LABORATORIO # 5
PROGRAMACIÓN ORIENTADA A OBJETOS:
INTERFAZ

PRESENTADO A:
MARIA IRMA DIAZ ROSO

PRESENTADO POR:

CAROLINA CEPEDA

UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO

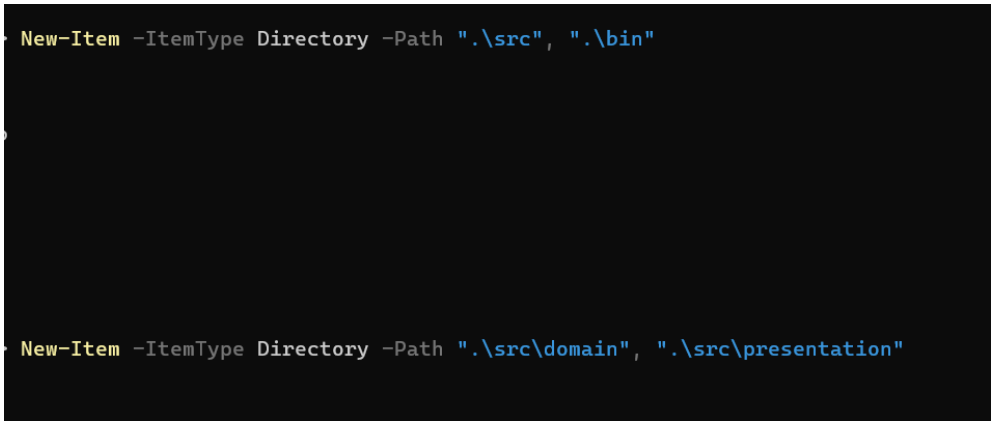
BOGOTA D.C
2025

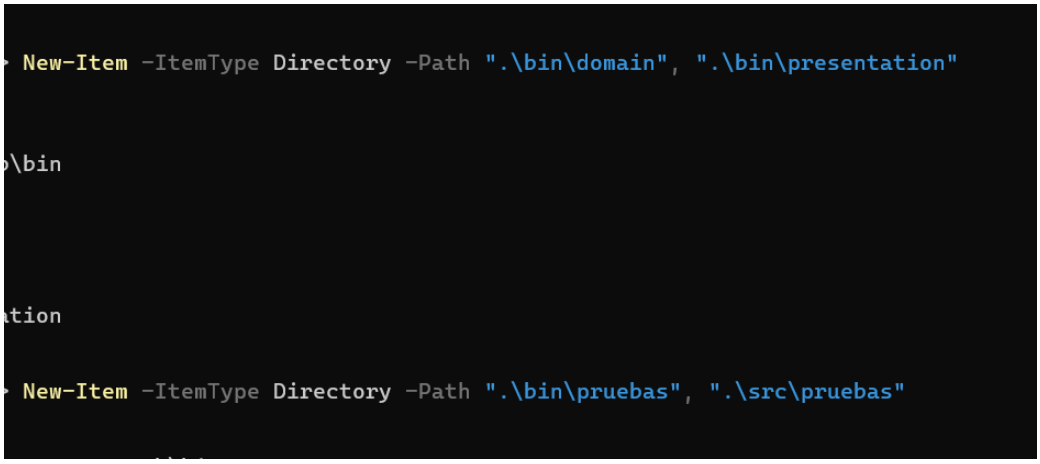
DESARROLLO

Directorios

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

1. 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.





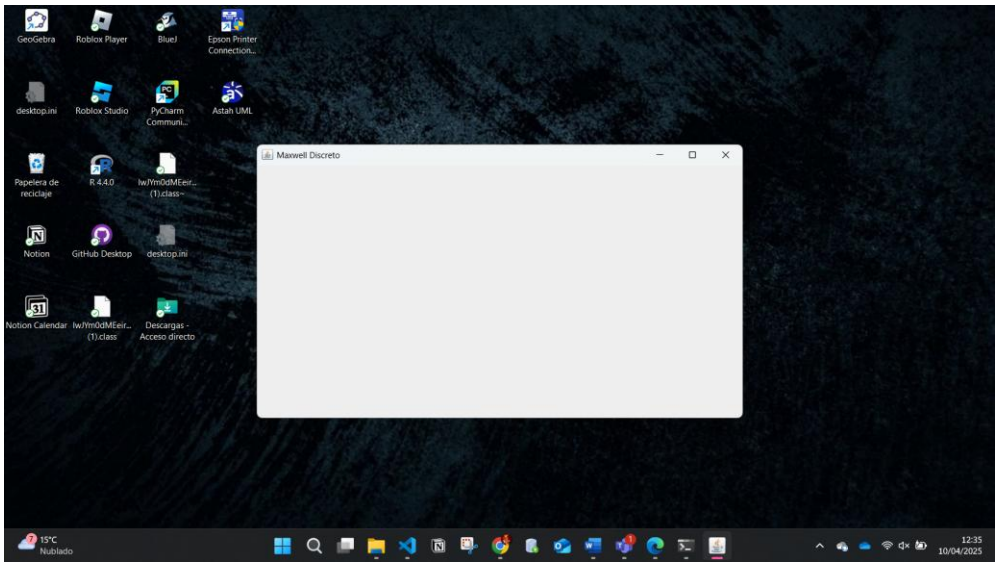
Ciclo 0: Ventana vacía – Salir

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)

```
javac -d bin src/presentation/DMaxwellGUI.java
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.



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

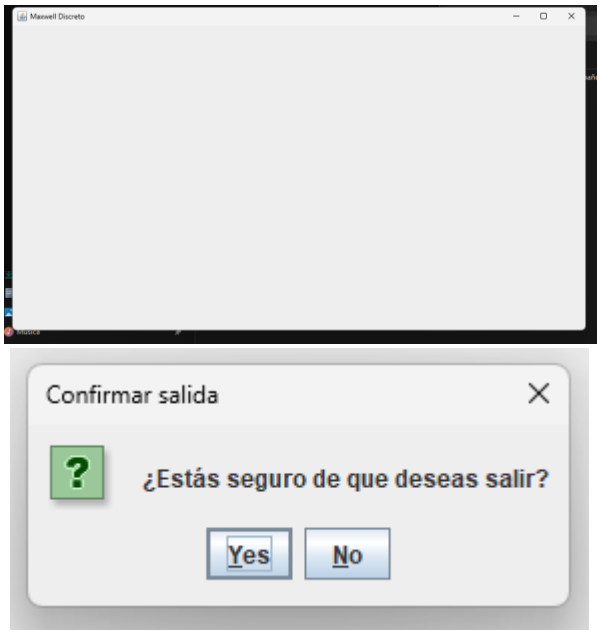
   No se termina la ejecución y para terminarla desde consola se hace uso del atajo ctrl +c.

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?

El método sirve para definir que acción debe realizarse cuando el usuario intente cerrar una ventana en una aplicación de Java.Para confirmar el cierre de la aplicación se puede usar DO_NOTHING_ON_CLOSE y agregar un oyente a la ventana que le pregunte al usuario si realmente quiere cerrar el juego.En el caso de querer simplemente cerrar la aplicación, podemos usar EXIT_ON_CLOSE para que al cerrar la ventana se termine toda la aplicación por defecto.

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.



Al confirmar el cierre de la ventana, esta se cierra y termina la ejecución del programa correctamente.

Ciclo 1: Ventana con menú

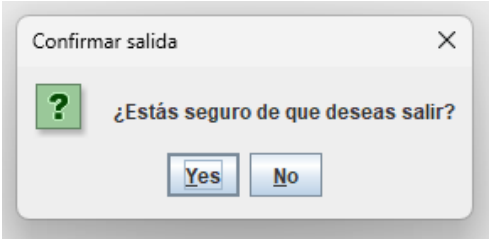
El objetivo es implementar un menú clásico para la aplicación con un final adecuado desde la opción del menú para salir. El menú debe ofrecer mínimo las siguientes opciones :Nuevo, Abrir – Salvar y Salir . Incluyan los separadores de opciones.

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

Los componentes necesarios podrían ser JMenuBar, JMenu, JMenuItem. - los atributos serían los ítems del menú (JMenuItem) y podría las variables ser…

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

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



Ciclo 2: Salvar y abrir

El objetivo es preparar la interfaz para las funciones de persistencia

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

   El componente JFileChooser proporciona un mecanismo sencillo para que el usuario elija un archivo.

   Algunos métodos son:

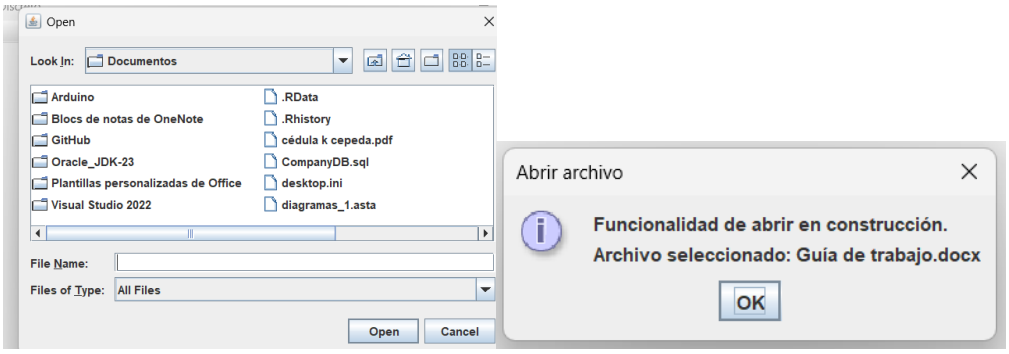
   - EL método JFileChooser construye un objeto de esta clase apuntando al directorio predeterminato del usuario.
   - showOpenDialog hace aparecer un cuadro de dialogo de selección de archivos " Open File"
   - showSaveDialog maneja un cuadro de dialogo para guardar el archivo.
   - getSelectedFile devuelve el archivo seleccionado.

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.

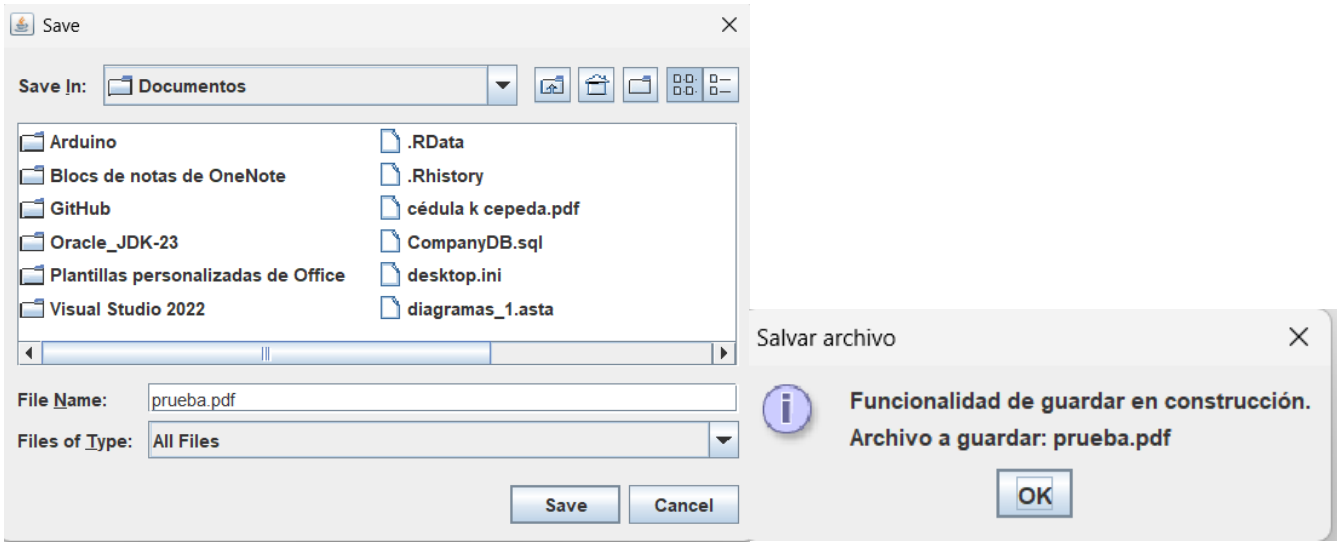
```java
private void prepareActionsMenu() {
    itemNuevo.addActionListener(e -> {
        JOptionPane.showMessageDialog(this, message:"Nuevo archivo creado.");
    });

    itemAbrir.addActionListener(e -> {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            String fileName = fileChooser.getSelectedFile().getName();
            JOptionPane.showMessageDialog(this,
                "Funcionalidad de abrir en construcción.\nArchivo seleccionado: " + fileName,
                title:"Abrir archivo",
                JOptionPane.INFORMATION_MESSAGE);
        }
    });

    itemSalvar.addActionListener(e -> {
        JFileChooser fileChooser = new JFileChooser();
        int result = fileChooser.showSaveDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            String fileName = fileChooser.getSelectedFile().getName();
            JOptionPane.showMessageDialog(this,
                "Funcionalidad de guardar en construcción.\nArchivo a guardar: " + fileName,
                title:"Salvar archivo",
                JOptionPane.INFORMATION_MESSAGE);
        }
    });

    itemSalir.addActionListener(e -> exit());
}
```

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

Abrir



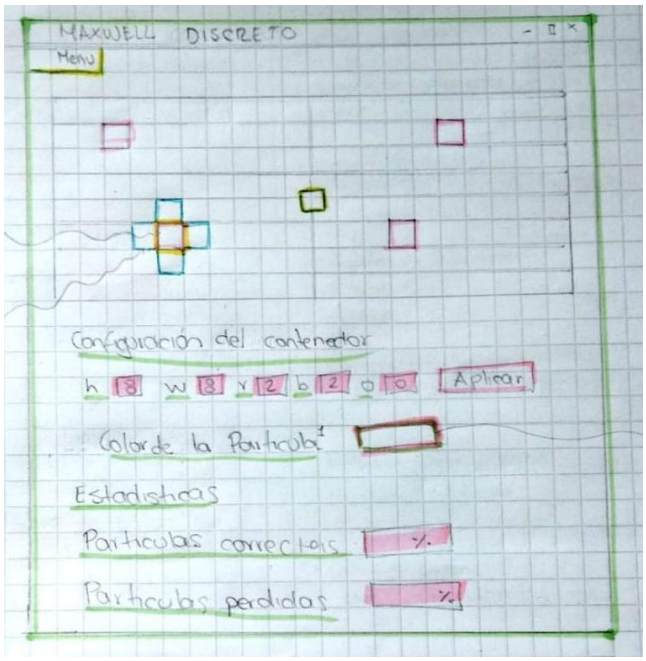
Salvar


Save
Save In:
Documentos
Arduino
Blocs de notas de OneNote
GitHub
Oracle_JDK-23
Plantillas personalizadas de Office
Visual Studio 2022
.RData
.Rhistory
cédula k cepeda.pdf
CompanyDB.sql
desktop.ini
diagramas_1.asta
File Name:
prueba.pdf
Files of Type:
All Files
Save
Cancel
Salvar archivo
Funcionalidad de guardar en construcción.
Archivo a guardar: prueba.pdf
OK

Ciclo 3: Forma de la ventana principal

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)

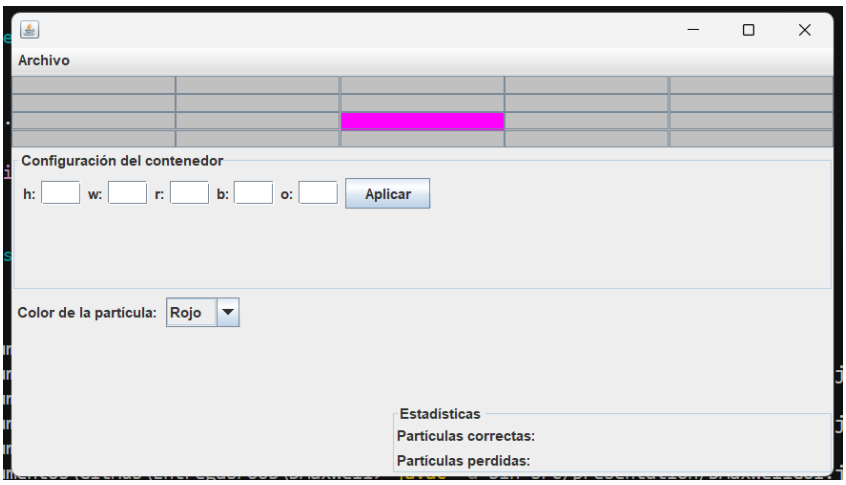


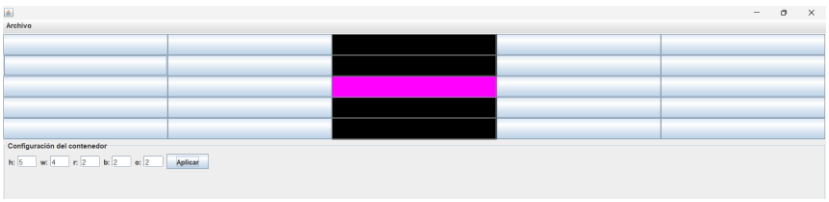
2. Continúen con la implementación definiendo los atributos necesarios y extendiendo el método prepareElements().
   Para la zona del tablero definan un método prepareElementsBoard() y un método refresh() que actualiza la vista del tablero considerando, por ahora, el tablero inicial por omisión. Este método lo vamos a implementar realmente en otros ciclos.

```
}
/**
 * metodo para preparar el panel de simulacion
 */
private void prepareElementsBoard(){

    simulacionPanel = new JPanel();
    simulacionPanel.setLayout(new GridLayout(rows:4, cols:5));

    for(int i = 0; i < 4; i++) {
        for(int j = 0; j <= 4; j++) {
            JButton button = new JButton();
            button.setPreferredSize(new Dimension(width:1, height:1));
            if (i == 2 && j == 2) {
                button.setBackground(Color.MAGENTA); // demonio
            } else {
                button.setBackground(Color.LIGHT_GRAY);
            }
            simulacionPanel.add(button);
        }
    }
    refresh();

}
```

```java
*/
private void prepareElements(){
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

    int width = screenSize.width / 2;
    int height = screenSize.height / 2;
    setSize(width, height);
    setLocation(screenSize.width / 2 - width/2, screenSize.height / 2 - height/2);
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

    //paneles
    prepareConfigPanel();
    prepareElementsBoard();
    prepareColorPanel();
    prepareStatsPanel();
    organizaPaneles();

    //menu
    prepareElementsMenu();

    //acciones
    prepareActions();

}
```

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.
   - Botón elegir color
   - El indicador visual en el borde del botón seleccionado
   - Oyente de los eventos de elegirColor : guarda el botón seleccionado por el usuario , abre el JColorchooser y cambia el color del botón.
   - Oyente de los eventos del botón seleccionado

2. Detalle el comportamiento de JColorChooserespecialmente el método estático showDialog

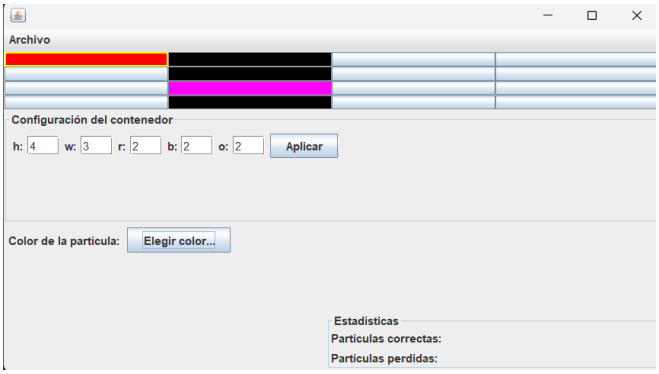
JColorChooser.showDialog es un método estático de la clase JColorChooser que muestra un dialogo donde el usuario puede escoger un color.

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

```java
/**
 * * Método para preparar las acciones del color.
 * */
private void prepareActionsColor() {
    elegirColorButton.addActionListener(e -> {
        if (botonSeleccionado == null) {
            JOptionPane.showMessageDialog(this, message:"Primero selecciona una celda del tablero.");
            return;
        }

        Color colorElegido = JColorChooser.showDialog(this, title:"Selecciona un color para la partícula", currentParticleColor);
        if (colorElegido != null) {
            currentParticleColor = colorElegido;
            botonSeleccionado.setBackground(currentParticleColor);
        }
    });
}
```

```java
button.addActionListener(f -> {
    if (botonSeleccionado != null) {
        botonSeleccionado.setBorder(border:null);
    }
    botonSeleccionado = button;
    botonSeleccionado.setBorder(BorderFactory.createLineBorder(Color.YELLOW, thickness:2));
```

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



Ciclo 5: Modelo DMaxwell

El objetivo es implementar la capa de dominio para **DMaxwell**

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

```java
 */
public DMaxwell(int h, int w, int r, int b, int o) {
    this.h = h;
    this.w = w;
    this.cantidadRojas = r;
    this.cantidadAzules = b;
    this.cantidadHoles = o;
    this.afectadas = 0;

    Set<String> posiciones = new HashSet<>();
    Random random = new Random();

    generarParticulas(r, esRoja:true, random, posiciones);
    generarParticulas(b, esRoja:false, random, posiciones);
    generarAgujeros(o, random, posiciones);

    elementos.add(new Demonio(w / 2, h / 2)); // demonio al centro
}
```

```java
/**...
public void moverParticula(int px, int py, int aumentoX, int aumentoY) {

    if (simulacionTerminada) return;

    for (Iterator<Elemento> it = elementos.iterator(); it.hasNext();) {

        Elemento elemento = it.next();

        if (!(elemento.estoyAhi(px, py) && elemento instanceof Particula particula)) continue;

        int nuevaX = px + aumentoX;
        int nuevaY = py + aumentoY;

        if (interactuarConDemonio(particula, nuevaX, nuevaY, aumentoX, aumentoY, it)) return;

        if (!posicionOcupadaPorParticula(nuevaX, nuevaY, particula) &&
            nuevaX >= 0 && nuevaX < w + 1 && nuevaY >= 0 && nuevaY < h) { // sería mejor manejar el método "posicionCorrecta" en esta clase para manejar las nuevas
            particula.mover(aumentoX, aumentoY);

            if (verificarAgujero(particula, it)) return;
        }

        finish();
        return;
    }
    finish();
}
```

```java
/**...
public double calcularParticulasCaidas() {
    return (afectadas * 100.0)/(cantidadRojas + cantidadAzules);
}

/**...

public double calcularParticulasCorrectas() {
    int totalParticulas = cantidadRojas + cantidadAzules;
    if (totalParticulas == 0) return 0.0;

    int correctas = 0;

    for (Elemento elemento : elementos) {
        if (elemento instanceof Particula particula) {
            if (particula.estoyPosicionCorrecta(h, w)) {
                correctas++;
            }
        }
    }
    return (correctas * 100.0) / totalParticulas;
}
```

2. Ejecuten las pruebas y capturen el resultado.

```
PS C:\Users\Karol\OneDrive\Documento
JUnit version 4.13.2
............Caídas: 100.0%

Time: 0,028

OK (12 tests)
```

Ciclo 6: Jugar

El objetivo es implementar el caso de uso jugar.

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

```
simulacionPanel.repaint();
int rValor = Integer.parseInt(r.getText());
int bValor = Integer.parseInt(b.getText());
int oValor = Integer.parseInt(o.getText());
dMaxwell = new DMaxwell(hTablero, wTablero, rValor, bValor, oValor);
```

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

```
/**
 * Metodo para refrescar el panel de simulacion
 */

private void refresh() {
    for (ActionListener al : aplicarButton.getActionListeners()) {
        aplicarButton.removeActionListener(al);
    }
    aplicarButton.addActionListener(e -> {
        if (leerConfiguracion()) {
            inicializarModelo();
            dibujarTablero();
            colocarElementosEnTablero();
            actualizarEstadisticas();
        }
    });

}
/**
 * metodo para leer la informacion dada en el panel de configuracion
 * @return
 */
private boolean leerConfiguracion() {
    try {
        hTablero = Integer.parseInt(h.getText());
        wTablero = Integer.parseInt(w.getText());
        return true;
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, message:"Por favor, introduce valores válidos para h y w.");
        return false;
    }
}

private void inicializarModelo() {
    int rValor = Integer.parseInt(r.getText());
    int bValor = Integer.parseInt(b.getText());
    int oValor = Integer.parseInt(o.getText());
    dMaxwell = new DMaxwell(hTablero, wTablero, rValor, bValor, oValor);
}
```

```java
private void colocarElementosEnTablero() {
    if (dMaxwell == null) return;

    Component[] components = simulacionPanel.getComponents();
    int cols = wTablero + 1;

    for (Elemento e : dMaxwell.getElementos()) {
        int x = e.getPx();
        int y = e.getPy();
        int index = y * cols + x;

        if (index >= 0 && index < components.length) {
            JButton btn = (JButton) components[index];

            if (e instanceof Particula) {
                Color colorPersonalizado = ((Particula) e).getColorPersonalizado();
                if (colorPersonalizado != null) {
                    btn.setBackground(colorPersonalizado);
                } else {
                    btn.setBackground(((Particula) e).isRed() ? Color.RED : Color.BLUE);
                }
            } else if (e instanceof Agujero) {
                btn.setBackground(Color.BLACK);
            } else if (e instanceof Demonio) {
                btn.setBackground(Color.GRAY);
            }
        }
    }
}
```

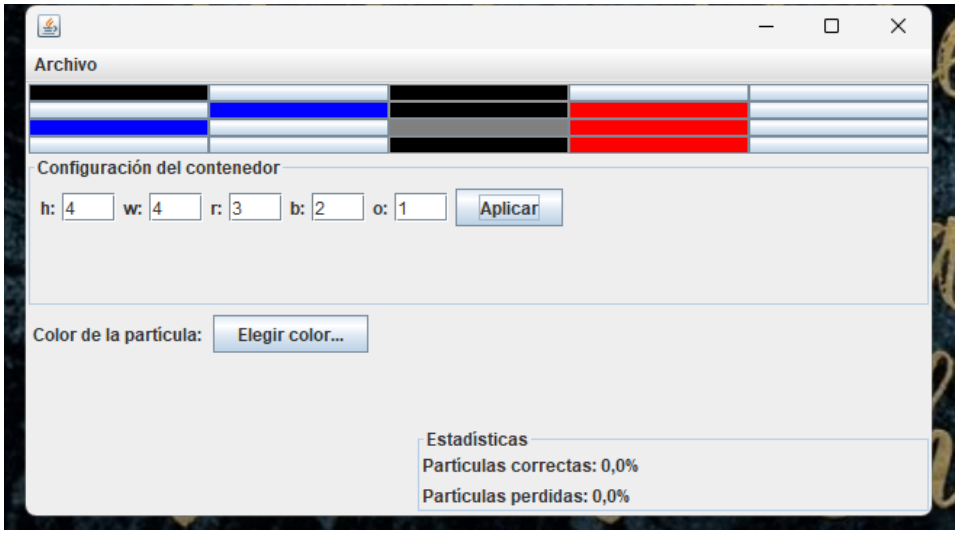
3. Expliquen los elementos necesarios para implementar este caso de uso.
Los elementos clave para implementar "jugar" son:

- Elementos (botones) (Particula, Agujero, Demonio):
- Panel de simulación (tablero)
- Estadísticas referentes al juego

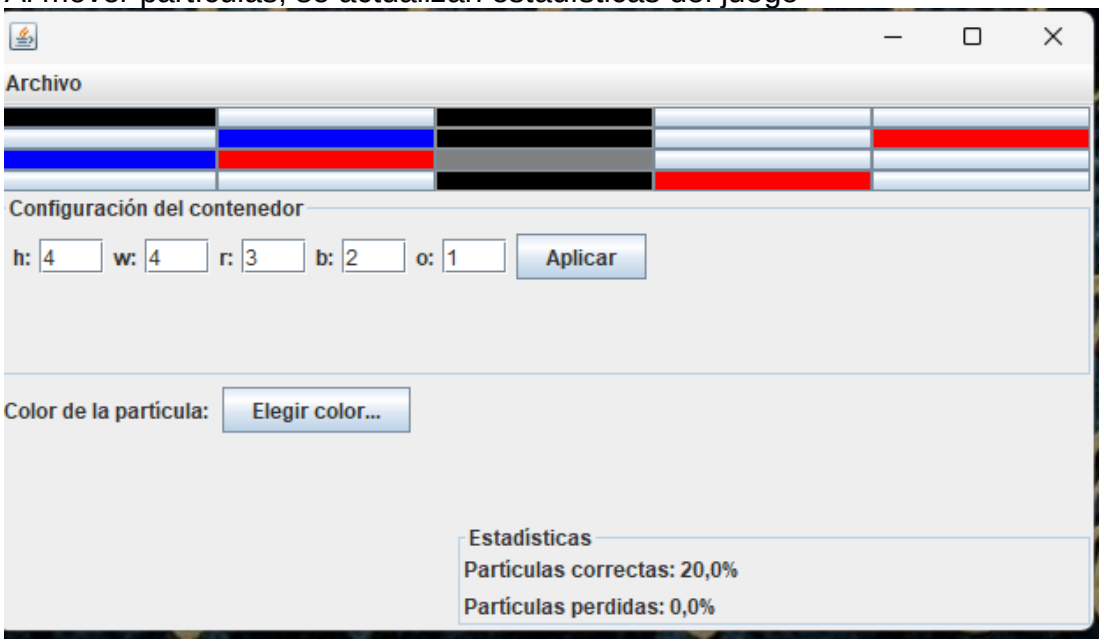
4. Implementen los componentes necesarios para jugar. ¿Cuántos oyentes necesitan?
   ¿Por qué?
   Se hace uso de un oyente para seleccionar una celda para todos los botones que representan los elementos de partículas demonios y agujeros, en el caso de particula si se selecciona este botón se puede elegir alguno de los botones (norte, sur , este , oeste) para su movimiento destino.

5. Ejecuten el caso de uso y capture las pantallas más significativas.
   (se hizo arreglo en partículas perdidas para que muestre el signo de porcentaje en el panel de estadísticas pero por cuestiones de tiempo no se hizo la toma de capturas nuevamente)

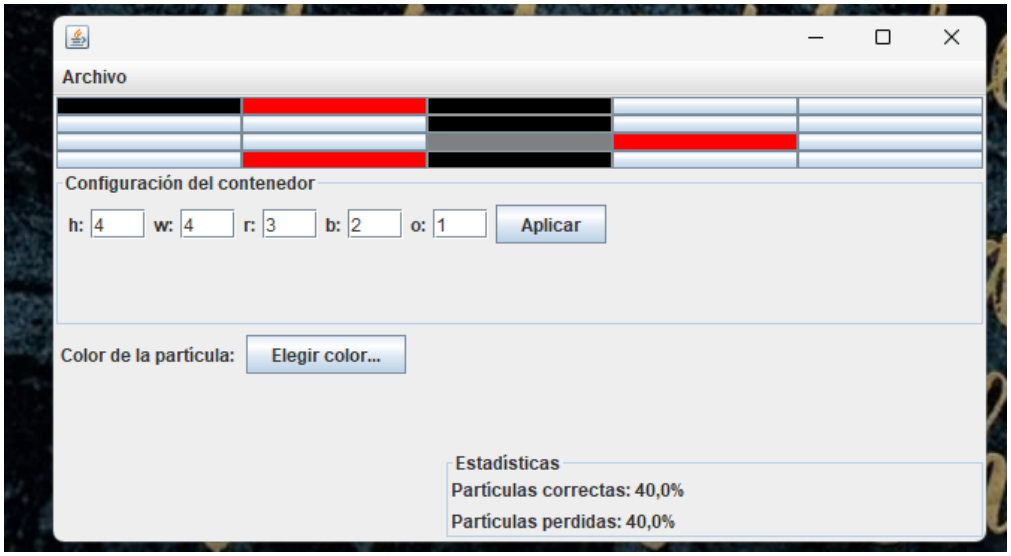
   Después de colocar valores y dar clcik al botón aplicar:



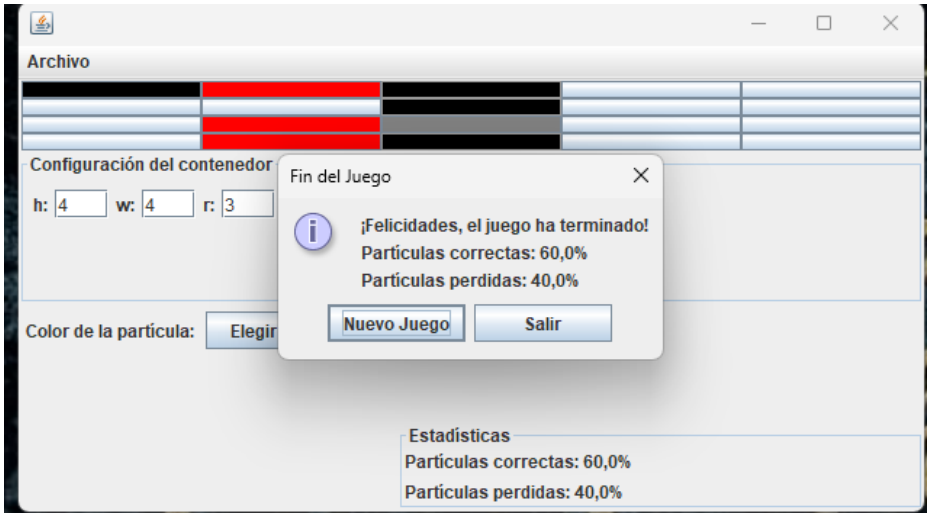
Al mover partículas, se actualizan estadísticas del juego



Después de varios movimientos…



Después de mover la particula roja para ganar el juego:



[BONO] 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 hace uso del modelo (DMaxwell), los campos de entrada (h,w,r,b,o), el tablero (conjunto de botones que conforman las partículas, agujeros, demonios, y espacios vacíos).Tambien, se hace uso de un oyente en el botón nuevo juego que aparece al terminar el juego y en el menú al seleccionar nuevo.

2. Implementen los elementos necesarios para reiniciar

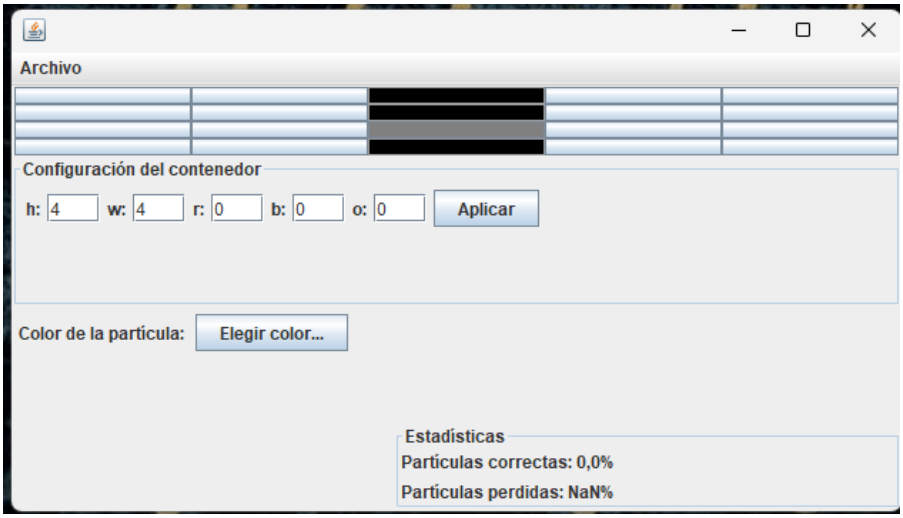
```java
/**
 * * Método para preparar las acciones del menú.
 * se utiliza un listener para cada elemento del menú.
 */
private void prepareActionsMenu() {
    itemNuevo.addActionListener(e -> {
        JOptionPane.showMessageDialog(this, message:"Nuevo archivo creado.");
        reiniciarJuego();
    });
```

```java
private void mostrarMensajeFinal() {
    Object[] opciones = {"Nuevo Juego", "Salir"};
    int eleccion = JOptionPane.showOptionDialog(
        this,
        "¡Felicidades, el juego ha terminado!\n" +
        "Partículas correctas: " + String.format(format:"%.1f%%", dMaxwell.calcularParticulasCorrectas()) + "\n" +
        "Partículas perdidas: " + String.format(format:"%.1f%%", dMaxwell.calcularParticulasCaidas()),
        title:"Fin del Juego",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.INFORMATION_MESSAGE,
        icon:null,
        opciones,
        opciones[0]
    );
    if (eleccion == JOptionPane.YES_OPTION) {
        reiniciarJuego();
```

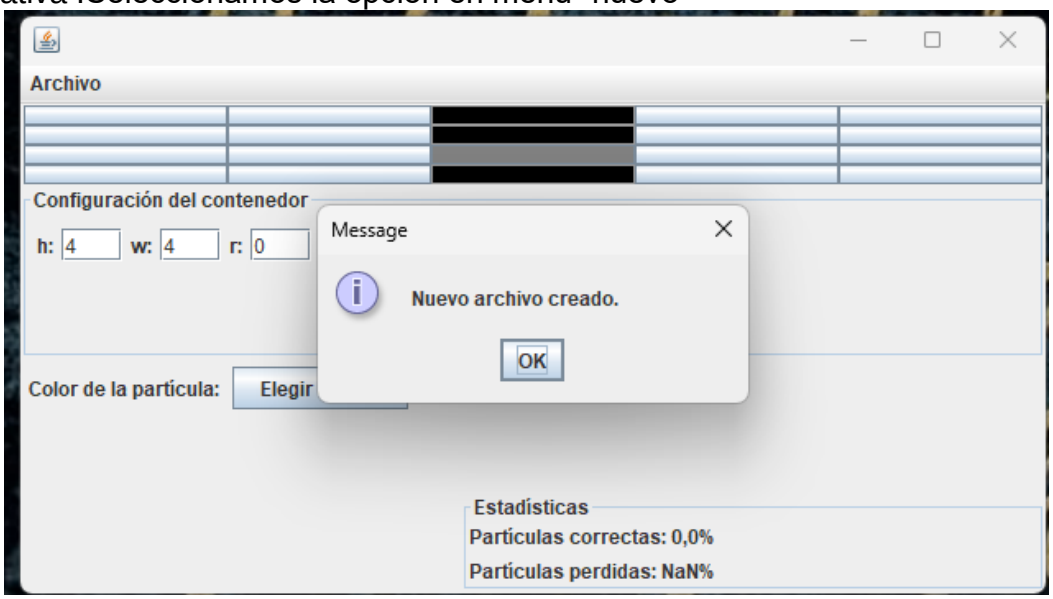
```java
private void reiniciarJuego() {
    dMaxwell = null;
    h.setText(t:"4");
    w.setText(t:"4");
    r.setText(t:"0");
    b.setText(t:"0");
    o.setText(t:"0");
    if (leerConfiguracion()) {
        inicializarModelo();
        dibujarTablero();
        colocarElementosEnTablero();
        actualizarEstadisticas();
    }
}
```

3. Ejecuten el caso de uso y capture las pantallas más significativas.
   Empezamos un juego con valores arbitrarios

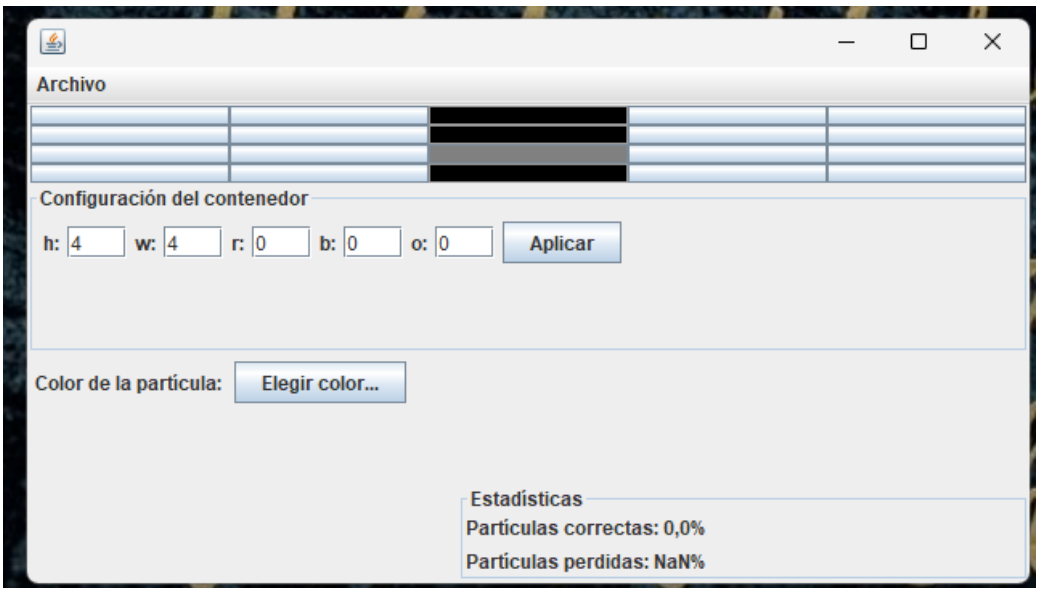
   Siguiendo la prueba anterior, damos click al botón nuevo Juego :



Alternativa :Seleccionamos la opción en menú “nuevo”



Después del click en Ok:



RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?
   22 horas
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
   completo

3. Seleccionen una práctica XP del laboratorio ¿por qué consideran que es importante?
   When a bug is found tests are created. Considero que es una práctica muy importante ya que permite hacer mejoras en los casos que inicialmente no se tomaron en cuenta o que , a la hora de diseñar, no se creyeron de importancia. Esta práctica evita que a la hora de entregar un producto este tenga errores inesperados que deberían ser manejadas en el dominio de el proyecto.

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

   Considero que el mayor logro fue entender el manejo de interfaces con java, específicamente en Swing, ya que el tema de posicionamiento y diseño gráfico es una dificultad para mi y lograr que fuera algo coherente visualmente y funcional fue un gran progreso.

   El mayor problema fue realizar la conexión entre el modelo y la presentación, debido a que al principio no tenía claro cómo estructurar adecuadamente la comunicación entre ambas capas. Se resolvió investigando más sobre el patrón MVC, revisando ejemplos prácticos (laboratorio 4 Plan 15) y ajustando el código para que los componentes pudieran interactuar correctamente respetando la separación de responsabilidades.

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

   Se hizo un buen manejo del tiempo y un buen entendimiento de los temas.

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

   Todas fueron de gran utilidad.

   JFrame (Java Platform SE 8).
   https://docs.oracle.com/javase/8/docs/api/javax/swing/JFrame.html#setDefaultCloseOperation-int-

   JFileChooser (Java Platform SE 8).
   https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html

   How to Use BoxLayout (The Java™ Tutorials > Creating a GUI With Swing > Laying Out Components Within a Container).

https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html

JButton (Java Platform SE 8).
https://docs.oracle.com/javase/8/docs/api/javax/swing/JButton.html

JColorChooser (Java SE 24 & JDK 24).
https://docs.oracle.com/en/java/javase/24/docs/api/java.desktop/javax/swing/JColorChooser.html

Se hace uso de IA para saber la posición de un elemento en el tablero sin la necesidad de pedirle la información al elemento en la lógica.Se hace el comentario en el código.