Programación III

Trabajo práctico 1

Autores:

Elías Espinillo 40.663.777

Federico Farias 36.495.959

Profesores:

Javier Marenco

Patricia Bagnes

Contenidos

1 Introducción

El objetivo del trabajo práctico es implementar una aplicación para jugar al juego llamado **Lights Out**:

- Se juega en una grilla de 4x4, y en cada posición se tiene una luz, que puede estar encendida o apagada.
- Inicialmente las luces tienen una combinación aleatoria de encendidos y apagados.
- En cada turno, el jugador hace click sobre una luz, y este click tiene el efecto de cambiar el estado de la luz de la casilla y de las cuatro luces vecinas (ubicadas en las casillas de arriba, abajo, izquierda y derecha).
- El objetivo es lograr que todas las luces de la grilla terminen apagadas.

2 Diseño y especificación

Primeramente, se planteó la creación de 3 packages diferentes, en uno se guardó la lógica del juego o código de negocio. En otro, la vista del juego y en el último, los elementos multimedia (sonidos e imágenes).
En primera instancia, se había creado una matriz de botones fija, para darle un 4x4. También se había elegido que al ejecutarse el programa, se ingresase directamente al juego, y una vez ganado, salga la pantalla de "Ganaste"

Luego, se evolucionó a una matriz definida por el usuario, donde el mismo decide ingresar la cantidad de filas y columnas con las que desea jugar, agregándose también una pantalla de inicio, donde el mismo define la matriz y selecciona para jugar, dándole así 3 vistas para el usuario, una pantalla de inicio, una de juego y una de finalización:

- La de menú de inicio donde el usuario indica como quiere el tablero e iniciarlo.
- La de juego, que es la matriz anteriormente definida por el usuario, donde puede clickear los botones e interactuar con el programa.
- La de victoria, donde se indica el fin del juego, muestra la cantidad de movimientos que realizo y se le da la opción de jugar o de salir del mismo.

En cuanto al código de negocio, se había decido utilizar el Math.random pero observamos que en ocasiones el mismo daba combinaciones que eran imposibles de resolver (esto mismo se chequeo con un corroborador Online, además de revisar el teorema de Gauss sobre matrices), además de que teníamos un error en cómo se creaba el JLabel de cantidad de movimientos, lo que nos impedía actualizar la pantalla correctamente y no permitía la aleatoriedad de los botones del tablero por como lo estábamos implementando
Luego de revisar estos errores decidimos usar la librería Random y crear el JLabel de los movimientos primero, lo cual nos permitió tener siempre soluciones y poder mostrar el cálculo correspondiente

Una vez que el funcionamiento del juego estaba como se requería se mejoro el diseño de la interfaz dándole un aspecto arcade de los 80's tanto desde el lado visual como el audio para pulir y darle fin al trabajo.

- **Paquete comportamiento:**

  - **Clase Flow:**

    - juegoTerminado y noPreparado: Boolean
    - filas, columnas y contador: Integer
    - juego: Tablero
    - musica: Clip

    Operaciones:

    - Constructor del Flow del juego. Se encarga del comportamiento general del juego y define según el estado de este que vista mostrarle al usuario.
    - private void columnasYFilas():

      Se encarga de tomar los datos ingresados del usuario y usarlos en el constructor.

  - **Clase LucesFuera:**

    - Se encarga del main del juego

- **Paquete externalMedia:**

  Contiene los archivos de fondo, fuente y sonidos del juego.

- **Paquete vistas:**

  - **Clase PantallaInicial:**

    - botonInicioJuego, botonColumnas y botonFilas: JButton
    - titulo: Jlabel

Operaciones:

- Constructor de la interfaz de la pantalla inicial.

  Clase nesteada ImagenFondo:

  - Se encarga de pintar el panel de contenido dentro del constructor con una imagen personalizada

- **Clase PantallaGanador:**
  - seguirJugando y salir: JButton
  - cartelGanador y cartelMovFinales: Jlabel

  Operaciones:

  - Constructor de la interfaz de la pantalla final.

    Clase nesteada ImagenFondo:

    - Se encarga de pintar el panel de contenido dentro del constructor con una imagen personalizada

- **Clase Tablero:**
  - botonesDelTablero: matriz de JButton
  - panelDelTablero y panelDePuntaje: JPanel
  - cartelMovimientos: JLabel
  - cantMovimientos: Integer
  - sonidoBoton: Clip

  Operaciones:

  - Constructor del tablero del juego. Crea la matriz con la cantidad de botones ingresada por el usuario, les agrega listener a cada uno y le da aleatoriedad cada vez que se crea uno nuevo.

Clase nesteada Input:

- Se encarga de verificar en el listener las casillas que necesitan cambiarse de color según la acción del usuario sobre cada uno.