

**LABORATORIO #5**

**JULIAN CAMILO LOPEZ BARRERO**

**JUAN SEBASTIAN PUENTES JULIO**

**ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO**

**POOB**

**BOGOTÁ, COLOMBIA**

**11 DE ABRIL DE 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.

```
C:\Users\julia\Downloads\DMaxwell>tree
Listado de rutas de carpetas
El número de serie del volumen es 5248-DADC
C:..
├── DMaxwell
│   ├── bin
│   │   ├── domain
│   │   ├── presentation
│   │   └── Test
│   ├── docs
│   └── src
│       ├── domain
│       ├── presentation
│       └── Test
```

### 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.

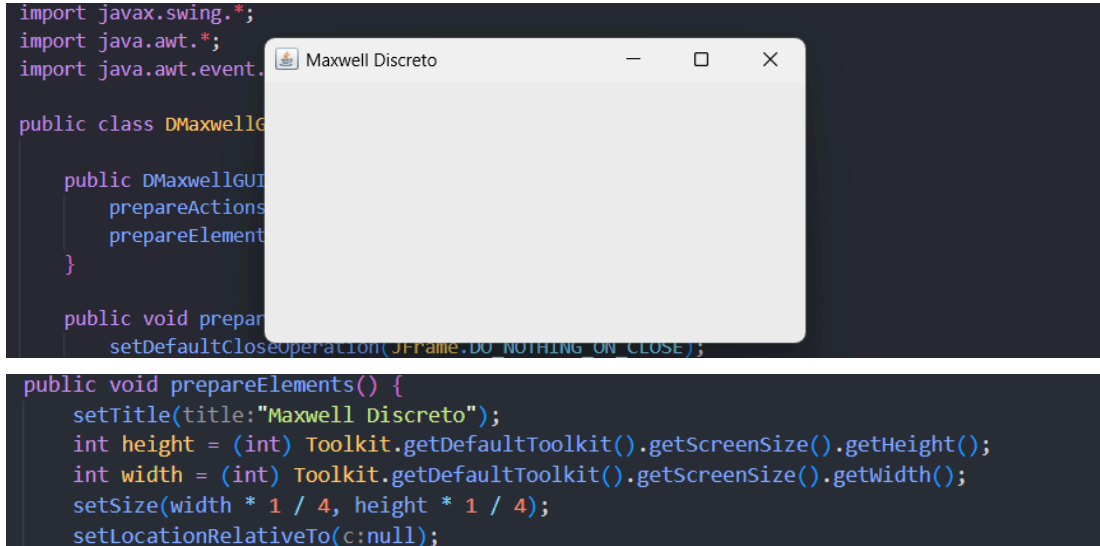
Ejecútenlo. Capturen la pantalla.

```
import javax.swing.*;
import java.awt.*;
import java.awt.Toolkit.*;
public class DMaxwellGUI extends JFrame{
    public DMaxwellGUI(){
        super(title:"Maxwell Discreto");
        setVisible(b:true);
    }
    Run | Debug
    public static void main(String[] args){
        DMaxwellGUI gui = new DMaxwellGUI();
    }
}
```



(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 ubíquenla en el centro. Para eso inicien la codificación del método `prepareElements`. Capturen esa pantalla.



The screenshot shows a code editor with Java code for a class `DMaxwellGUI`. The code includes imports for `javax.swing`, `java.awt`, and `java.awt.event`. The `prepareElements` method is being implemented. Overlaid on the code is a window titled "Maxwell Discreto" with standard Windows window controls (minimize, maximize, close). The window is centered and has a small size, representing 1/4 of the screen.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class DMaxwellGUI {
    public DMaxwellGUI() {
        prepareActions();
        prepareElements();
    }

    public void prepareElements() {
        setTitle(title:"Maxwell Discreto");
        int height = (int) Toolkit.getDefaultToolkit().getScreenSize().getHeight();
        int width = (int) Toolkit.getDefaultToolkit().getScreenSize().getWidth();
        setSize(width * 1 / 4, height * 1 / 4);
        setLocationRelativeTo(c:null);
    }
}
```

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

La ventana sigue ejecutándose en segundo plano, tenemos que forzar el cierre para que este acabe desde la consola presionando `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 **`setDefaultCloseOperation`** sirve para acabar el proceso de la aplicación cuando la cerramos, además, define el comportamiento que tendrá la ventana cuando oprimimos "X". Controla si la ventana simplemente se oculta o se cierra por completo.

Lo usaremos para que a la hora de cerrar nuestro `JFrame` este se cierre y no haga nada.

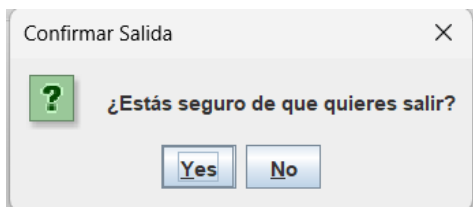
Lo usaremos de manera que termine por completo su ejecución ya que es una ventana que realmente no necesita guardar datos de lo contrario, lo usamos de manera distinta.

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.

```
private void prepareActions() {
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

    //Boton Salir Ventana OYENTE
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            closeWindow();
        }
    });
};
```

```
private void closeWindow(){
    JOptionPane optionPane = new JOptionPane(message:"¿Estás seguro de que quieres salir?", JOptionPane.QUESTION_MESSAGE,
    JOptionPane.YES_NO_OPTION);
    JDialog dialog = optionPane.createDialog(DMaxwellGUI.this, title:"Confirmar Salida");
    dialog.setVisible(b:true);
    if (optionPane.getValue().equals(JOptionPane.YES_OPTION)) {
        System.exit(status: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: proporciona un mecanismo simple para que un usuario cualquiera elija un archivo.
- showOpenDialog: aparece un cuadro de diálogo de selección de archivos “Abrir Archivo”.
- showSaveDialog: aparece un cuadro de diálogo de selección de archivos “Guardar Archivo”.
- getSelectedFile: retorna el archivo seleccionado.

2. Implementan 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.

```
private void prepareElementsMenu() {
    JMenuBar classicMenu = new JMenuBar();
    JMenu fileMenu = new JMenu(s:"File");
    JMenu opciones = new JMenu(s:"Options");
    openItem = new JMenuItem(text:"Open");
    saveItem = new JMenuItem(text:"Save");
    exitItem = new JMenuItem(text:"Exit");
    newItem = new JMenuItem(text:"New");
}
```

```

//Menu Añadir
fileMenu.add(newItem);
fileMenu.addSeparator();
fileMenu.add(openItem);
fileMenu.add(saveItem);
fileMenu.addSeparator();
fileMenu.add(exitItem);
//Menu Clasico Añadir Menu
classicMenu.add(fileMenu);
classicMenu.add(opciones);
setJMenuBar(classicMenu);

```

```

private void openItems(){
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showOpenDialog(DMaxwellGUI.this) == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        JOptionPane.showMessageDialog(DMaxwellGUI.this, "Funcionando" + " " + selectedFile.getName(), title: "Abrir archivo",
        JOptionPane.INFORMATION_MESSAGE);
    }
}

private void saveItems(){
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showSaveDialog(DMaxwellGUI.this) == JFileChooser.APPROVE_OPTION) {
        File selectedFile = fileChooser.getSelectedFile();
        JOptionPane.showMessageDialog(DMaxwellGUI.this, "Funcionando" + " " + selectedFile.getName(), title: "Salvar archivo",
        JOptionPane.INFORMATION_MESSAGE);
    }
}

private void exit(){
    JOptionPane optionPane = new JOptionPane(message: "¿Estás seguro de que quieres salir?",
    JOptionPane.QUESTION_MESSAGE, JOptionPane.YES_NO_OPTION);
    JDialog dialog = optionPane.createDialog(DMaxwellGUI.this, title: "Confirmar Salida");
    dialog.setVisible(b:true);
    if (optionPane.getValue().equals(JOptionPane.YES_OPTION)) {
        System.exit(status:0);
    }
}

```

```

//Abrir Oyente
openItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        openItems();
    }
});

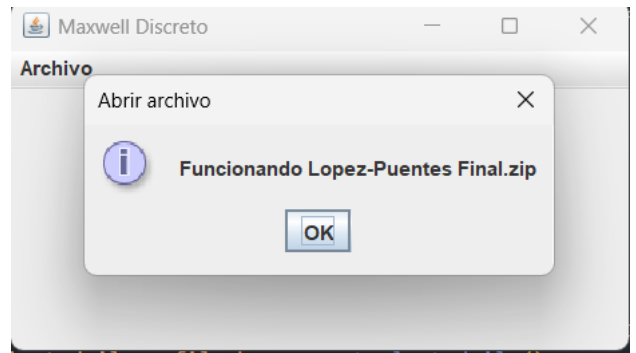
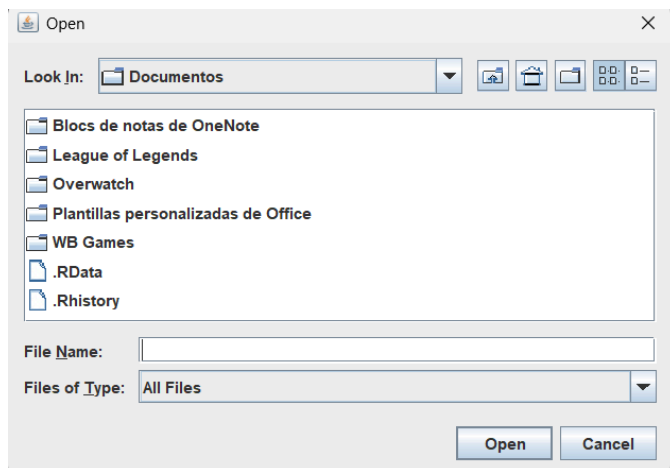
//Salvar Oyente
saveItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveItems();
    }
});

//Salir Oyente
exitItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        exit();
    }
}

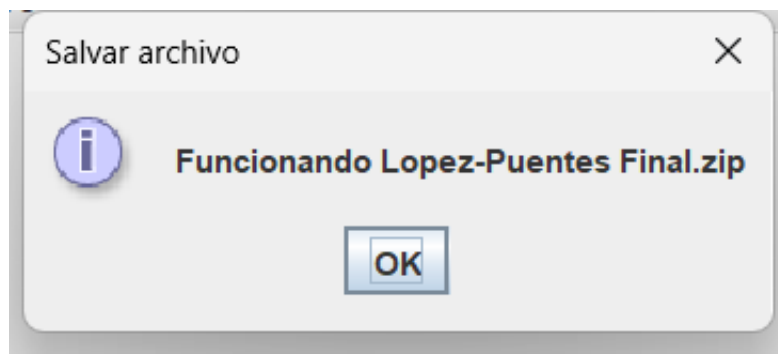
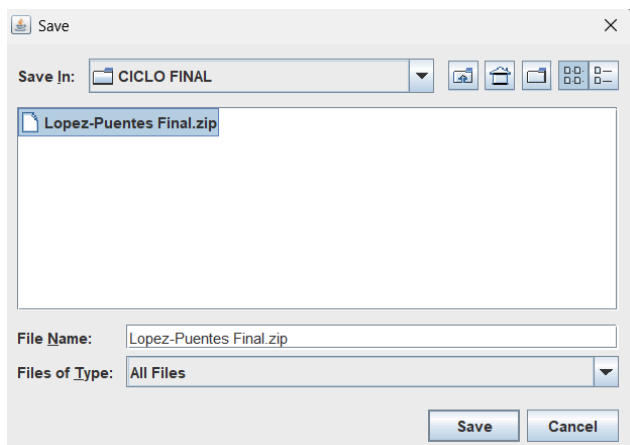
```

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

Para Abrir



Para Salvar



### 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)

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.}

```
/*
 * Prepare the main elements
 */
private void prepareElements() {
    setTitle(title:"Maxwell Discreto");
    height = (int) Toolkit.getDefaultToolkit().getScreenSize().getHeight();
    width = (int) Toolkit.getDefaultToolkit().getScreenSize().getWidth();
    setSize(width * 1 / 4, height * 1 / 4);
    setLocationRelativeTo(c:null);
    prepareElementsBoard();
    prepareElementsMenu();
}
```

```
/*
 * Prepare The elements of the board
 */
private void prepareElementsBoard() {
    setLayout(new GridLayout(rows:3, cols:1));
    board = new Maxwell(maxwell.container());
    add(board);
    textPanel = new JPanel();
    textPanel.setBackground(Color.BLACK);
    label = new JLabel();
    label.setFont(new Font(name:"Times New Roman", Font.BOLD, size:15));
    label.setForeground(Color.WHITE);
    updateTextPanel();
    textPanel.add(label);
    add(textPanel);

    JPanel south = new JPanel(new FlowLayout(FlowLayout.CENTER));
    JPanel maxPanel = new JPanel(new BorderLayout());
    upButton = new JButton(text:"↑");
    downButton = new JButton(text:"↓");
    rightButton = new JButton(text:"→");
    leftButton = new JButton(text:"←");

    maxPanel.add(upButton, BorderLayout.NORTH);
    maxPanel.add(downButton, BorderLayout.SOUTH);
    maxPanel.add(leftButton, BorderLayout.WEST);
    maxPanel.add(rightButton, BorderLayout.EAST);
    south.add(maxPanel);

    add(south);
}
```

```

private void prepareElementsMenu() {
    JMenuBar classicMenu = new JMenuBar();
    JMenu fileMenu = new JMenu(s:"File");
    JMenu opciones = new JMenu(s:"Options");
    openItem = new JMenuItem(text:"Open");
    saveItem = new JMenuItem(text:"Save");
    exitItem = new JMenuItem(text:"Exit");
    newItem = new JMenuItem(text:"New");

    changeColorRedItem = new JMenuItem(text:"Change Color Red Particles");
    changeColorBluesItem = new JMenuItem(text:"Change Color Blue Particles");
    resetItem = new JMenuItem(text:"Reset");
    //Opciones
    opciones.add(changeColorRedItem);
    opciones.addSeparator();
    opciones.add(changeColorBluesItem);
    opciones.addSeparator();
    opciones.add(resetItem);
    //Menu Añadir
    fileMenu.add(newItem);
    fileMenu.addSeparator();
    fileMenu.add(openItem);
    fileMenu.add(saveItem);
    fileMenu.addSeparator();
    fileMenu.add(exitItem);
    //Menu Clasico Añadir Menu
    classicMenu.add(fileMenu);
    classicMenu.add(opciones);
    setJMenuBar(classicMenu);
}

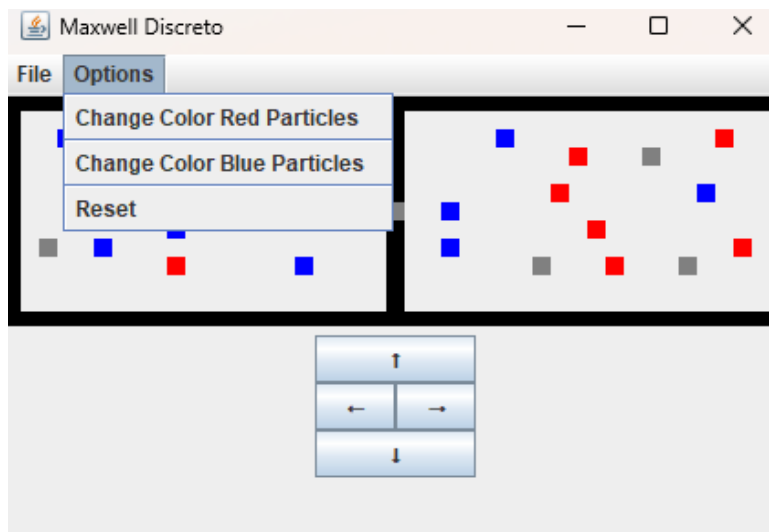
```

```

private void refresh() {
    repaint();
}

```

3. Ejecuten y capturen la pantalla.





## 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.

Se define unos botones que se encuentran en la barra de menú para luego hacer uso del JColorChooser. Se crea uno para cambiar el color de las partículas rojas y otro para las azules. Así mismo, se hace un método que vuelve a pintar el tablero y que el cambio se vea reflejado en el JFrame.

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

- Muestra un cuadro de diálogo modal de selección de color y lo bloquea hasta que se oculta. Si el usuario pulsa el botón "Aceptar", este método oculta o cierra el cuadro de diálogo y devuelve el color seleccionado. Si el usuario pulsa el botón "Cancelar" o cierra el cuadro de diálogo sin pulsar "Aceptar", este método oculta o cierra el cuadro de diálogo y devuelve un valor nulo.

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

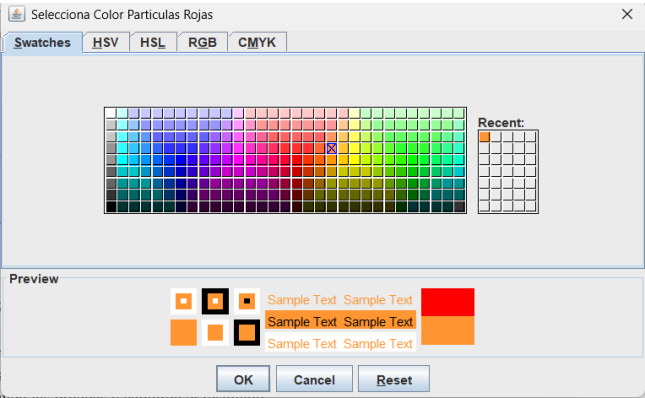
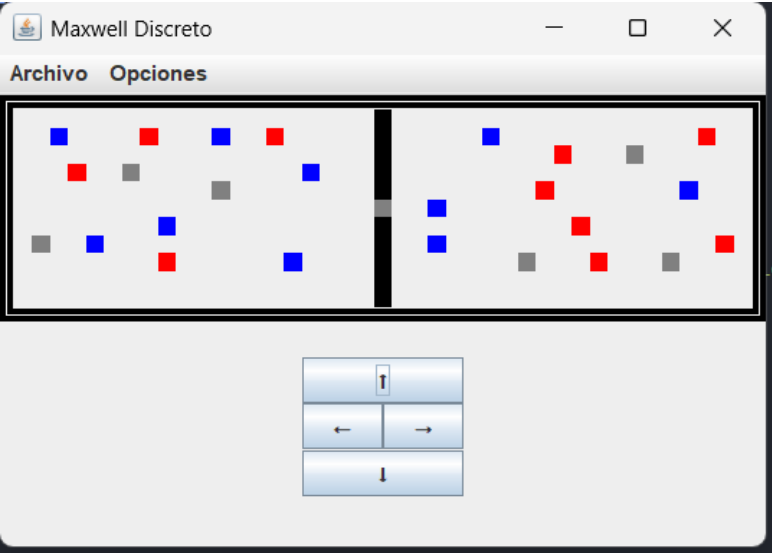
```
//Cambiar Color Rojas Oyente
changeColorRedItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        changeColorReds();
    }
});

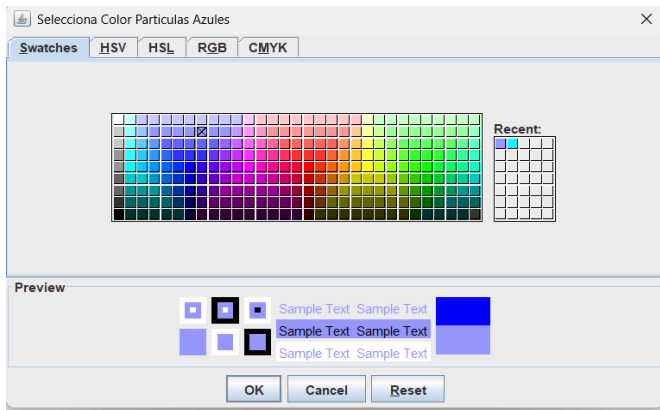
//Cambiar Color Azules Oyente
changeColorBluesItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        changeColorBlues();
    }
});
```

```
private void changeColorReds(){
    Color updateColor = JColorChooser.showDialog(DMaxwellGUI.this, title:"Selecciona Color Particulas Rojas", Maxwell.color1);
    if (updateColor != null) {
        Maxwell.color1 = updateColor;
        board.repaintComponents();
        refresh();
    }
}

private void changeColorBlues(){
    Color updateColor = JColorChooser.showDialog(DMaxwellGUI.this, title:"Selecciona Color Particulas Azules", Maxwell.color2);
    if (updateColor != null) {
        Maxwell.color2 = updateColor;
        board.repaintComponents();
        refresh();
    }
}
```

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





## 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)

```
public int[][] container(){
    return new int[][] { blues, reds, holes, wall};
}
```

```
public DMaxwell(){
    blues = blueDefault.clone();
    reds = redDefault.clone();
    holes = defaultHoles.clone();
    wall = wallDefault.clone();
    demon = posDemonDefault;
    h = 11;
    w = 11;
}

public DMaxwell(int height, int width, int numBlue, int numRed, int numHoles) throws DMaxwellExceptions {
    if(height < 0 || width<0||numBlue<0||numRed<0||numHoles<0) throw new DMaxwellExceptions(DMaxwellExceptions.ONLY_POSITIVE_DIMENTIONS);
    h = height;
    w = width+1;
    r = numRed;
    b = numBlue;
    o = numHoles;
    checkMidWall(h, w);
    createNewPositions();
}
```

```

/*
 * Method to move all particles based on the provided direction.
 * @param direction The direction to move particles. Valid values: 'u' (up), 'd' (down),
 * 'r' (right), 'l' (left).
 */
public void move(char direction) {
    int[] blues1 = blues.clone();
    int[] reds1 = reds.clone();

    ArrayList<Integer> ocupados = new ArrayList<>();
    for (int pos : blues1) ocupados.add(pos);
    for (int pos : reds1) ocupados.add(pos);

    for (int i = 0; i < blues1.length; i++) {
        ocupados.remove((Integer) blues1[i]);
        int pos = positions(blues1[i], direction, ocupados);
        if (verifyHole(pos)) {
            blues1[i] = pos;
            ocupados.add(pos);
        } else {
            blues1[i] = -1;
        }
    }

    for (int i = 0; i < reds1.length; i++) {
        ocupados.remove((Integer) reds1[i]);
        int pos = positions(reds1[i], direction, ocupados);
        if (verifyHole(pos)) {
            reds1[i] = pos;
            ocupados.add(pos);
        } else {
            reds1[i] = -1;
        }
    }

    blues = removePositions(blues1);
    reds = removePositions(reds1);
}

```

```

private void createNewPositions() {
    Random random = new Random();
    ArrayList<Integer> wall1 = arrayToArrayList(wall);
    ArrayList<Integer> disponibles = new ArrayList<>();

    for (int i = 0; i < h * w; i++) {
        if (!wall1.contains(i) && i != demon) {
            disponibles.add(i);
        }
    }

    Collections.shuffle(disponibles, random);

    blues = new int[b];
    reds = new int[r];
    holes = new int[o];

    int index = 0;
    for (int i = 0; i < b; i++) {
        blues[i] = disponibles.get(index++);
    }
    for (int i = 0; i < r; i++) {
        reds[i] = disponibles.get(index++);
    }
    for (int i = 0; i < o; i++) {
        holes[i] = disponibles.get(index++);
    }
}

```

2. Ejecuten las pruebas y capturen el resultado.

```

.
+-- JUnit Jupiter [OK]
|   '-- DMaxwellTest [OK]
|       +-- testVerifyHole() [OK]
|       +-- testConsultDefaultValues() [OK]
|       +-- testMoveDown() [OK]
|       +-- testMoveLeft() [OK]
|       +-- testContainerDefaultValues() [OK]
|       +-- testMoveUp() [OK]
|       '-- testMoveRight() [OK]
+-- JUnit Vintage [OK]
'-- JUnit Platform Suite [OK]

```

## 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.

```
public class DMaxwellGUI extends JFrame {  
    private Maxwell board;  
    private DMaxwell maxwell;  
    private int height;  
    private int width;  
  
    private JButton upButton;  
    private JButton downButton;  
    private JButton rightButton;  
    private JButton leftButton;  
  
    private JMenuItem openItem;  
    private JMenuItem saveItem;  
    private JMenuItem exitItem;  
    private JMenuItem newItem;  
    private JMenuItem changeColorRedItem;  
    private JMenuItem changeColorBluesItem;  
    private JMenuItem resetItem;  
}
```

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

```
private void refresh(){  
    board.refresh(maxwell.container());  
    repaint();  
}
```

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

Un panel con un GridLayout con JTextField para el ingreso de las diferentes dimensiones para la creación de un nuevo tablero con sus diferentes hoyos y partículas. Además, de los botones definidos anteriormente.

4. Implementen los componentes necesarios para jugar. ¿Cuántos oyentes necesitan?

Necesitamos por lo menos 8 oyentes para la realización de jugar.

¿Por qué?

Cuatro oyentes para el movimiento, 1 para crear un nuevo tablero, 2 para cambiar el color de las partículas y uno para la información del juego.

```

private void prepareActionMenu(){

    //Nuevo Oyente
    newItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            newDataMaxwell();
        }
    });

    //Abrir Oyente
    openItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            openItems();
        }
    });

    //Salvar Oyente
    saveItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            saveItems();
        }
    });

    //Salir Oyente
    exitItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            exit();
        }
    });

    //Cambiar Color Rojas Oyente
    changeColorRedItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            changeColorReds();
        }
    });

    //Cambiar Color Azules Oyente
    changeColorBluesItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            changeColorBlues();
        }
    });

    //Reiniciar Oyente BONO
    resetItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            resetToDefaultBoard();
        }
    });
}

```

```

private void prepareActions() {
    setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

    //Boton Salir Ventana OYENTE
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            closeWindow();
        }
    });
    //Oyente Boton Subir
    upButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            movement(direction:'u');
        }
    });
    //Oyente Boton Bajar
    downButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            movement(direction:'d');
        }
    });
    //Oyente Boton Izquierda
    leftButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            movement(direction:'l');
        }
    });
    //Oyente Boton Derecha
    rightButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            movement(direction:'r');
        }
    });

    prepareActionMenu();
}

```

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

### [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.

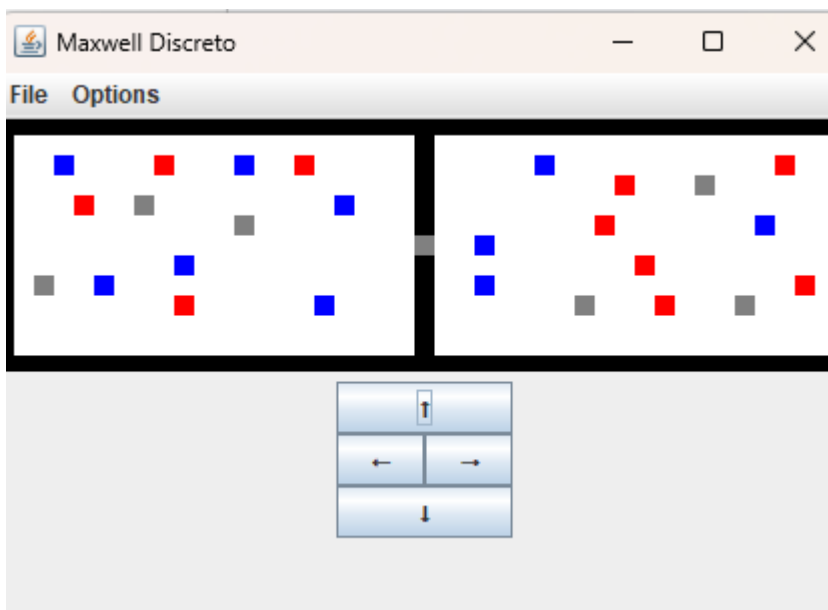
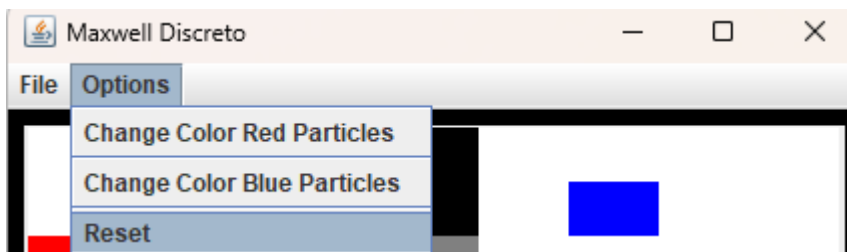
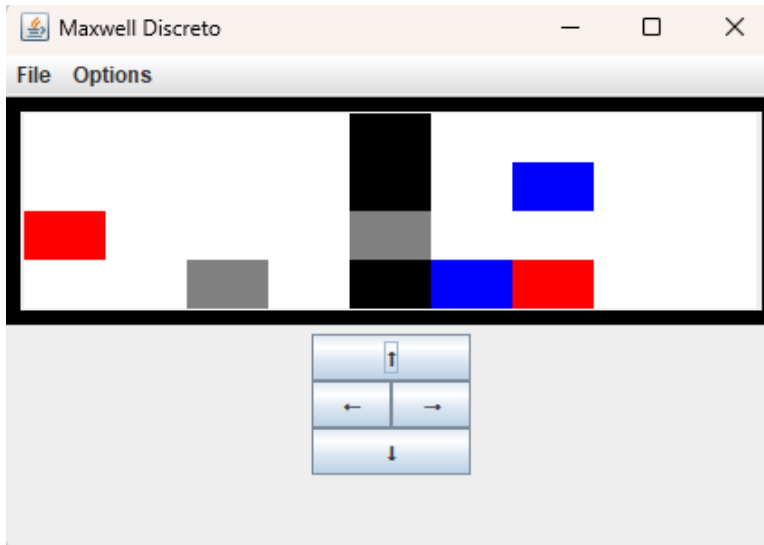
Se debe crear un botón que reinicie el juego al modelo propuesto. Remover el tablero anterior y agregar el predefinido nuevamente.

2. Implementen los elementos necesarios para reiniciar



```
private void resetToDefaultBoard(){
    maxwell = new DMaxwell();
    remove(board);
    board.resetColors();
    board = new Maxwell(maxwell.container());
    add(board,index:0);
    revalidate();
    refresh();
}
```

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)

Alrededor de 20 horas cada uno.

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

El estado del laboratorio está terminado ya que se implementó todo lo solicitado.

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

XP. Testing. When a bug is found test are created. Creemos importante esta práctica ya que nos ayuda a encontrar el fallo en nuestro código.

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

Nuestro mayor logro fue realizar el movimiento de las partículas porque tardamos la mayor parte del laboratorio realizando dicha funcionalidad. El mayor problema fue pintar el tablero inicial. Para resolverlo decidimos usar un GridLayout y paneles.

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

Nuevamente como equipo hicimos uso de pair programming así como también el dividir equitativamente el trabajo.

Nos comprometemos a mejorar nuestra comunicación para desarrollar los laboratorios eficientemente.

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

<https://docs.oracle.com/javase/8/docs/api/javax/swing/JColorChooser.html>

<https://www.jairogarciarincon.com/clase/interfaces-de-usuario-con-java-swing/componentes-jmenubar-jmenu-y-jmenuitem>

## REFERENCIAS

- [JFileChooser \(Plataforma Java SE 8\)](#)
- [JFileChooser - ChuWiki](#)
- [JMenu \(Java Platform SE 8\)](#)
- [JOptionPane \(Java Platform SE 8\)](#)
- [File \(Java Platform SE 8\)](#)
- [A Closer Look at the Paint Mechanism \(The Java™ Tutorials > Creating a GUI With Swing > Performing Custom Painting\)](#)
- [A Closer Look at the Paint Mechanism \(The Java™ Tutorials > Creating a GUI With Swing > Performing Custom Painting\)](#)
- [Color \(Java Platform SE 8\)](#)
- [🍷 Usar el Componente JColorChooser para la Selección de COLOR en JAVA ☕ | Curso JAVA 🔥 Episodio #67 - YouTube](#)
- [java - Does SwingUtilities.updateComponentTreeUI\(\) method set the current L&F to all super and sub components? - Stack Overflow](#)