Lógica III

Informe Manual de Técnico

Entregado a:
Roberto Florez Rueda
Joan Sebastian Gomez Acevedo

RESPONSABLES:
Daniel Felipe Rivera Arroyave <1152219543>
Mattius Alexander Cardona Franco<1036402365 >

Introducción

En el presente documento se explicará el desarrollo e implementación de la práctica computacional para el curso de Lógica y representación III que consiste en una aplicación desarrollada en java, que realice y muestre la solución de un solitarios inglés clásico o de uno personalizado por el usuario.

Objetivo General:

Resolver un solitario inglés o “senku” ingresado, enseñando los pasos para llegar a la solución y en caso de no tener hacer la aclaración de que no tiene.

Objetivos Específicos:

- Elegir configuración clásica o personalizada.
- Buscar solución al solitario usando la técnica de programación backtracking.
- Enseñar los pasos para solución.
- Reconocer la configuración de “senku” ingresada por el usuario.

Solución por backtracking

El backtracking ocurre en la capa “backend” de la aplicación y en ella se analiza una matriz de 7 filas por 7 columnas, donde las casillas que tengan “0” representan la zona que se encuentra fuera de la cruz de la versión inglesa, los “1” representa las casillas en las que no hay ficha y los “2” las casillas que se encuentran ocupadas por una ficha.

La condición de parada es que solo quede una casilla ocupada por un 2.

Hay un Arraylist que almacena todos los objetos “Posición” que se pueden mover en al menos una dirección, luego para cada objeto de dicho Arraylist se hace sus respectivos saltos y en cada uno hace el llamado recursivo al método backtracking con el cambio realizado en la matriz y una casilla menos hasta llegar a la condición de parada, en caso de no encontrar solución, restaura la matriz a su estado anterior e intenta con el siguiente salto.

Tras realizar todos los saltos de una Posición sin obtener éxito, pasa a la siguiente Posición contenida en el arraylist y repite el proceso anterior.

Si después de ralizar el proceso con todo el Arraylist no encontró solución, quiere

decir que esa configuración de tablero no tiene solución.

Clase Posición

La clase Posición tiene unas coordenadas x e y que representan la ubicación en la matriz 4 booleanos que permiten saber en que dirección se puede mover esta casilla y 1 objeto de la clase Salto para cada dirección.

Clase salto

Se usa para representar la posición actual y la siguiente del una ficha en el tablero, con coordenadas actuales y coordenadas futuras.

Entorno visual

Para el entorno visual usamos la tecnología de JavaFx por su facilidad para la realización de aplicaciones interactivas o multimedia.

La aplicación solo maneja una pantalla o “Scena” que es constantemente modificada por las interacciones del usuarios, usamos una matriz de botones que está ligada a la matriz usada en el backtracking, al seleccionar una de las opciones de juego se va a hacer un cambio visual al “Style” de los botones en la matriz.

Manejamos un booleano que está desactivado en el modo clásico y activado en el personalizado, que nos permite alterar el “Style” del boton apretado y así representarlo como si tuvera o no tuviera una pieza.

El boton Aceptar analiza todas la matriz de botones y se fija en el “Style” de cada una para poner su número respectivo en la matriz del backtracking y proceder a solucionarla, los movimientos respectivos se guardan en un “Stack” que por su estructura “LIFO” guarda un string que contiene los movimientos hechos.

EL botón Aceptar se transforma en el botón Iniciar, y este entra en un Clock que va leyendo las soluciones y las representa de forma grafica cambiando el Style de las casillas involucradas en cada movimiento hasta tener la pila totalmente desopilada, y el juego queda listo para volver a ser usado.