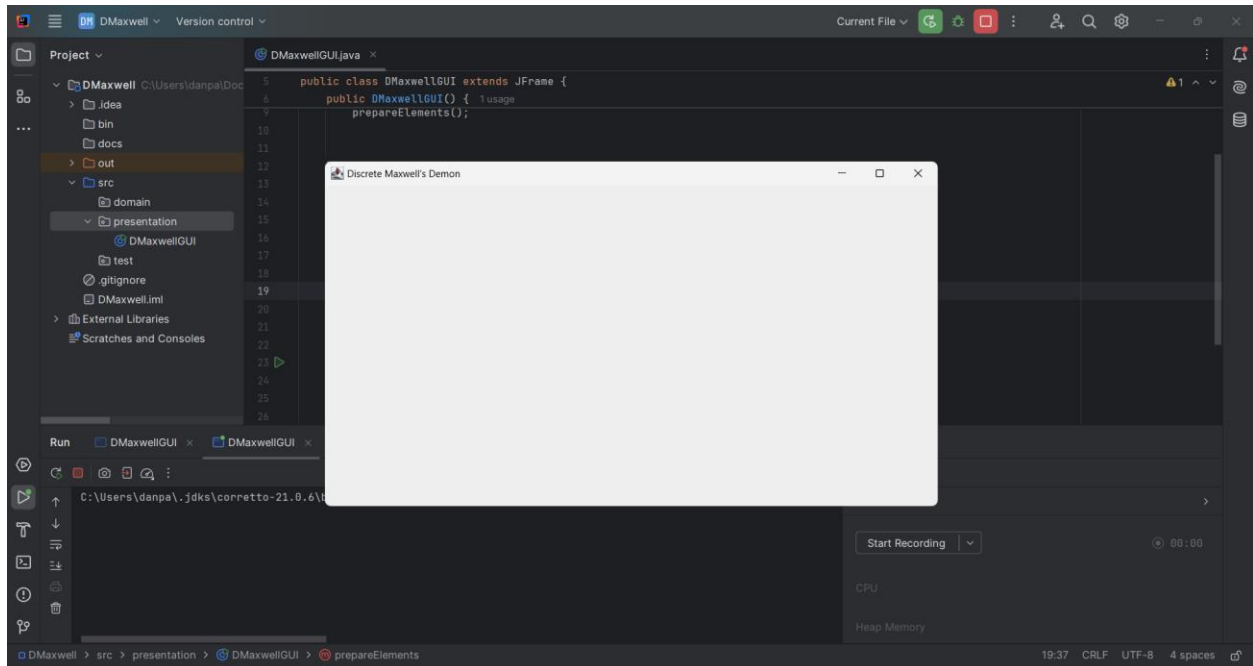
El objetivo es implementar la ventana principal de DMaxwell con un final adecuado desde el icono de cerrar. Utilizar el esquema de prepareElements-prepareActions.

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.



(Si la ventana principal no es la inicial en su diseño, después deberán mover el main al componente visual correspondiente)

2. Modifiquen el tamaño de la ventana para que ocupe un cuarto de la pantalla y ubíquela en el centro. Para eso inicien la codificación del método prepareElements. Capturen esa pantalla.



3. Traten de cerrar la ventana.

a. Termina la ejecución?

Al ejecutar el programa y tratar de cerrar la ventana, el programa no deja de ejecutarse, por lo que, si lo tratamos de volver a ejecutar, aparecerá que el programa se sigue ejecutando.

b. ¿Qué deben hacer en consola para terminar la ejecución?

Para terminar la ejecución desde consola debemos presionar Ctrl + C, así estamos forzando a finalizar la ejecución, este comando funciona tanto para windows, linux y macOS.

4. Estudien en JFrame el método setDefaultCloseOperation.

¿Para qué sirve?

El método sirve para definir qué debe hacer el programa cuando el usuario cierra la ventana.

Se implementa de la siguiente manera:

```
Gui.setDefaultCloseOperation(Aqui se coloca la operacion);
```

¿Cómo lo usarían si queremos confirmar el cierre de la aplicación?

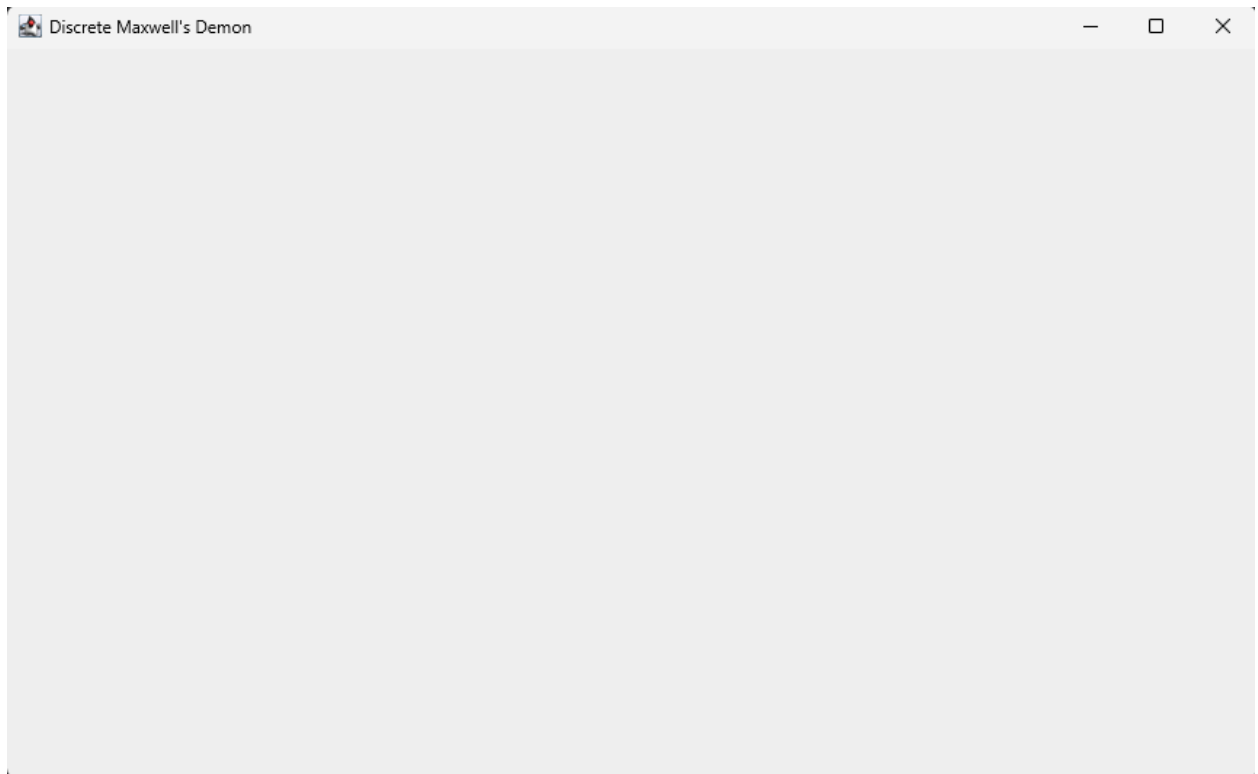
Para confirmar el cierre de la aplicación, usamos el `gui.setDefaultCloseOperation` y utilizamos la operación `JFrame.DO_NOTHING_ON_CLOSE`, esto con el fin de que el programa no se cierre solo al darle a la x de la ventana, luego, agregamos un oyente para que cuando se intente cerrar la aplicación, arroje un mensaje para confirmar si se quiere salir.

¿Cómo lo usarían si queremos simplemente cerrar la aplicación?

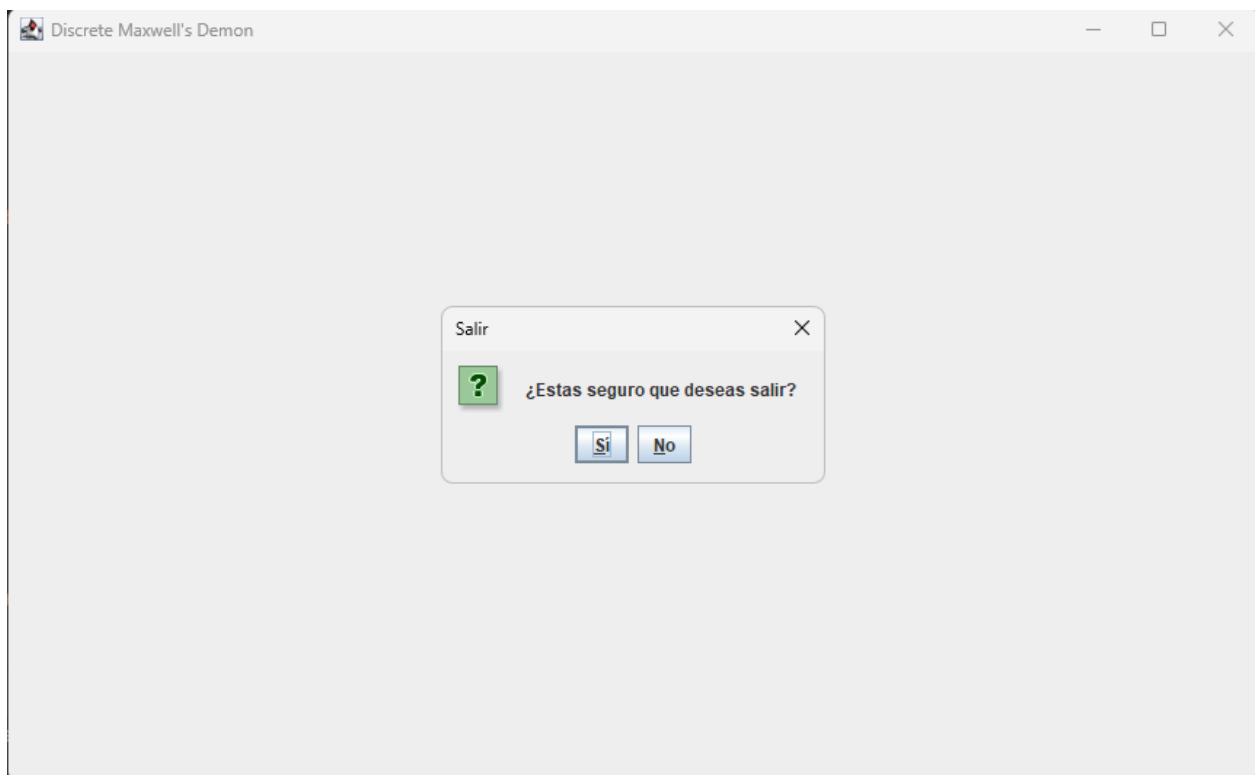
Para simplemente cerrar la aplicación, usamos el `gui.setDefaultCloseOperation` con la operación `EXIT_ON_CLOSE`.

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.

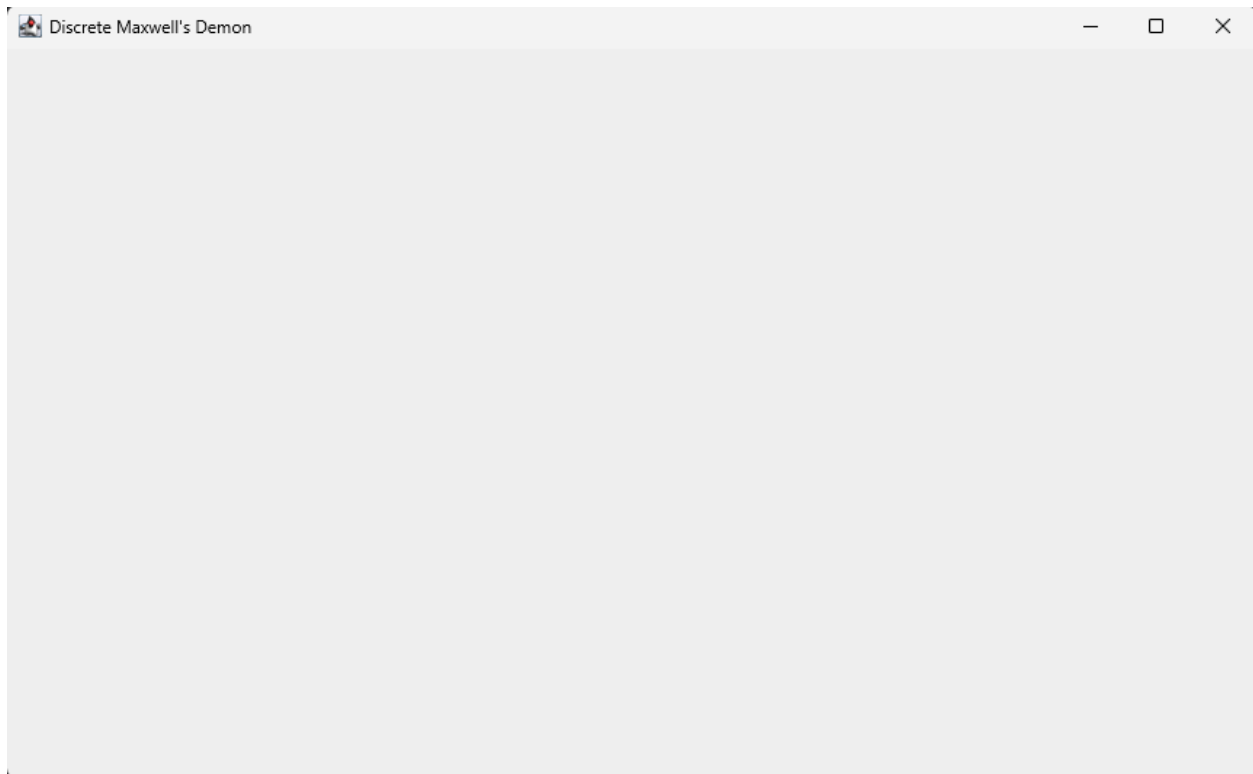
Ejecutando el programa:



A continuacion le daremos a la x:



Si le damos no, volvemos al programa normal:



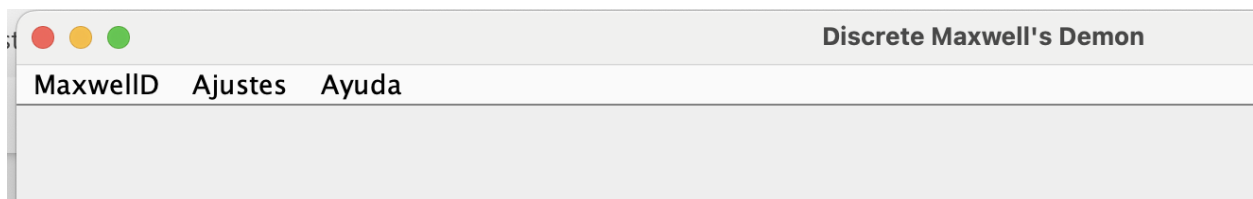
Si le damos que si se cierra totalmente.

Ciclo 1: Ventana con menú – Salir

[En *.java y lab05.doc]

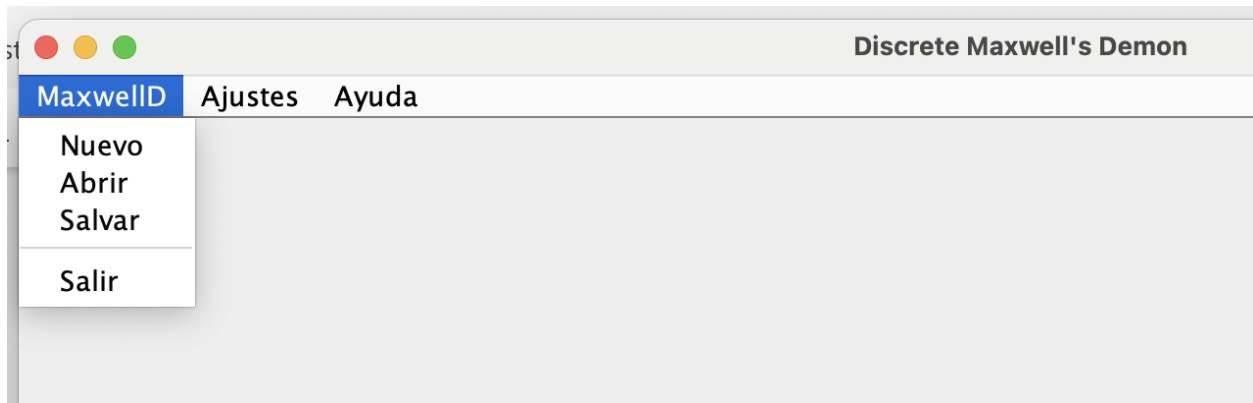
El objetivo es implementar un menú clásico para la aplicación con un final adecuado desde la opción del menú para salir. El menú debe ofrecer mínimo las siguientes opciones: Nuevo, Abrir – Salvar y Salir . Incluyan los separadores de opciones.

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



Integramos 3 Items al menu, que luego anclamos al Panel que esta en el Frame, así tenemos la estructura anterior.

Además, integramos las funcionalidades que eran requeridas:



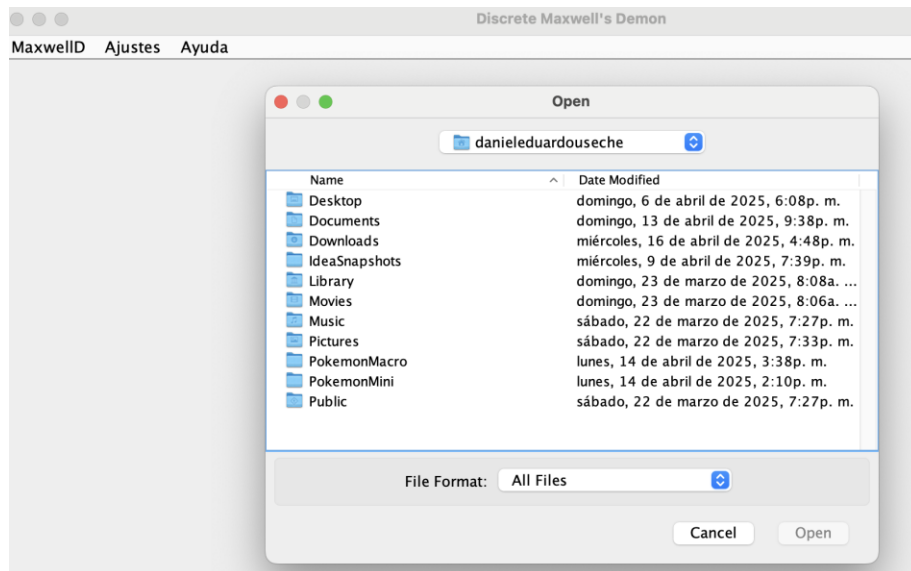
cada una funcional en su natural descripción

Se crea un JMenuBar y un JMenu el cual podría contener todos los elementos dichos anteriormente, los atributos serán los tipos de elementos, si son botones, menus, listas, etc y las variables son los constructores de los elementos que vayamos a utilizar.

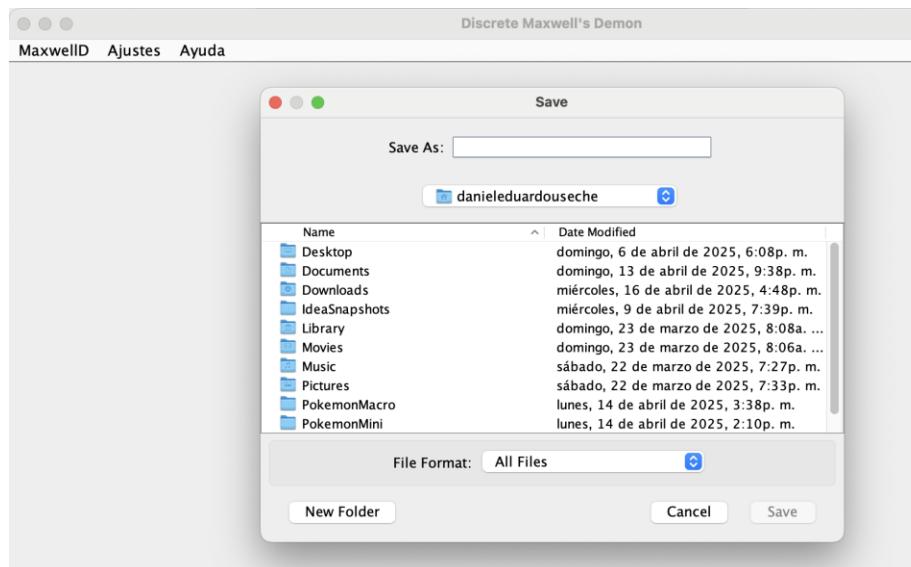
2. Construya la forma del menú propuesto (prepareElements - prepareElementsMenu) .

Ejecuten. Capturen la pantalla.

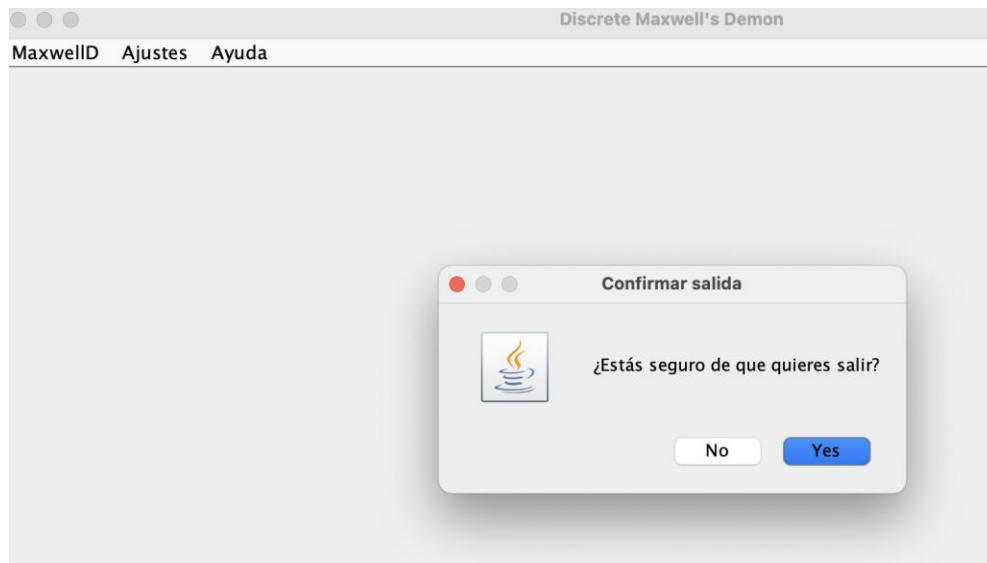
Abrir



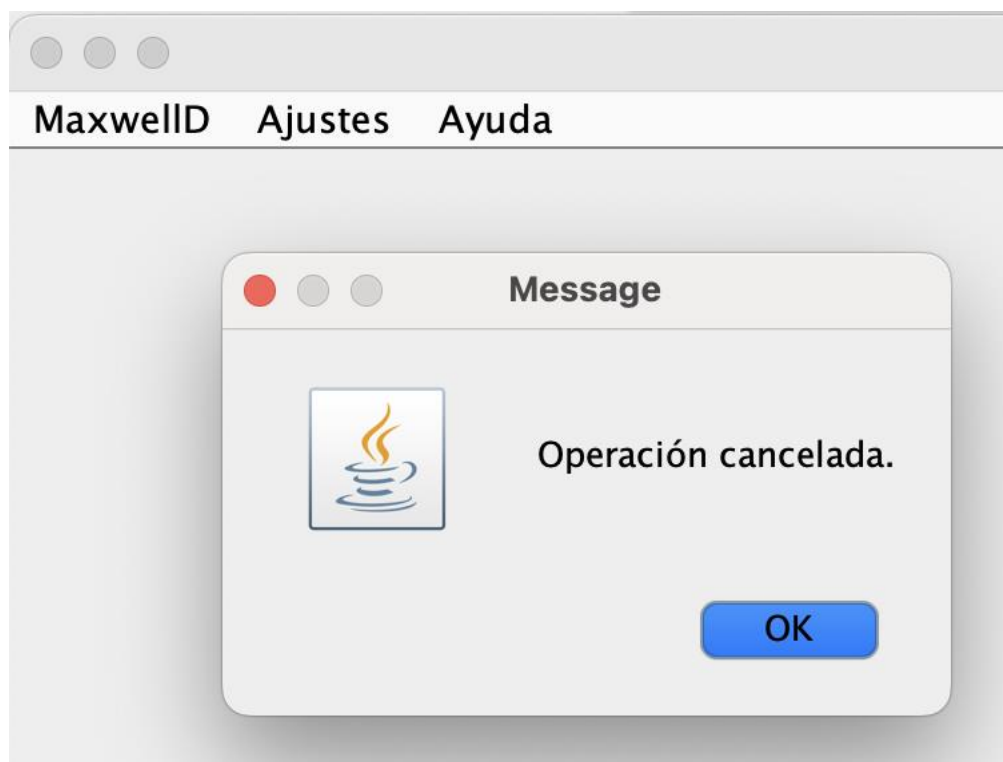
Salvar



Salir

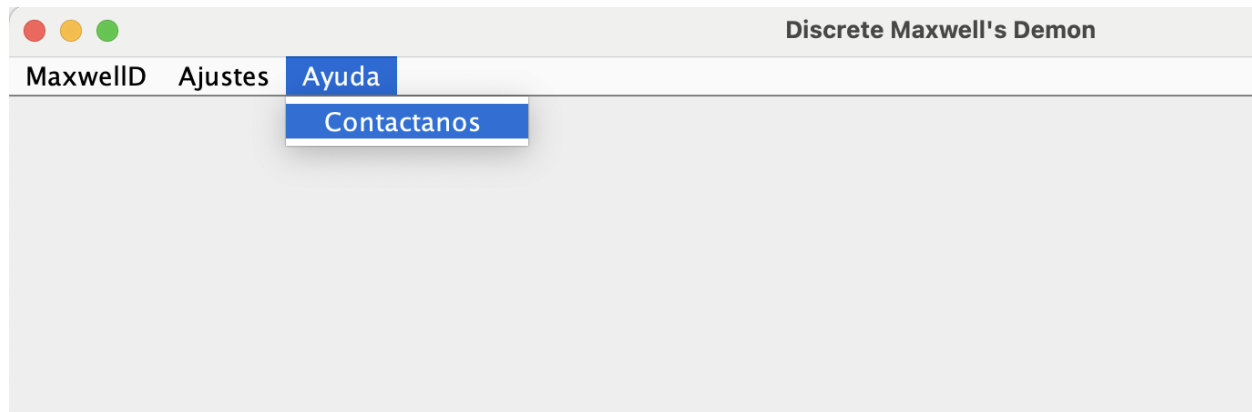


Además, cada elemento cuenta con un mensaje en caso de cancelación de dicha acción

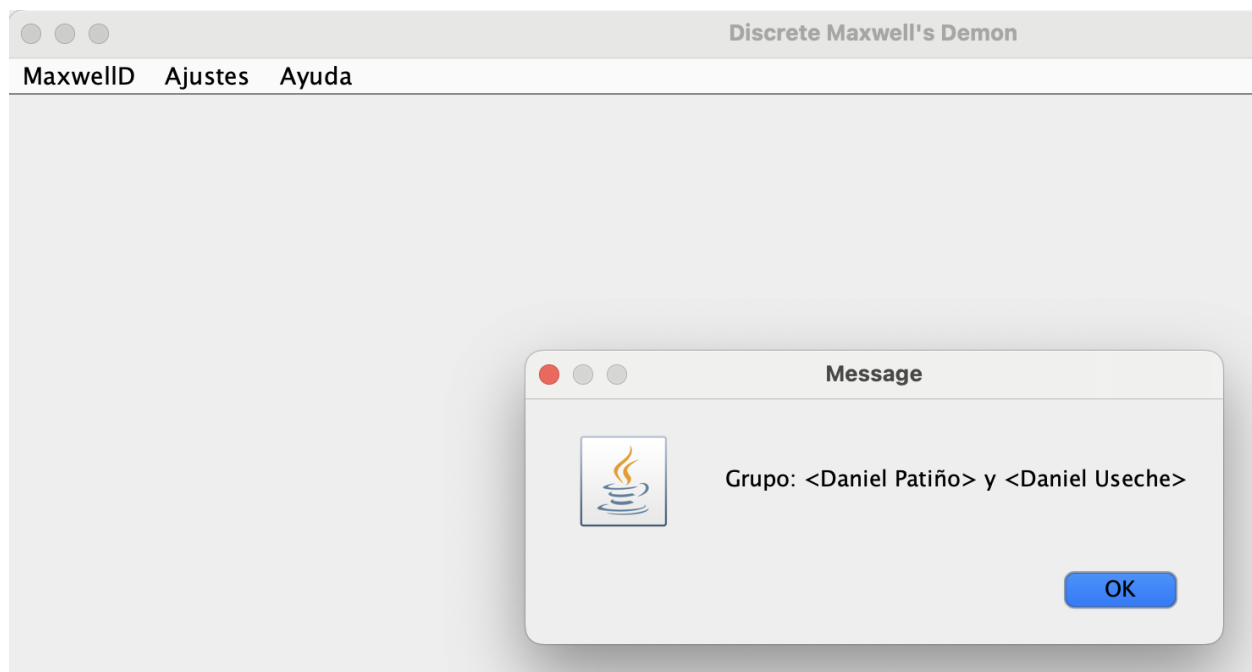


Extras como Opciones se integrará al juego para el ajuste de tamaño y otras opciones del Juego.

Por último un Ayuda que genera un mensaje con el Grupo de POOB que realizó este Proyecto

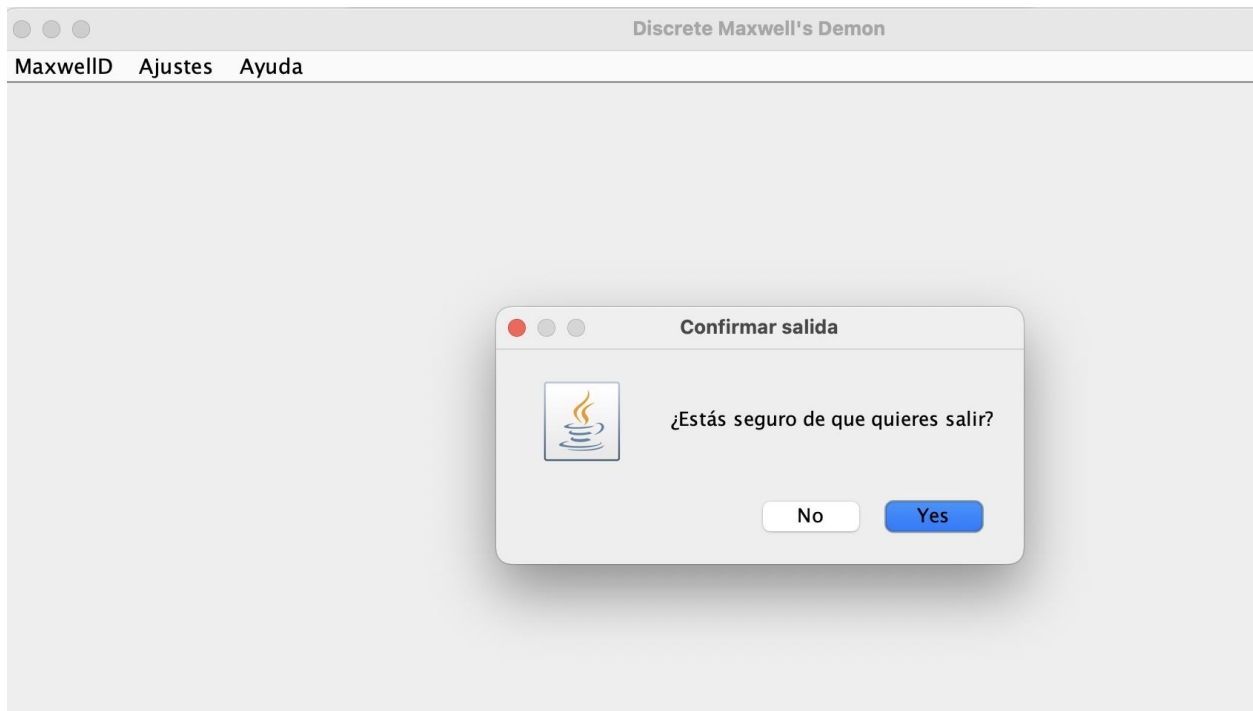


Así:



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

Desde el Menú



Una vez ejecutado:

```
/Library/Java/JavaVirtualMachines/jdk-24.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA
2025-04-17 13:44:08.474 java[40466:3544902] +[IMKClient subclass]: chose IMKClient_Modern
2025-04-17 13:44:08.474 java[40466:3544902] +[IMKInputSession subclass]: chose IMKInputSession_Modern

Process finished with exit code 0
```

Ciclo 2: Salvar y abrir

[En *.java y lab05.doc]

El objetivo es preparar la interfaz para las funciones de persistencia

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

-JFileChooser: Nos permite abrir un cuadro de dialogo donde el usuario puede seleccionar archivos o carpetas a través de una ventana grafica estándar. El constructor JFileChooser() crea un selector de archivos el cual puede recibir una ruta inicial predeterminada. Por ejemplo “C:/Downloads”.

-ShowOpenDialog: Su función es mostrar una ventana emergente para que el usuario abra el archivo existente. Su funcionamiento se basa en devolver un valor entero: APPROVE_OPTION si el usuario selecciona un archivo y confirma, o CANCEL_OPTION si cancela.

-ShowSaveDialog: Muestra una ventana emergente para que el usuario pueda guardar el archivo. Funciona igual que ShowOpenDialog, devuelve APPROVE_OPTION si el usuario confirma, o CANCEL_OPTION si cancela.

-GetSelectedFile: Su función es retornar un objeto File el cual representa el archivo seleccionado por el usuario despues de usar ShowOpenDialog o ShowSaveDialog. El metodo GetSelectedFile solo debe ser llamado si el usuario aceptó en los metodos mencionados anteriormente.

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.

Se implementan un método para salvar la partida saveGame() en el cual se llama al crea el constructor de un JFileChooser donde si se confirma la opción de guardar la partida actual la guarda en un File en la ubicación que el usuario desee.

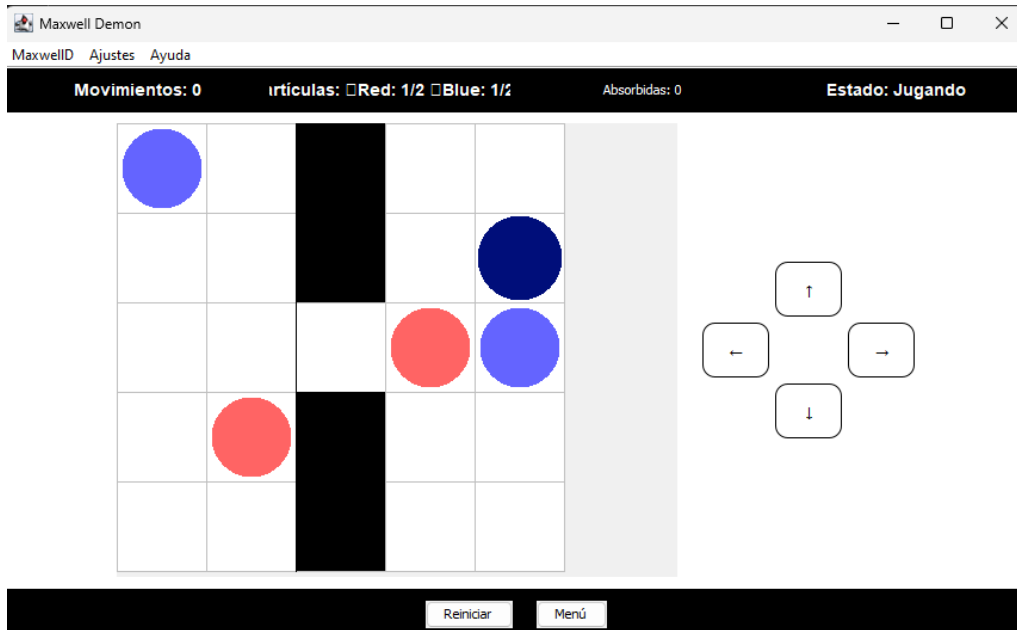
```
/**
 * Método para guardar la partida actual (simulación)
 */
private void saveGame() { 1 usage
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showSaveDialog( parent: this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        JOptionPane.showMessageDialog( parentComponent: this,
            message: "Partida guardada (simulación).\nArchivo: " + selectedFile.getName());
    }
}
```

Y el metodo de openGame() donde se vuelve a crear un constructor de JFileChooser donde se crea una variable (result) el cual nos abra una ventana para elegir el archivo, si el usuario aprueba la accion, se carga la partida donde se haya dejado, de hacer lo contrario no pasara nada.

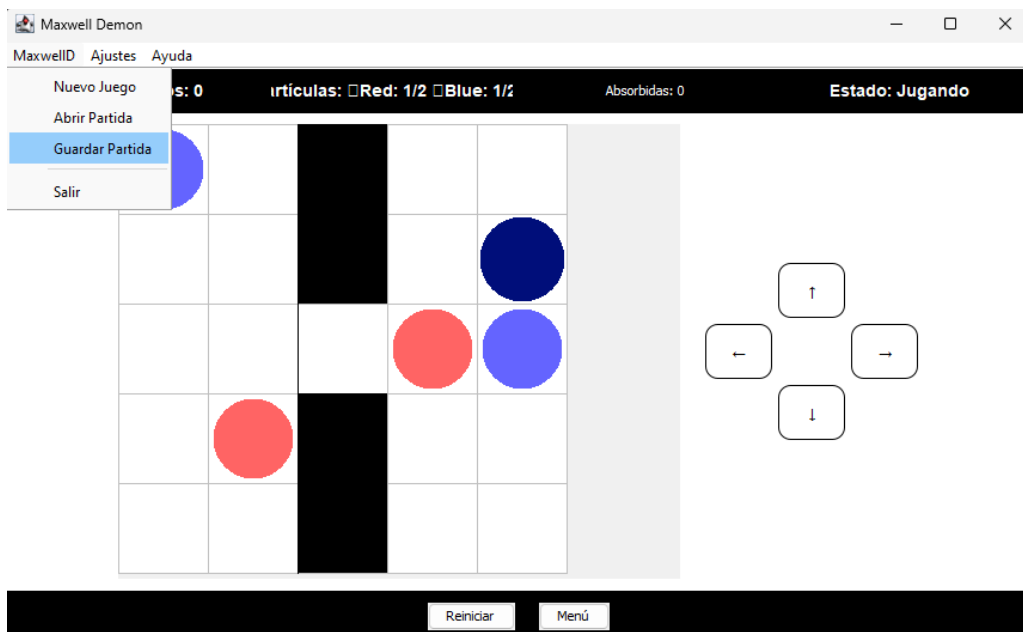
```
/**
 * Método para abrir una partida guardada (simulación)
 */
private void openGame() { 1 usage
    JFileChooser fileChooser = new JFileChooser();
    int result = fileChooser.showOpenDialog( parent: this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        JOptionPane.showMessageDialog( parentComponent: this,
            message: "Partida cargada (simulación).\nArchivo: " + selectedFile.getName());
    }
}
```

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

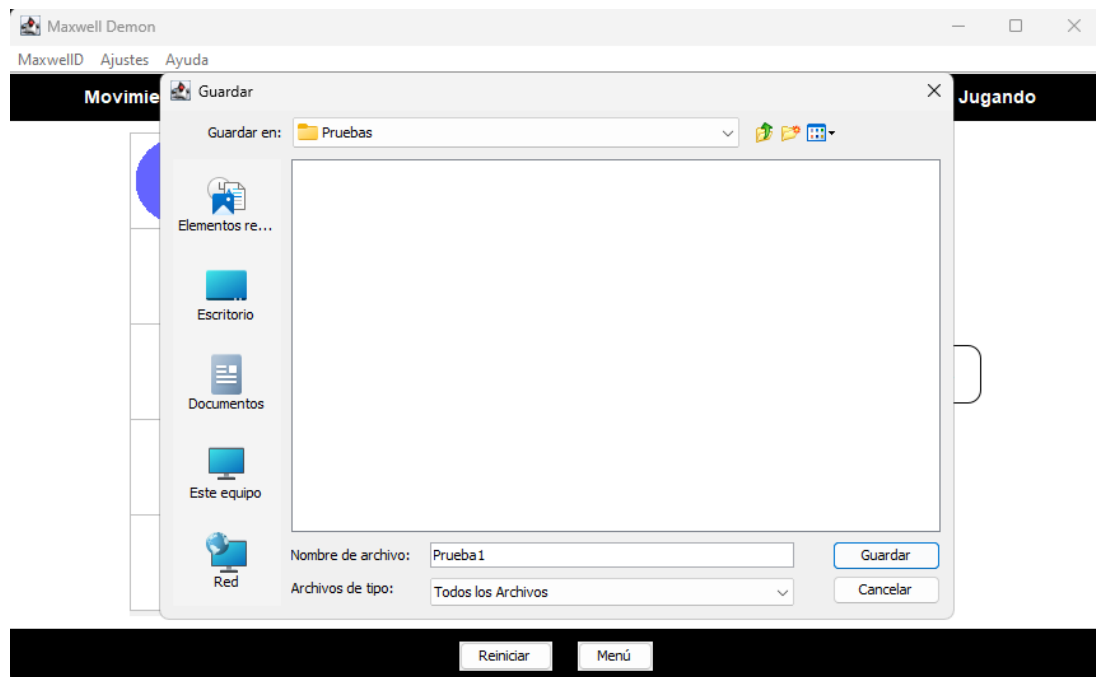
Creamos un juego nuevo:



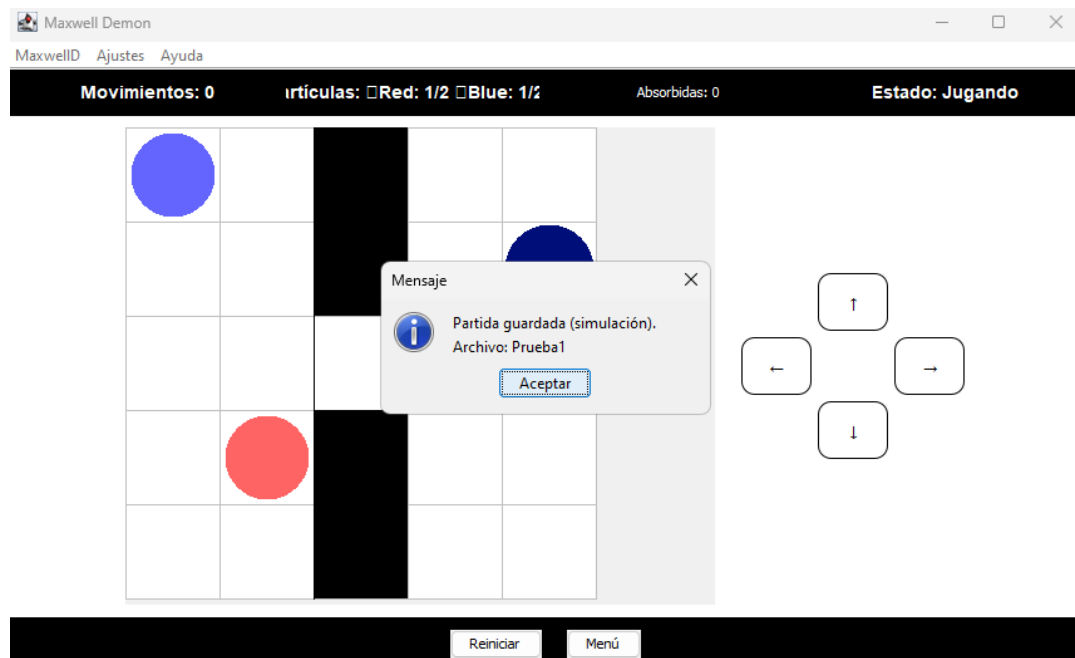
Le damos en MaxwellID y le damos en guardar partida:



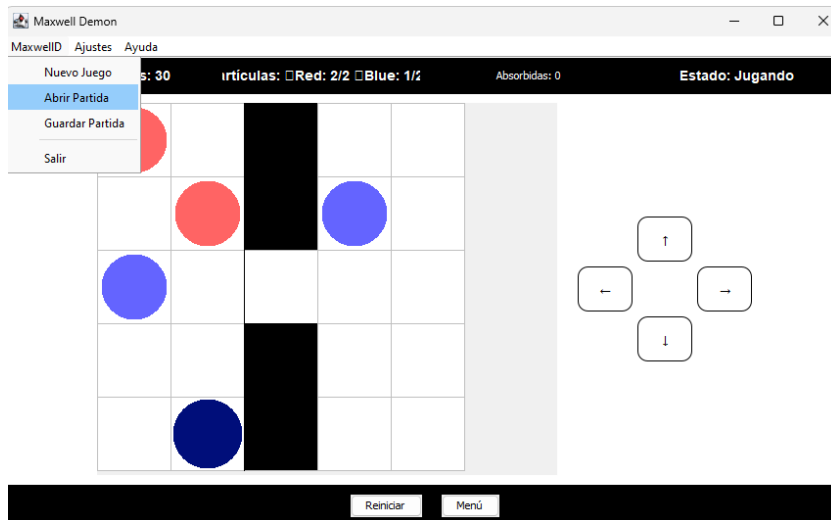
Seleccionamos la dirección donde se quiere guardar el archivo y el nombre:



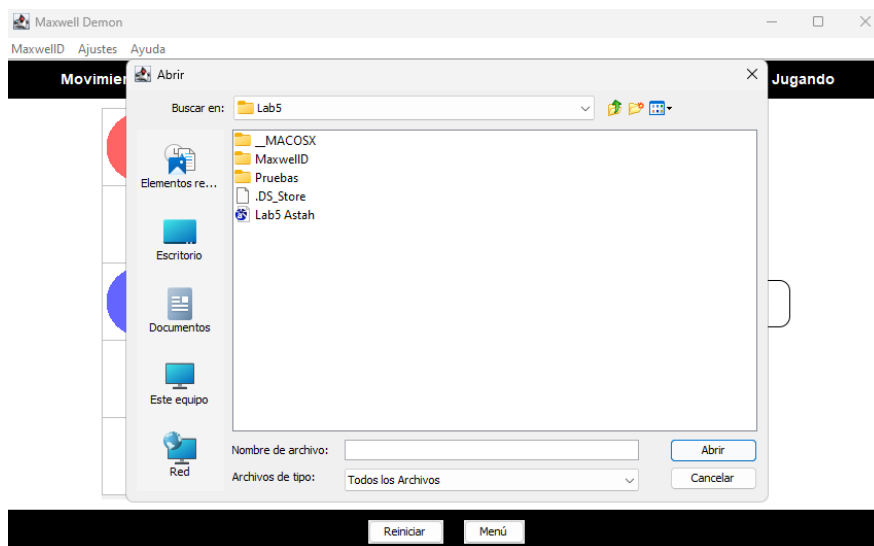
Finalmente lo guardamos y verificamos que el archivo se encuentre allí:



Ahora para abrir un archivo seleccionamos en abrir partida:



Y seleccionamos el archivos deseado.



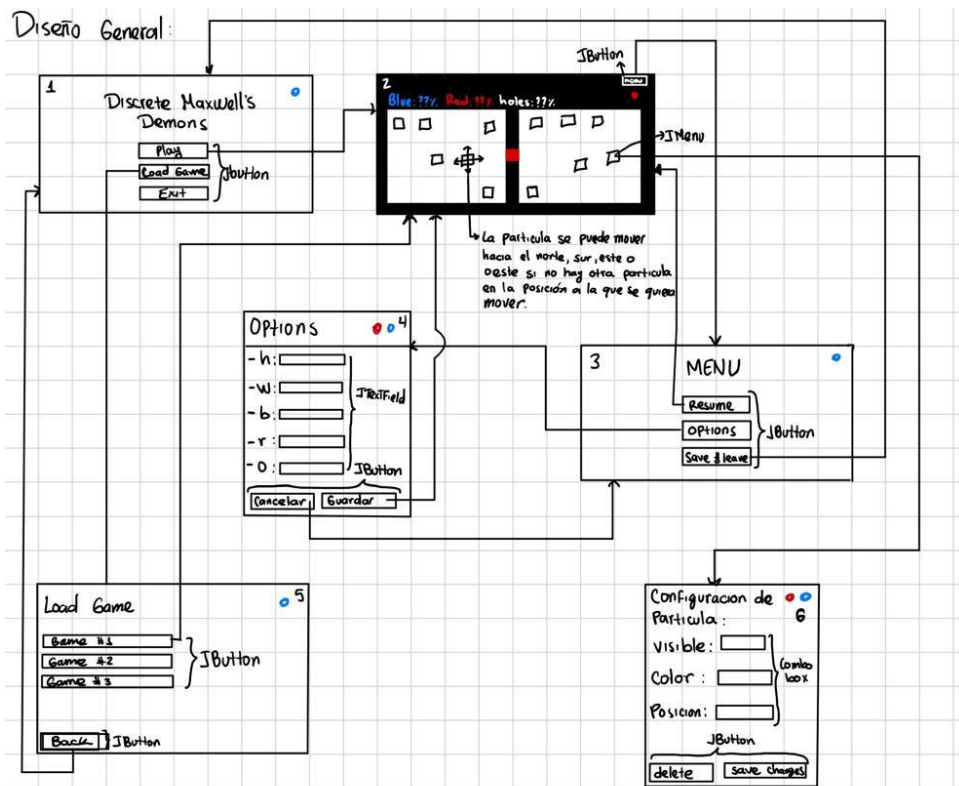
Ciclo 3: Forma de la ventana principal

[En *.java y lab05.doc]

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)



- Programados
- a programar
- Faltó configurarlo (GUI)

1. mainMenu
2. inGameGUI
3. pauseMenu
4. boardMenu
5. loadMatchMenu
6. particleOptions

Para mainMenu

mainMenuEvent:

mainMenuListener:

- jugarButton: Metodo para mostrar el juego al presionar el boton
- cargarPartidaButton: Metodo para desplegar el menu de cargar partida
- salirButton: Metodo para Salir del juego.

Para inGameGUI:

inGameGUIEvent:

inGameGUIListener:

- MenuButton: Metodo para desplegar el menu de pausa del juego.
- Particle button: Metodo para modificar la partícula
- MoveParticle: Metodo para mover la partícula (x)

Para PauseMenu:

PauseMenuEvent:

- resumeButton
- OptionsButton
- saveGameButton

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

Para la zona del tablero definan 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.

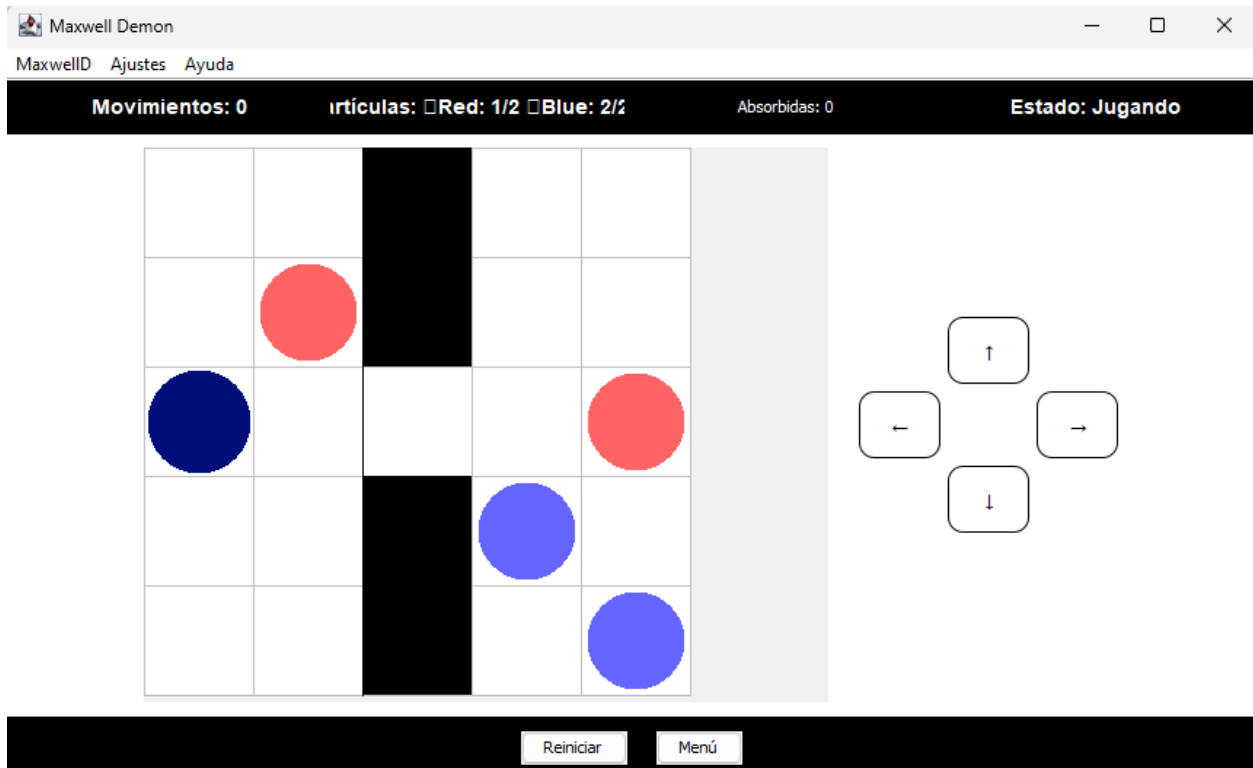
- **Revisar `DMaxwellGUI.java`**

3. Ejecuten y capturen la pantalla.



Jugar

Salir



Ciclo 4: Cambiar colores

[En *.java y lab05.doc]

El objetivo es implementar este caso de uso.

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

Primero implementamos un metodo (BoardPanel() (Vista)) el cual es una clase interna que representa el panel del tablero del juego, dentro de este se define el constructor del tablero donde implementaremos un JColorChooser (Vista) que es una ventana emergente la cual permite cambiar el color. Implementamos un mouseListener(Controlador) el cual controla los eventos del click, en este caso detecta si el usuario hace click en el tablero, boton o en este caso en la partícula y despliega el menu de cambio de color.

2. Detalle el comportamiento de JColorChooser especialmente el método estático ShowDialog

Es un componente de Swing que lanza una ventana emergente con un selector de colores. El metodo showDialog recibe tres parametros:

parent: Es la ventana sobre la cual se mostrará el dialogo.

title: Será el título de la ventana que en este caso es “Seleccionar color de la partícula”.

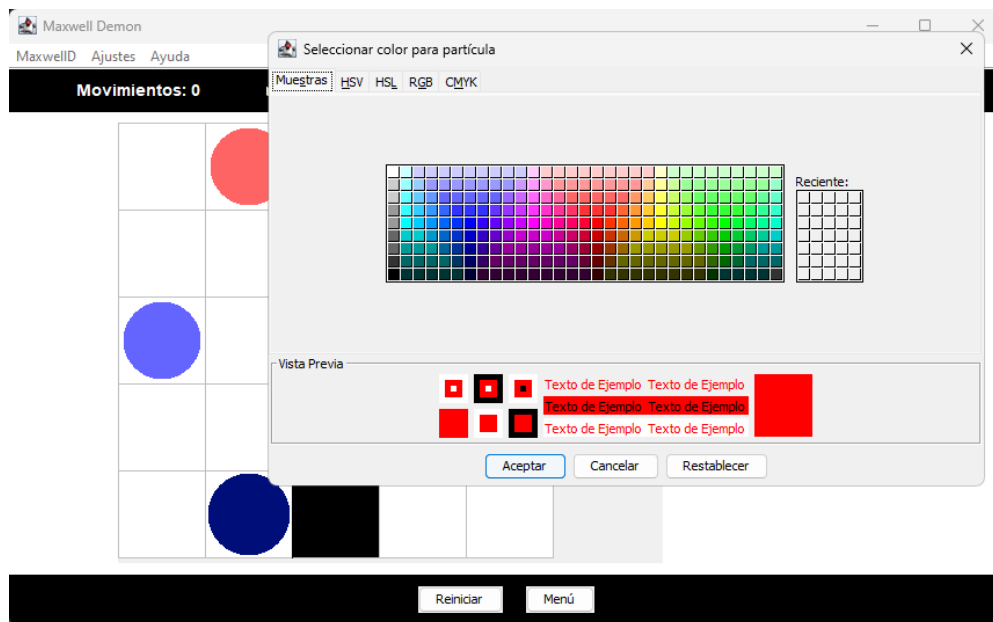
InitialColor: Es el color que aparece seleccionado inicialmente cuando se abre el cuadro si la partícula seleccionada es roja elegira el color rojo por defecto al abrirla y si es color azul, abra el color azul como por defecto.

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

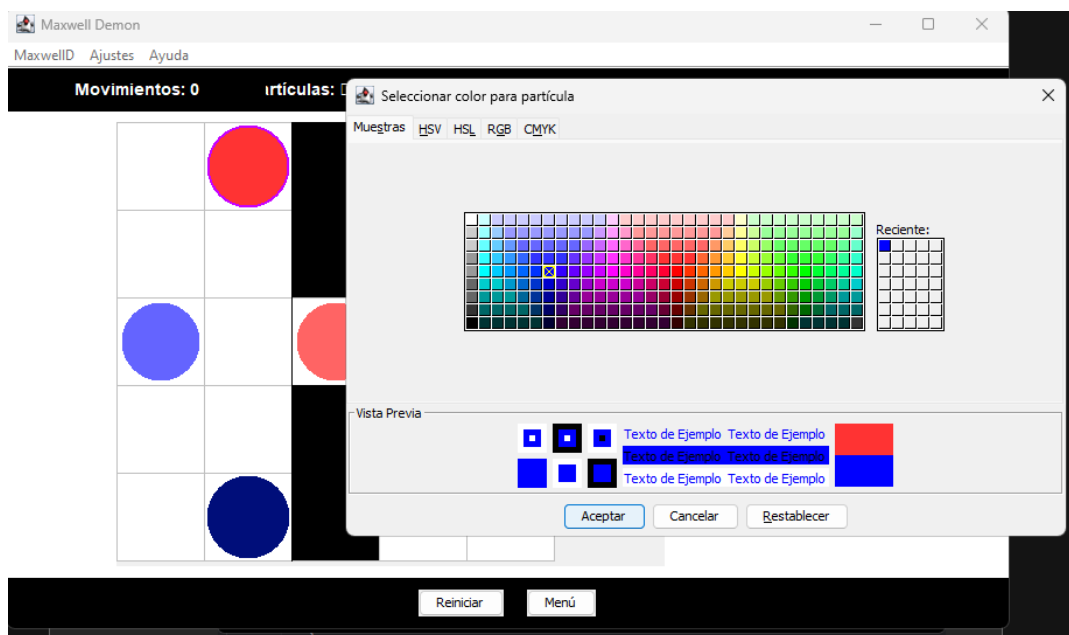
*** Revisar DMaxwellGUI.java**

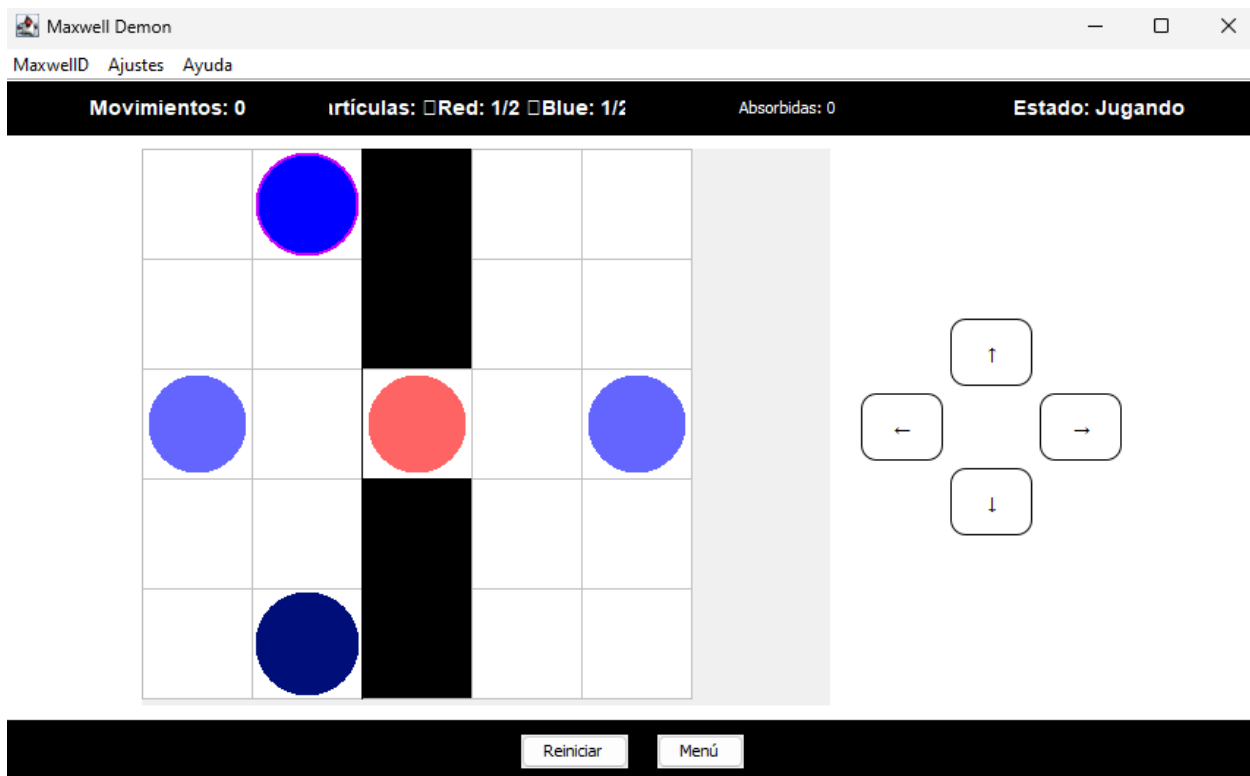
4. Ejecuten el caso de uso y capture las pantallas más significativas.

Seleccionando partícula roja:

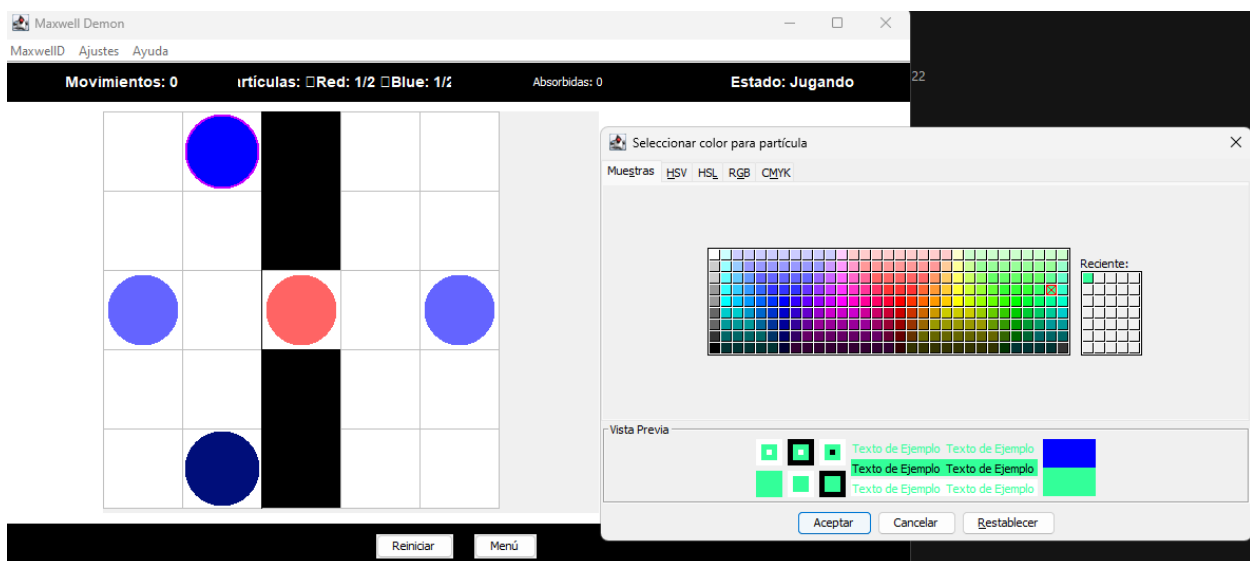


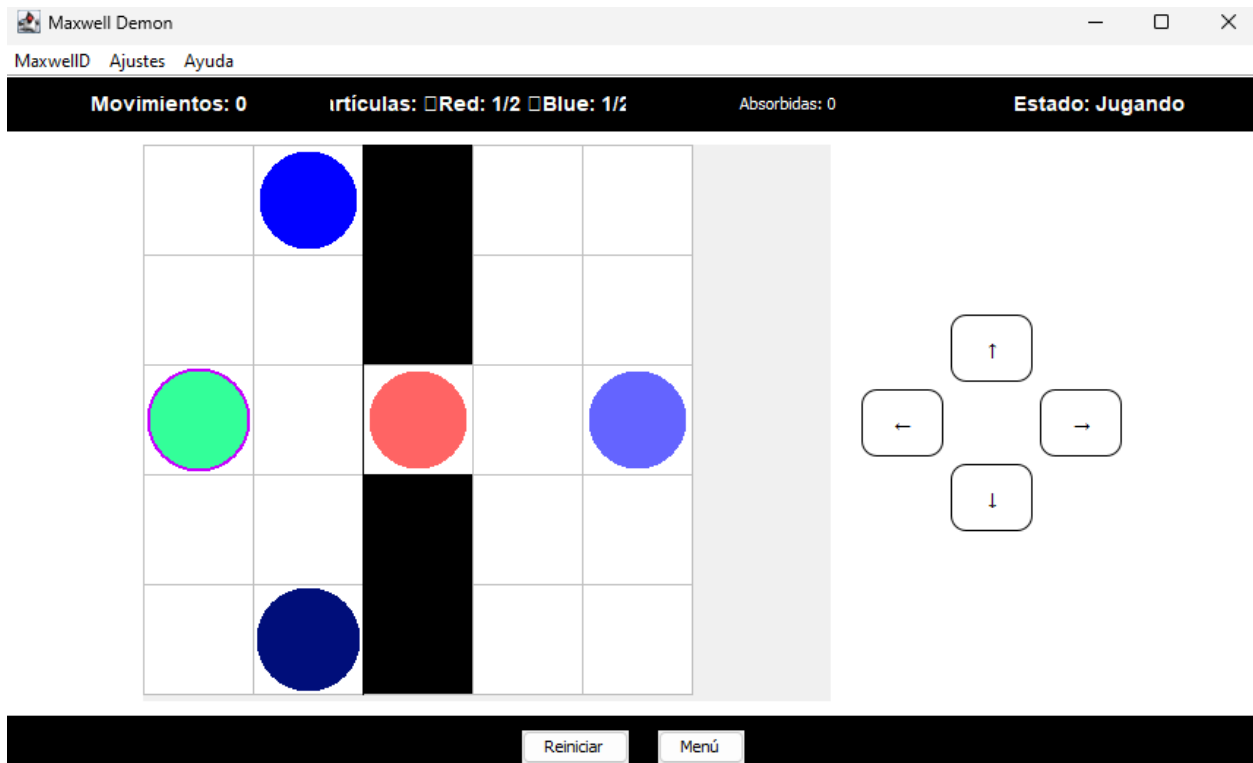
Elegimos el color azul y aplicamos los cambios:





Cambiándolo a color verde:





Ciclo 5: Modelo DMaxwell

[En *.java y lab05.doc]

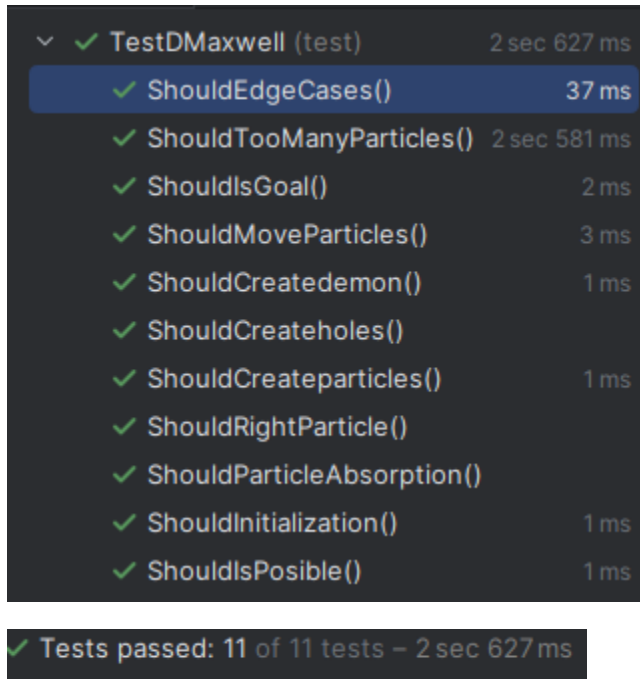
El objetivo es implementar la capa de dominio para DMaxwell

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

Revisar MaxwellD.java del paquete domain

En el archivo se encuentran los metodos basicos con la funcionalidad del juego, lo que hicimos fue adaptar lo que teniamos del proyecto inicial al juego actual omitiendo algunas cosas y agregando otras para que su funcionamiento sea adecuado.

2. Ejecuten las pruebas y capturen el resultado.



The screenshot shows a test runner interface with a dark background. At the top, a summary line reads '✓ TestDMaxwell (test) 2 sec 627 ms'. Below this, a list of 11 individual tests is displayed, each preceded by a green checkmark and followed by its execution time. The tests are: ShouldEdgeCases() (37 ms), ShouldTooManyParticles() (2 sec 581 ms), ShouldIsGoal() (2 ms), ShouldMoveParticles() (3 ms), ShouldCreatedemon() (1 ms), ShouldCreateholes() (1 ms), ShouldCreateparticles() (1 ms), ShouldRightParticle() (1 ms), ShouldParticleAbsorption() (1 ms), ShouldInitialization() (1 ms), and ShouldIsPossible() (1 ms). The 'ShouldEdgeCases()' test is highlighted with a blue background. At the bottom of the list, a summary bar states '✓ Tests passed: 11 of 11 tests – 2 sec 627 ms'.

Test Name	Duration
✓ TestDMaxwell (test)	2 sec 627 ms
✓ ShouldEdgeCases()	37 ms
✓ ShouldTooManyParticles()	2 sec 581 ms
✓ ShouldIsGoal()	2 ms
✓ ShouldMoveParticles()	3 ms
✓ ShouldCreatedemon()	1 ms
✓ ShouldCreateholes()	1 ms
✓ ShouldCreateparticles()	1 ms
✓ ShouldRightParticle()	1 ms
✓ ShouldParticleAbsorption()	1 ms
✓ ShouldInitialization()	1 ms
✓ ShouldIsPossible()	1 ms

✓ Tests passed: 11 of 11 tests – 2 sec 627 ms

Ciclo 6: Jugar

[En *.java y lab05.doc]

El objetivo es implementar el caso de uso jugar.

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

```
// Lógica del juego
private MaxwellD simulation; // Instancia del modelo del juego (el dominio)
```

Instanciamos en la lógica del juego bajo el nombre de simulation y aplicamos el funcionamiento del programa en el paquete de presentacion.

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

- **Revisar DMaxwellGUI.java**

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

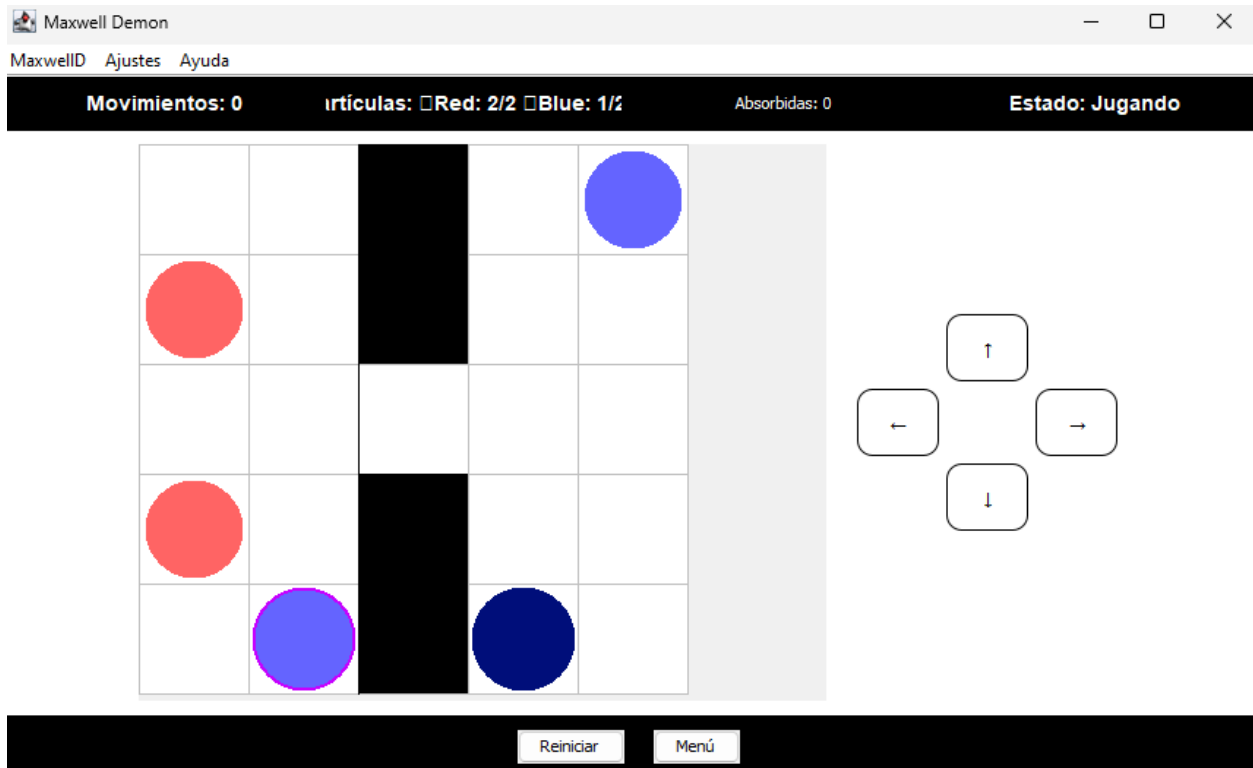
Para implementar este caso de uso debemos que tener primero la logica del programa, en este caso es nuestro archivo MaxwellD.java ya que contiene toda la jugabilidad del programa y contiene información de las particulas, demonio, tablero, etc. También necesitamos la vista que seria el BoardPanel, aqui implementamos lo mencionado anteriormente, y es la vista que va a tener el usuario a la hora de abrir el programa, tambien el MouseListener ya que es como se va a interactuar con el programa a la hora de de ejecutar.

4. Implementen los componentes necesarios para jugar .

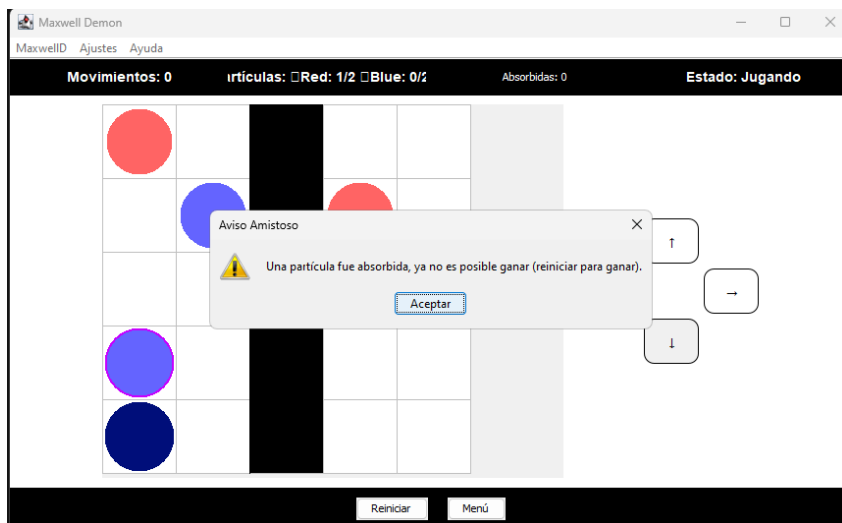
¿Cuántos oyentes necesitan? ¿Por qué?

Para jugar se necesita el MouseListener, ya que todo lo estamos accionando desde el mouse y los listener para accionar cada uno de los elementos puestos en el juego como lo son los botones.

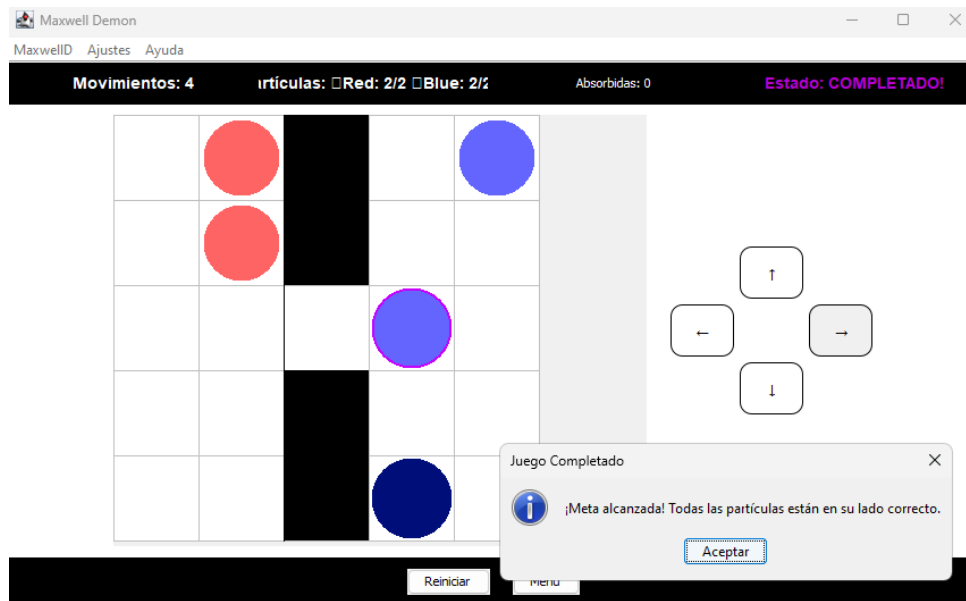
5. Ejecuten el caso de uso y capture las pantallas más significativas.



Otras opciones:

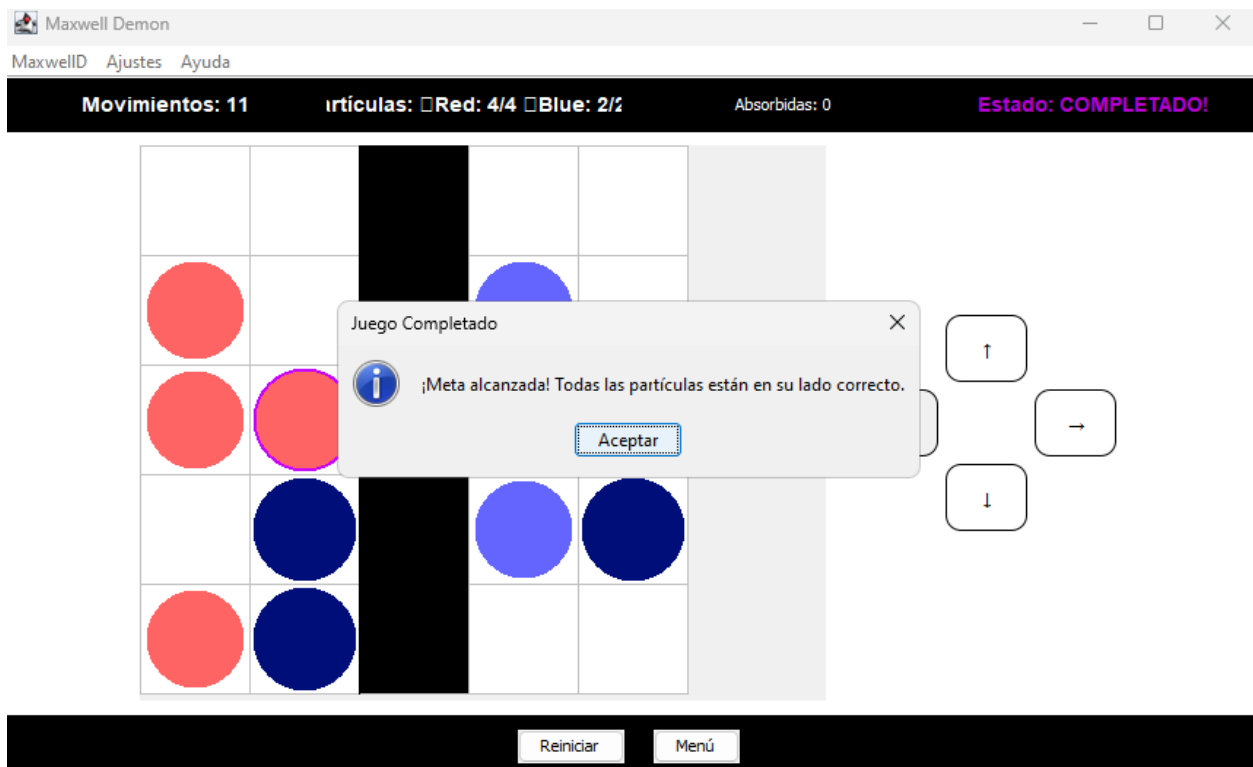
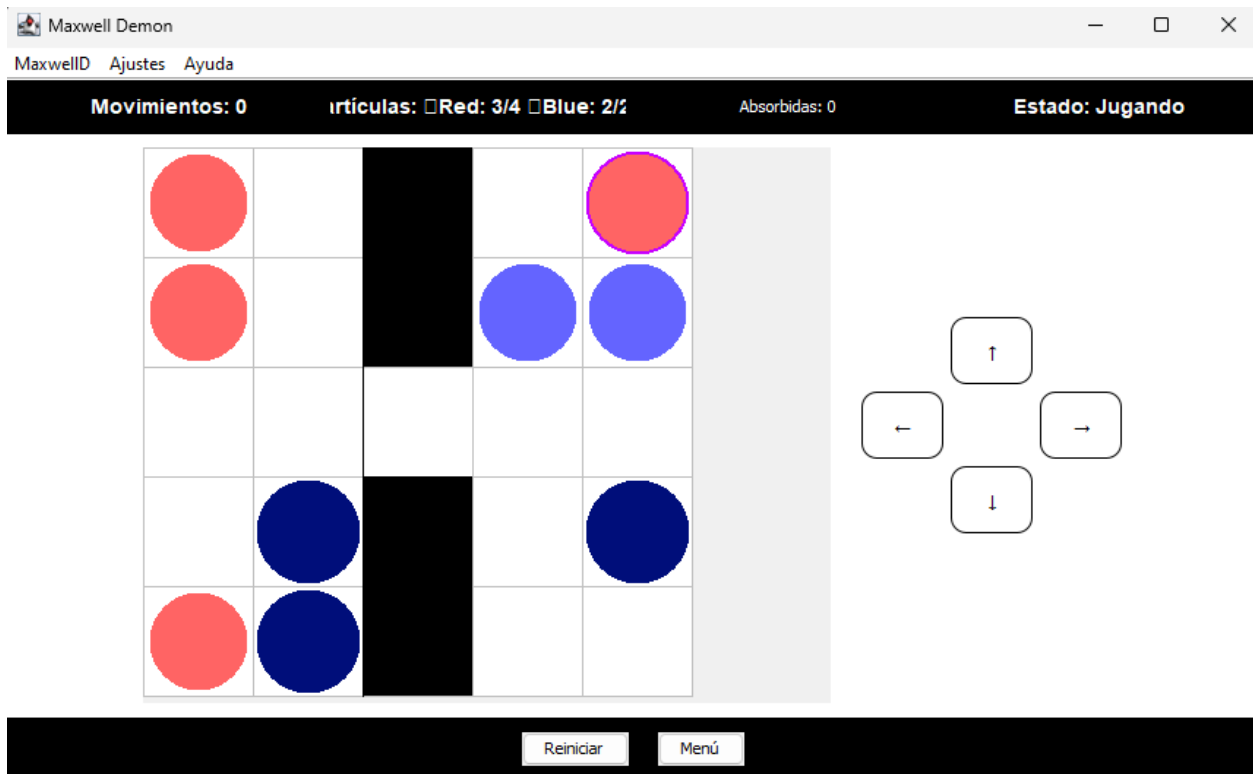


Cuando ganas:



Configuración del juego:





[BONO] Ciclo 7: Reiniciar

[En *.java y lab05.doc]

El objetivo es implementar este caso de uso.

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

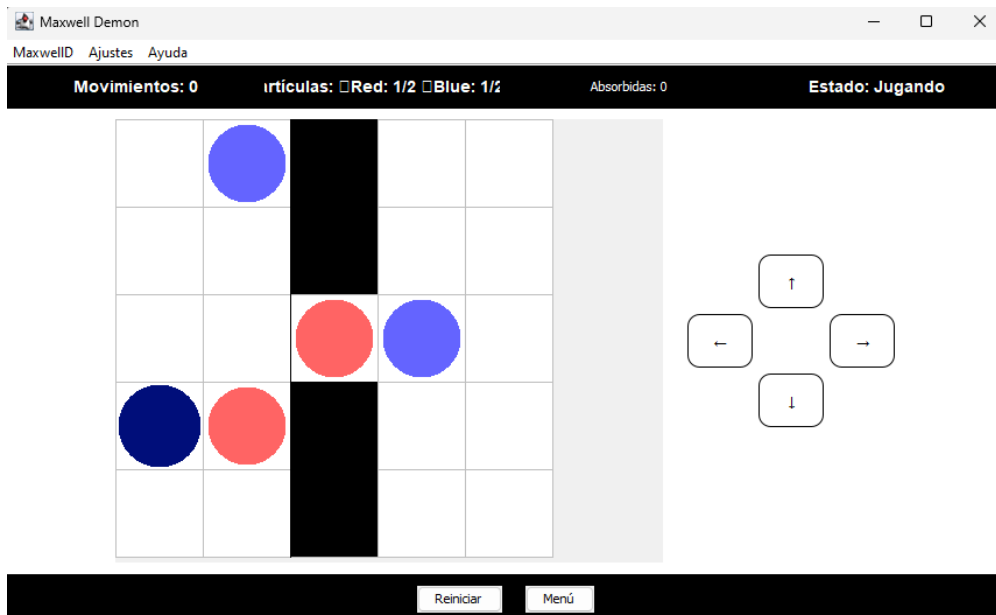
Se implementa un JButton, lo creamos en el constructor y añadimos los botones al panel. Con un listener, configuramos la accion del boton y luego se implementa el método para implementarlo.

2. Implementen los elementos necesarios para reiniciar

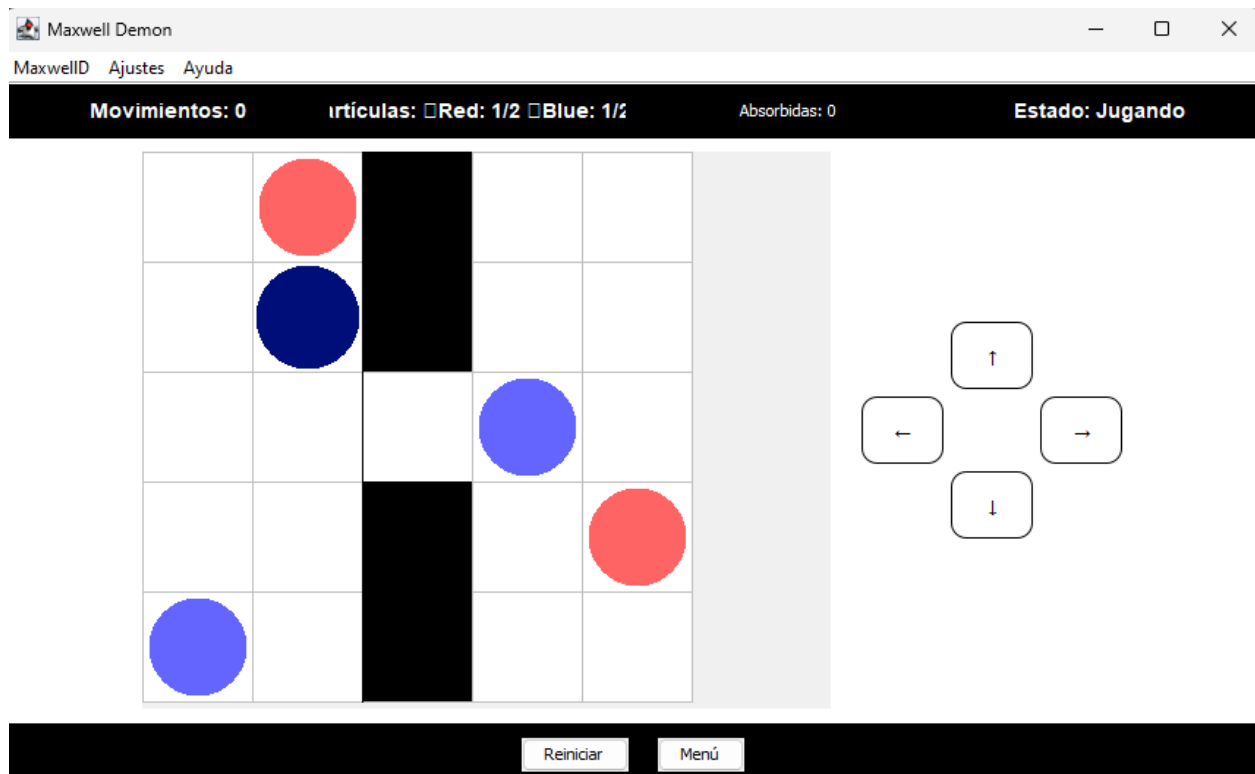
***Revisar DMaxwellGUI.java**

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

Tablero inicial:



Cuando se reinicia:



[BONO] Ciclo 8: Cambiar el tamaño

[En *.java y lab05.doc]

El objetivo es implementar este caso de uso.

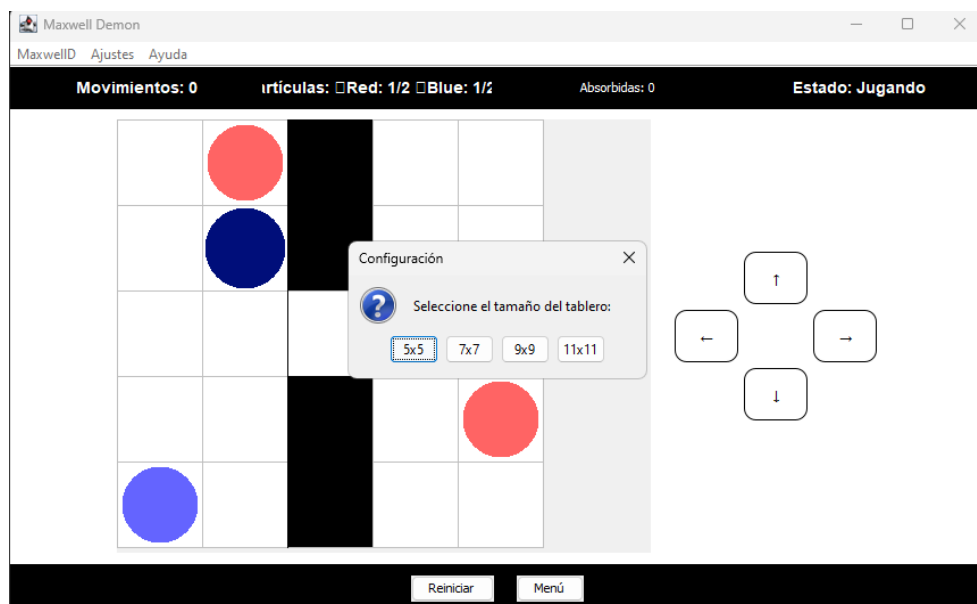
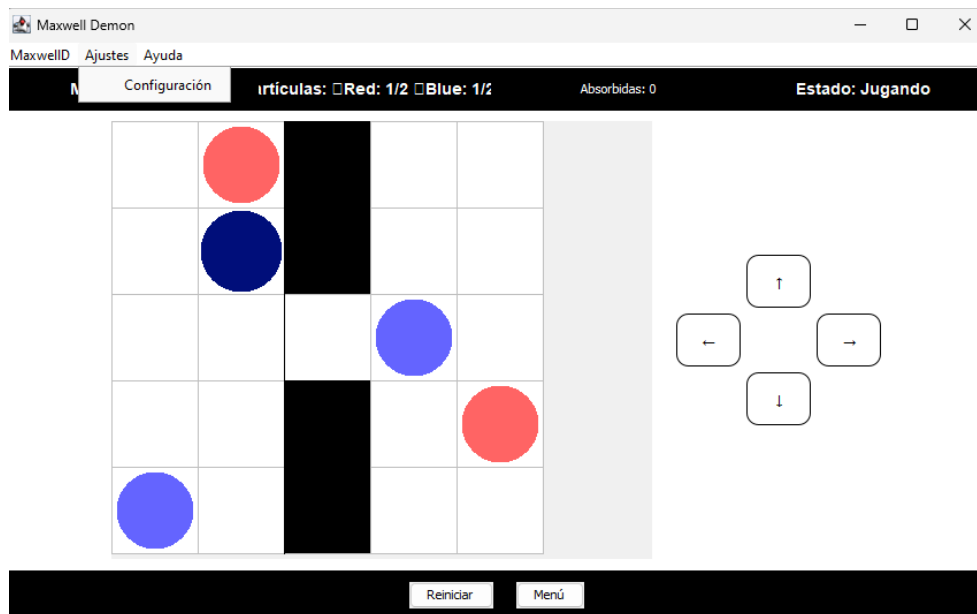
1. Expliquen los elementos a usar para implementar este caso de uso

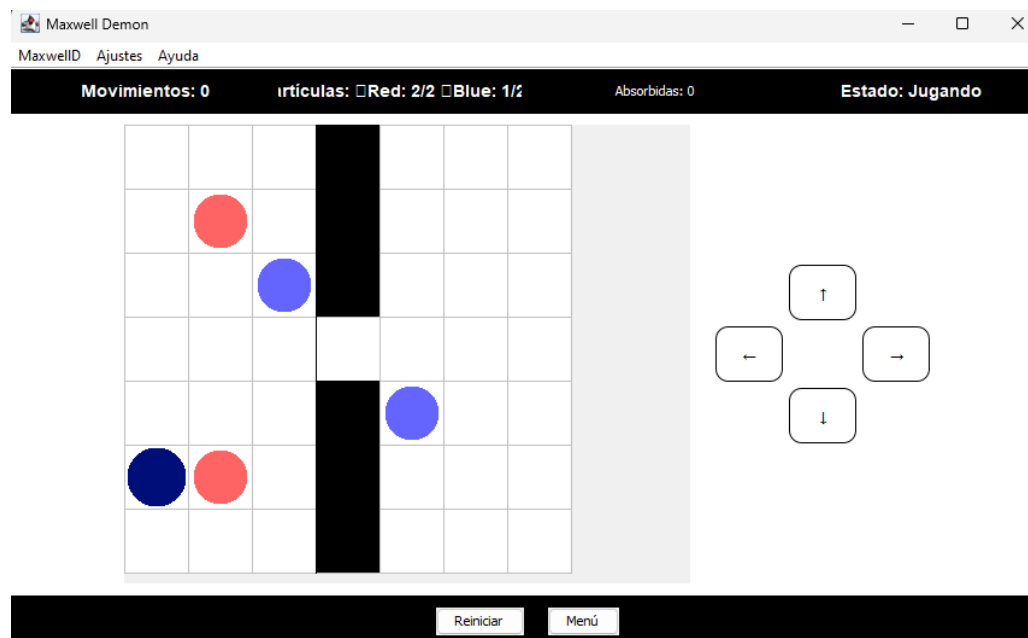
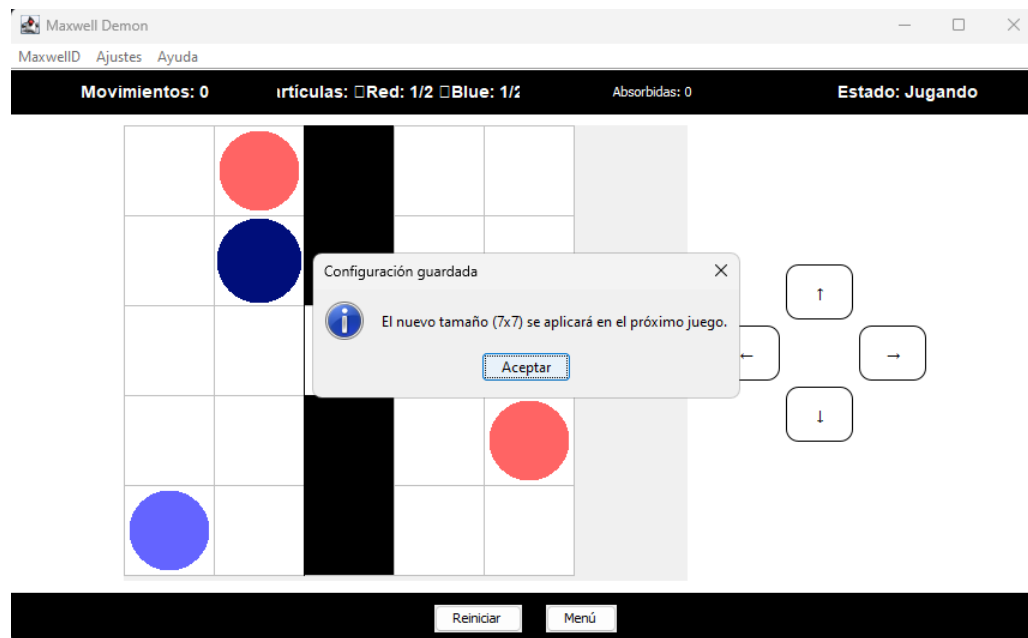
En el JMenuBar añadimos un elemento de ajustes, en este al accionarlo, se despliega un Jmenu donde habrá una opción de configuración, luego se desplegará una ventana emergente donde podremos elegir el tamaño del tablero.

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

*Revisar DMaxwellGUI.java

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/Hombre)

El tiempo invertido fue de aproximadamente 18 horas por hombre.

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

El laboratorio es totalmente terminado, ya que se hicieron todos los puntos en su totalidad cumpliendo con todas las condiciones solicitadas.

3. Seleccionen una práctica XP del laboratorio ¿por qué consideran que es importante?

Nosotros consideramos que la prueba de “When a bug is found” ya que nos ayudó a reducir los errores que teníamos y a fortalecer nuestro código para que su funcionamiento fuera más robusto y tuviera menos errores.

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

Uno de los mayores logros fue todo el desarrollo del gui, ya que fue una de las cosas que mas nos tomo tiempo del laboratorio.

El mayor problema del laboratorio fue la sintaxis del gui, ya que era algo que desconocíamos en su totalidad y nos tomo bastante tiempo la investigación de cada una de las funcionalidades de la librería de awt y swing.

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

Como equipo siempre tratamos de apoyarnos mutuamente, apoyandonos el uno al otro en nuestras debilidades, nos comprometemos a seguir mejorando en la calidad de entrega de nuestros laboratorios.

6. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares Adecuados.

Usamos las siguientes referencias:

Folgado, L. (s.f.). *JFileChooser*. LauraFolgado.es. Recuperado el 26 de abril de 2025, de <https://www.laurafolgado.es/develop/interfaces/tag/jfilechooser/>

ChatGPT. (2025). *Asistencia en programación, lógica y desarrollo de software*. OpenAI.

García Rincón, J. (s.f.). *Componentes JMenuBar, JMenu y JMenuItem*. Recuperado el 26 de abril de 2025, de <https://www.jairogarciarincon.com/clase/interfaces-de-usuario-con-java-swing/componentes-jmenubar-jmenu-y-jmenuitem>

DeepSeek. (2024). DeepSeek Chat (Versión 1.0) [Modelo de lenguaje de IA]. DeepSeek.
<https://www.deepseek.com>