

Objetivo general del taller

El objetivo general de este taller es aprender a utilizar el framework Swing, integrarlo con Java2D y construir una interfaz gráfica de usuario para una aplicación existente.

Objetivos específicos del taller

Durante el desarrollo de este taller se buscará el desarrollo de las siguientes habilidades:

1. Reconocer las principales clases del framework Swing y poder explicar su uso dentro de un GUI.
2. Explicar y ser capaz de utilizar los principales Layouts de Swing.
3. Entender y ser capaz de utilizar el modelo de eventos de Swing.
4. Implementar un GUI para un programa usando Swing.
5. Explicar y ser capaz de utilizar Java2D para dibujar sobre una superficie Graphics2D.
6. Aprender a utilizar elementos de tipo JList, reconociendo el rol del modelo y de los *renderers*.

Instrucciones generales

En este taller tendrán que construir la interfaz gráfica a partir de un esqueleto que sólo incluye las clases con la lógica de la aplicación. **NO pueden hacerse modificaciones a estas clases.** Todo el código que escriba para solucionar el taller debe quedar dentro de un paquete nuevo.

Utilice como guía para organizar las clases de la interfaz y asignarles responsabilidades el estilo definido en las aplicaciones disponibles para los cursos APO1 y APO2 (<http://cupi2.uniandes.edu.co/>). Utilice también la referencia (disponible en Brightspace):

[11] Capítulo 5 (Construcción de la Interfaz Gráfica): J. Villalobos, Fundamentos de Programación - Aprendizaje Activo Basado en Casos, 2005. <https://cupi2.virtual.uniandes.edu.co/libro-del-curso-pdf>

Descripción del Juego: Lights Out

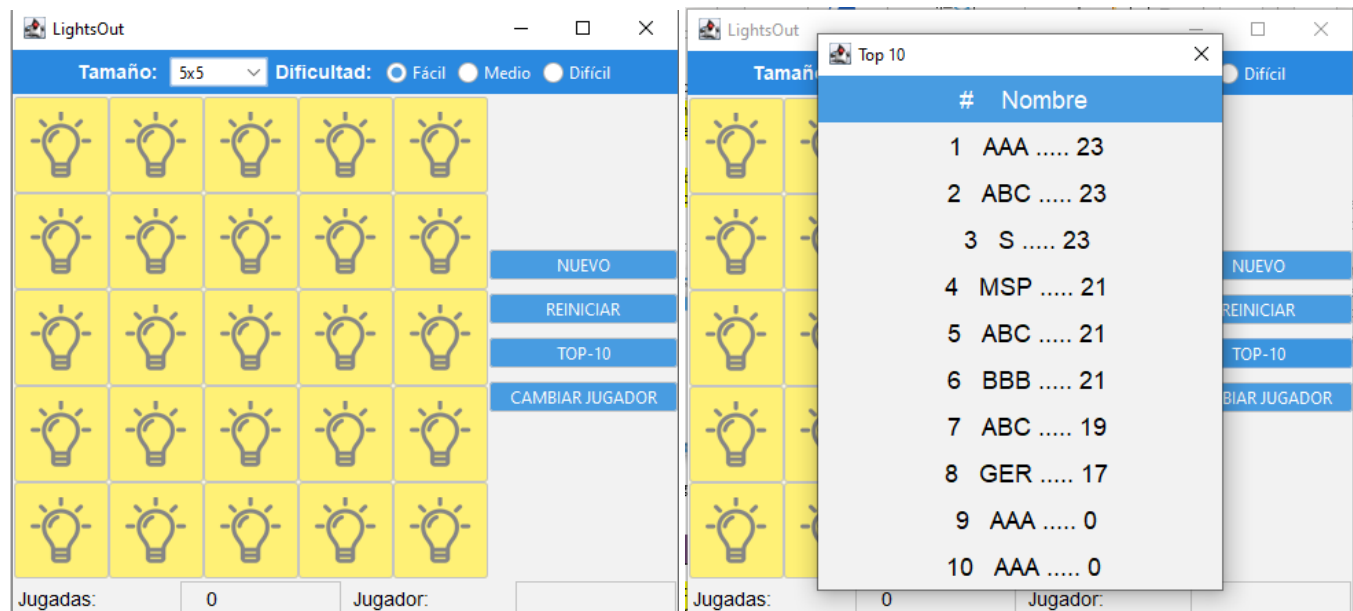
El juego Lights Out se basa en un tablero con luces que pueden estar prendidas o no. Cuando el jugador selecciona una luz, esa luz cambia de estado: si estaba apagada se enciende y si estaba prendida se apaga. Lo mismo ocurre con las casillas adyacentes a la seleccionada. El objetivo del juego es, a partir de una configuración inicial con luces encendidas y apagadas, llegar en la menor cantidad de jugadas a una configuración donde todas las luces estén encendidas ¹.

¹ Más información sobre el juego en: [https://en.wikipedia.org/wiki/Lights_Out_\(game\)](https://en.wikipedia.org/wiki/Lights_Out_(game))

Aunque usualmente se juega en un tablero de 5x5, nuestra implementación del juego tendrá tableros de tamaño variable y las configuraciones iniciales se generarán empezando de un tablero iluminado y jugando en posiciones aleatorias para garantizar que exista una solución.

Nuestra implementación también tendrá un sistema de puntuación que combina el tamaño del tablero con la cantidad de jugadas necesarias para resolver el tablero, y será capaz de mantener las 10 mejores puntuaciones obtenidas en el juego (Top-10).

Las siguientes imágenes muestran cómo podría verse la aplicación:



La captura de la izquierda muestra el tablero en medio de un juego. La figura de la derecha muestra la ventana con el Top-10 de puntuaciones.

Parte 1: Estudiar el esqueleto

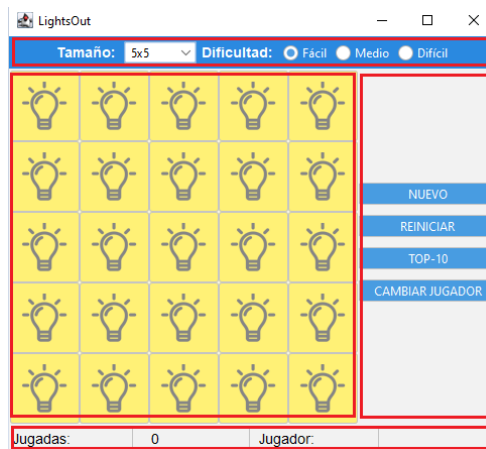
Revise las clases que hacen parte del esqueleto y que implementan la lógica del juego. Familiarícese con ellas porque tendrá que utilizarlas para construir el resto de la aplicación.

Parte 2: Construir la interfaz

En esta parte usted debe construir la interfaz gráfica para la aplicación. La aplicación debería verse muy similar a las capturas de pantalla que se mostraron antes: pueden hacer modificaciones estéticas, pero tienen que usar la misma organización básica y los mismos widgets. El tablero de juego (la cuadrícula de bombillos) **DEBE hacerse utilizando Java2D**. Es decir que no puede utilizar botones ni ningún otro widget dentro del panel que muestra el tablero de juego.

Ayudas

La siguiente imagen muestra la descomposición en paneles de la ventana principal de la aplicación (que usa BorderLayout).



La ventana para mostrar el Top10 puede ser un JDialog para que bloquee la ventana principal.

Puede usar el siguiente código dentro del constructor de la ventana principal para que se salve el Top10 actualizado cada vez que se cierre la aplicación:

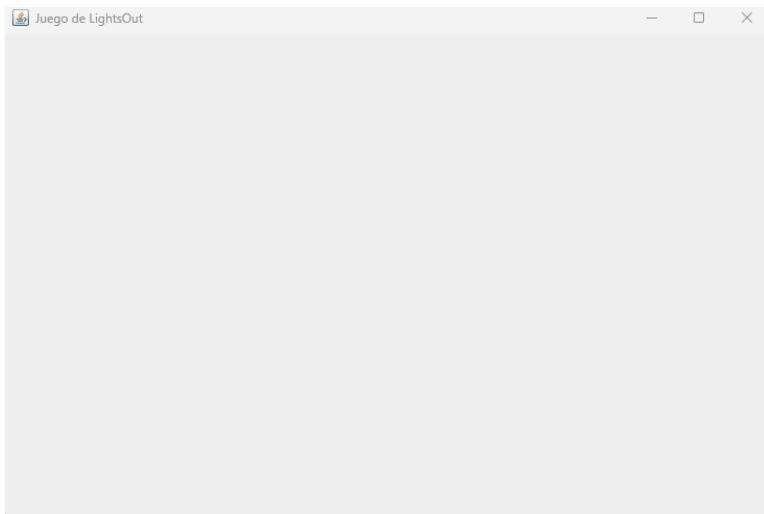
```
// Esto se usa para que al cerrar la ventana se salven los resultados
addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent e)
    {
        salvarTop10();
    }
});
```

El esqueleto incluye una librería con un tema basado en Material Design. Puede utilizarlo agregando la siguiente línea al método main:

```
FlatLightLaf.install();
```

Actividades

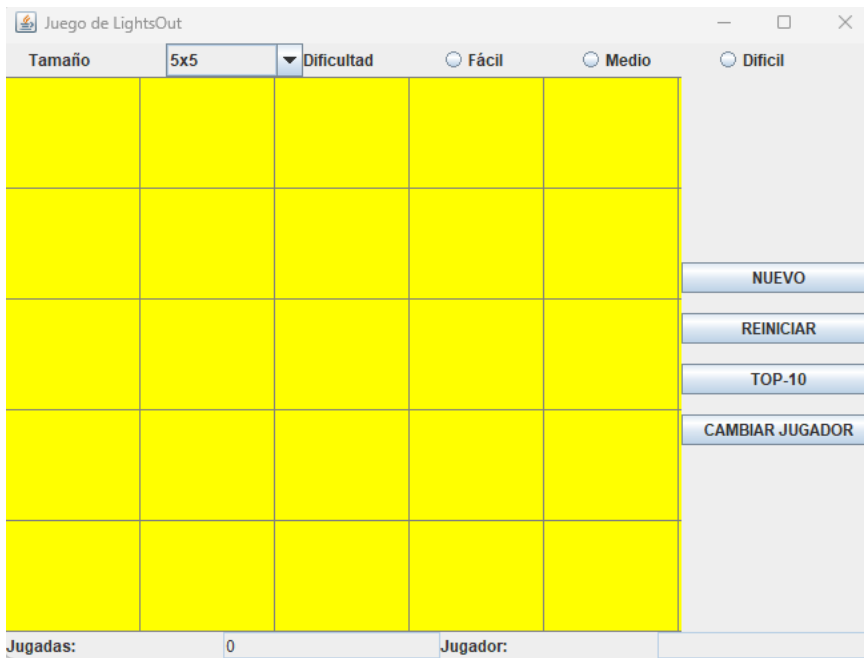
1. En el paquete `src/uniandes.dpoo.taller7.interfaz1` cree una clase que va a funcionar como la ventana principal de la aplicación, póngale un tamaño por defecto y un título, para que al ejecutarla se vea como la imagen a continuación.



2. En el paquete `src/uniandes.dpoo.taller7.interfaz2` copie y pegue la(s) clase(s) que creó en el punto anterior. En este paso tendrá que crear nuevas clases para la implementación de los paneles. Para el panel superior tenga en cuenta que los elementos son `JLabel`, `JComboBox` y `JRadioButton`. Para el panel de la derecha los elementos son `JButton`. Para el panel inferior los elementos son `JLabel` y `JTextField`. De momento, no se implementará el panel con el tablero de juego. Implemente las clases que serán los paneles y modifique la ventana principal para que incluya dichos paneles en un `BorderLayout` y se vea como la imagen a continuación.



3. En el paquete `src/uniandes.dpoo.taller7.interfaz3` copie y pegue las clases que implementó en el punto anterior. En este paso tendrá que crear por lo menos una nueva clase para la implementación del panel del tablero de juego. Este panel debe sobrescribir el método `paint(Graphics g)` de `JPanel` para que **utilizando Java2D** pinte el tablero de juego. El tamaño del tablero puede cambiar y depende del valor seleccionado en el `JComboBox` del panel superior que fue implementado en el punto anterior. A continuación se muestra un ejemplo de un tablero 5x5.



4. En el paquete `src/uniandes.dpoo.taller7.interfaz4` copie y pegue las clases que implementó en el punto anterior. Modifique tanto los páneles, como la ventana principal, para personalizar su apariencia. Puede cambiar, tamaño, color, tema, entre otros.

5. Defina cómo va a hacer para reaccionar a las acciones del usuario: asígneles a las diferentes clases las responsabilidades relacionadas con reaccionar a los eventos del usuario y realizar las acciones adecuadas. Para el panel del tablero de juego le recomendamos usar `MouseListener` y para el panel de la derecha `ActionListener`.

6. Implemente los métodos de respuesta para que todos los elementos funcionen correctamente:

6a. Al hacer clic en **NUEVO**, se debe tomar el tamaño y la dificultad del panel superior para crear un nuevo tablero y mostrarlo en el panel de juego.

6b. Al hacer clic en **REINICIAR** se debe mostrar el tablero que fue generado al momento de dar **NUEVO**, además el contador de jugadas se reestablece en 0.

6c. Al hacer clic en **TOP-10** se debe abrir una nueva venta (`JDialog`) que muestra el top10 de puntajes (no olvide revisar la sección de restricciones),

6d. Al hacer clic en **CAMBIAR JUGADOR** se le debe mostrar al usuario la opción de ingresar su nombre para que se vea reflejado en el panel inferior y se registre el nombre con el puntaje una vez el jugador complete el tablero.

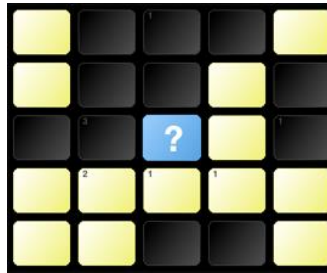
6e. Al hacer clic en cualquier espacio del tablero de juego, se deben cambiar las casillas según las reglas del juego.

7. Construya un diagrama de clases para la aplicación entera: analice la forma en la que asignó las responsabilidades y haga ajustes al diseño y la implementación para corregir problemas que haya descubierto durante su análisis.

Restricciones

1. El tablero de juego **DEBE** crearse por medio de `Java2D`.

La siguiente imagen muestra otra opción de cómo podría verse el tablero de juego, pero ustedes podrían tomar sus propias decisiones estéticas y funcionales:



En la figura mostrada las casillas se están dibujando como rectángulos con esquinas redondeadas (`fillRoundRect`), el color de las casillas se está estableciendo con gradientes (`GradientPaint`), el color y el texto de las casillas cambia según el ratón esté o no sobre la casilla (`MouseListener`) y además cada casilla tiene un indicador que muestra cuántas veces ha sido presionada.

Ayuda:

Uno de los aspectos que pueden crearles problema es saber en qué casilla se hizo clic. Para eso necesitan tener un *listener* sobre el ratón y hacer los cálculos correspondientes para saber a qué casilla corresponden las coordenadas del panel donde se hizo clic.

Como ejemplo, esta es la implementación para la aplicación de donde salió la captura de pantalla.

```
@Override
public void mousePressed(MouseEvent e)
{
    int click_x = e.getX();
    int click_y = e.getY();
    int[] casilla = convertirCoordenadasACasilla(click_x, click_y);
    cantidades[casilla[0]][casilla[1]]++;
    principal.jugar(casilla[0], casilla[1]);
    this.ultima_fila = casilla[0];
    this.ultima_columna = casilla[1];
    repaint();
}

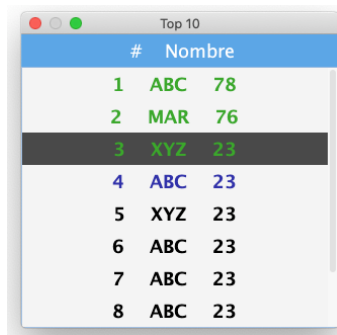
private int[] convertirCoordenadasACasilla(int x, int y)
{
    int ladoTablero = tablero.length;
    int altoPanelTablero = getHeight();
    int anchoPanelTablero = getWidth();

    int altoCasilla = altoPanelTablero / ladoTablero;
    int anchoCasilla = anchoPanelTablero / ladoTablero;
    int fila = (int) (y / altoCasilla);
    int columna = (int) (x / anchoCasilla);

    return new int[] { fila, columna };
}
```

2. La ventana del Top10 **DEBE** usar un elemento de tipo `JList` y tiene que mostrar los resultados de alguna manera personalizada (no puede ser la visualización por defecto de un `JList`).

En la siguiente captura de pantalla se muestra un ejemplo:



#	Nombre
1	ABC 78
2	MAR 76
3	XYZ 23
4	ABC 23
5	XYZ 23
6	ABC 23
7	ABC 23
8	ABC 23

En este caso, cada elemento de la lista se está mostrando como un panel con tres elementos de tipo JLabel, se está cambiando el color de los elementos de acuerdo con la posición, y se está usando una fuente más grande de lo normal y en negrita. Además, la lista está metida dentro de un JScrollPane para que aparezcan barras de desplazamiento a la derecha.

Parte 3: Documentar el diseño

Construya el diagrama de clases para la nueva interfaz.

Entrega

1. Cree una carpeta en el repositorio donde quedará el taller 7 y deje ahí todos los archivos relacionados con su entrega. La entrega debe incluir el proyecto Eclipse con su solución y una imagen con el diagrama de la aplicación completa. Incluya instrucciones para ejecutar la aplicación.
2. Entregue a través de Bloque Neón el URL para el repositorio, en la actividad designada como “**Taller 7**”.