TP número 1

Integrantes: Javier Hernán Monzón.

DNI: 30066008.

Implementación:

**Clase Tpui:**

Metodo *inizialize()*

Se agregó un GridLayout 4x4 por defecto para el frame. Por otro lado se creó los componentes para crear un dialogo al iniciar el juego, con la dificultad y los labels para informar cómo se juega el juego. Luego se creó la clase Tpmodel, donde estará la lógica de nuestro juego.

**Clase Tpmodel**

***public Tpmodel(JFrame frame, GridLayout gridLayout, JComponent[] dialogoComponentes)***

Inicializamos los componentes y cargamos los recursos, en nuestro caso la imagen de la luz prendida y la luz apagada. Y se muestra el dialogo del inicio del juego.

***private void dialogoIniciarJuego()***

Mostramos al usuario las opciones de dificultad, en donde en nuestro caso será las cantidad de luces por lado (4, 6, 8). El usuario acepta o cancela. Y el juego entra en ejecución, sin antes cargar los componentes del juego.

***public void prepararComponentes()***

Obtenemos las opciones que ha elegido el usuario, la dificultad será lo que determine la cantidad de luces para el juego, además modificaremos el Grid de la cuadricula, para que coincida el número de luces en cada posición. También agregaremos al frame las luces creadas, y colocaremos los eventos de click, en cada luz. También colocaremos algunas luces prendidas en forma aleatoria en la cuadricula.

***private void agregarLucesAlFrame(JButton [][] luces, JFrame pFrame )***

Agregamos las luces al frame

***public JButton[][] crearLuces(int lucesPorLado, ImageIcon luzApagada)***

Devuelve un Array de luces creadas, con su posición correspondiente, con un background de color negro y un icono de luz apagada.

***private void agregarLucesAlFrame(JButton [][] luces, JFrame pFrame )***

Agregamos las luces recién creadas al frame.

***private void aniadirEventoDeLuz(JButton [][] luces)***

Colocamos los eventos de los botones, para cuando el usuario hace click en alguna luz. Además comprobamos si el usuario ha terminado de apagar todas las luces.

***public LinkedList<FilaColumna> obtenerLucesAleatorias(JButton[][] luces, int cantidadAleatoria)***

Obtenemos un grupo de luces de forma aleatoria, no repetidos, que serán los que simularemos al simular clicks.

***private void simularClickAleatorios(JButton[][] luces)***

Simulamos clicks o dicho de otra manera prendemos luces en forma aleatoria, el usuario vera como se apagan y prenden las luces, una vez terminada la secuencia, el usuario puede jugar el juego.

***private boolean luzYaSeleccionada(int fila, int columna, LinkedList<FilaColumna> filasColumnas)***

Verifica que no seleccione la misma luz en forma aleatoria, si sucede esto, se genera la siguiente luz aleatoria.

***public void clickLuz(int fila, int columna, JButton [][] luces)***

Se realice un click en la luz, procederemos a apagarla o prenderla, y así con sus luces contiguas.

***public boolean comprobarJuegoGanado(JButton [][] luces)***

Comprobamos si el juego ya se ha ganado, recorriendo las luces y verificar que ya están todas apagadas.

***private void prenderOApagarLuz(int f, int c, JButton [][] luces)***

Apagamos o prendemos la luz, en lo que resulta en cambiar el icono del botón y el fondo, según corresponda, de apagado a prendido y de prendido a apagado.

***private void dialogoReiniciarJuego()***

Dialogo que aparece cuando se reinicia el juego, el usuario puede volver a jugar nuevamente.

***public JSpinner dameSpinner() throws InstanceNotFoundException***

Recorre los componentes del dialogo para encontrar el spinner, donde es la selección de dificultad.

***public ImageIcon cargarImagenes(String nombreImagen)***

Carga las imágenes de los botones de apagado y prendido.

***public boolean esLuzExistente(int fila, int columna, JButton [][] luces)***

Verifica que la luz que queremos apagar realmente existe.

**public boolean esAnchoDeLucesValido(int cantidadLucesLado)**

Verifica que el ancho es un acho de luces horizontal valido, esto también determina la dificultad.

**Clase TpmodelTest**

Posee test importantes para verificar el correcto funcionamiento.

@Before

public void inicializarComponentes()

Inicializamos los componentes que utilizaremos con frecuencia en el test.

@Test(expected = IllegalArgumentException.class)

public void cargarImagenInexistente()

Verifica que cargue una imagen inexistente

@Test

public void cargarImagenExistente()

Vemos que la imagen si exista

@Test

public void comprobarJuegoNoGanado()

Verificamos que el juego no se haya ganado.

@Test

public void comprobarJuegoGanado()

Vemos que si hemos ganado el juego, sin necesidad de jugarlo.

@Test

public void aniadirLucesCantidadNoPermitida()

Añadimos luces que no corresponden al invariante de nuestra aplicación.

@Test

public void aniadirLucesCantidadPermitida()

Añadimos luces que si corresponden al invariante de nuestra aplicación.

@Test

public void esLuzExistente()

Verificamos que la luz que pedimos si exista en el juego.

**Clase FilaColumna**

Clase genérica del tipo Tupla para las filas y columna.