**Tarea Practica (0226): Juego del 15**

**(Universidad de Guanajuato, División de ingenierías campus Irapuato – Salamanca,** [**j.chavezislas@ugto.mx**](mailto:j.chavezislas@ugto.mx)**, Inteligencia Artificial)**

**INTRODUCCION**

El juego del 15 visto en clase es un juego creado en el siglo XIX, el juego consiste en un tablero 4x4 (16 casillas) en la cual solo 15 casillas contienen sus respectivos números y la casilla correspondiente al número 16 está vacía dejando un hueco en el tablero por el cual se deslizaran los números, la misión del juego es revolver los números con n cantidad de movimientos por medio de la casilla vacía (Imagen 1) y posteriormente intentar acomodarlos en orden ascendente dejando el hueco en la casilla inferior derecha (Imagen 2)

Una calculadora científica

Descripción generada automáticamente con confianza media Una calculadora científica

Descripción generada automáticamente con confianza media  
 Imagen 1 Imagen 2

**METODOLOGIA**

¿Cómo podemos llegar a esta solución?

¿Cómo podemos desarrollar este programa?

Para el desarrollo de la aplicación, consideré por cuenta propia hacerla más dinámica, es decir, que el tablero no sea solo de 4x4 sino que pueda hacerse de cualquier cantidad nxn deseada por el usuario (siendo recomendable no más que 8) podemos recurrir a crear una matriz bidimensional, en este caso nxn, donde se llenan todas las casillas de la misma de forma ascendente exceptuando la última casilla que la reemplazamos con un número 0 para representar la casilla vacía, entonces creamos un método para revolver el tablero y es donde comienza el juego, se le pide de manera indefinida (hasta que el jugador complete el tablero exitosamente) un movimiento a ingresar arriba, abajo, izquierda o derecha (w,s,a,d, respectivamente) y es hacia la dirección donde la casilla 0 (situado en una posición al azar) y la casilla en la dirección digitada cambian de lugar, intercambiando sus coordenadas mutuamente (a lo que se le consideraría deslizar la casilla) para que al final si es que todas las casillas están ordenadas correctamente, se termina el juego dándole la victoria al jugador.

Utilizando java se consideran 3 clases, la clase del tablero la cual tendrá los atributos y los métodos de movimiento, la clase la cual inicializara el tablero y comenzara el juego, y la clase main que es la que correrá el juego.

**Código utilizado:**

*package* org.example;  
  
*import* java.util.Random;  
*import* java.util.Scanner;  
  
*public class* Tablero {  
 *private int*[][] matriz;  
  
 *public* Tablero(*int* n)  
 {  
 matriz = *new int*[n][n];  
 *int* contador = 1;  
 *for* (*int* i = 0; i < n; i++)  
 {  
 *for* (*int* j = 0; j < n; j++)  
 {  
 matriz[i][j] = contador;  
 *if* (contador == n \* n)  
 {  
 matriz[i][j] = 0;  
 }  
 contador++;  
 }  
 }  
 }  
 *public void* imprimirTablero() {  
 *for* (*int* i = 0; i < matriz.length; i++) {  
 *for* (*int* j = 0; j < matriz[i].length; j++) {  
 System.***out***.printf("%4d", matriz[i][j]);  
 }  
 System.***out***.println();  
 }  
 }  
 *public void* revolverTablero()  
 {  
 Random rand = *new* Random();  
 *for*(*int* i = matriz.length -1; i>0; i--)  
 {  
 *for*(*int* j = matriz[i].length -1; j > 0; j--)  
 {  
 *int* m = rand.nextInt(i+1);  
 *int* n = rand.nextInt(j+1);  
  
 *int* temp = matriz[i][j];  
 matriz[i][j] = matriz[m][n];  
 matriz[m][n] = temp;  
 }  
 }  
 }  
 *public void* mover(String direccion)  
 {  
 *int* fila = 0;  
 *int* columna = 0;  
 *for*(*int* i = 0; i < matriz.length;i++)  
 {  
 *for*(*int* j = 0; j < matriz[i].length; j++)  
 {  
 *if*(matriz[i][j] == 0)  
 {  
 fila = i;  
 columna = j;  
 }  
 }  
 }  
  
 *if*(direccion.equals("W") || direccion.equals("w"))  
 {  
 *if*(fila -1 >=0)  
 {  
 *int* temp = matriz[fila][columna];  
 matriz[fila][columna] = matriz[fila-1][columna];  
 matriz[fila-1][columna] = temp;  
 }  
  
 }  
 *else if*(direccion.equals("S") || direccion.equals("s"))  
 {  
 *if*(fila +1 < matriz.length)  
 {  
 *int* temp = matriz[fila][columna];  
 matriz[fila][columna] = matriz[fila+1][columna];  
 matriz[fila+1][columna] = temp;  
 }  
 }  
 *else if*(direccion.equals("A") || direccion.equals("a"))  
 {  
 *if*(columna -1 >=0)  
 {  
 *int* temp = matriz[fila][columna];  
 matriz[fila][columna] = matriz[fila][columna-1];  
 matriz[fila][columna-1] = temp;  
 }  
 }  
 *else if*(direccion.equals("D") || direccion.equals("d"))  
 {  
 *if*(columna +1 < matriz.length)  
 {  
 *int* temp = matriz[fila][columna];  
 matriz[fila][columna] = matriz[fila][columna+1];  
 matriz[fila][columna+1] = temp;  
 }  
 }  
 *else if*(direccion.equals("trap") || direccion.equals("T") || direccion.equals("t") || direccion.equals("TRAP") || direccion.equals("trap") || direccion.equals("Trap") || direccion.equals("TRAP") || direccion.equals("Trap"))  
 {  
 trap();  
 }  
 }  
  
 *public void* solicitarMovimiento() {  
 System.***out***.println("Ingrese el movimiento (arriba, abajo, izquierda, derecha):");  
 Scanner scanner = *new* Scanner(System.***in***);  
 String movimiento = scanner.nextLine();  
 mover(movimiento);  
 imprimirTablero();  
 }  
  
 *public boolean* verificarGanador()  
 {  
 *int* contador = 1;  
 *for*(*int* i = 0; i < matriz.length; i++)  
 {  
 *for*(*int* j = 0; j < matriz[i].length; j++)  
 {  
 *if*(contador == matriz.length \* matriz.length)  
 {  
 System.***out***.println("¡Felicidades! Has ganado");  
 *return true*;  
 }  
 *if*(matriz[i][j] != contador)  
 {  
 *return false*;  
 }  
 contador++;  
 }  
 }  
 *return false*;  
 }  
 *public void* trap()  
 {  
 System.***out***.println("Elige los 2 numeros que quieres intercambiar");  
 Scanner scanner = *new* Scanner(System.***in***);  
 *int* num1 = scanner.nextInt();  
 *int* num2 = scanner.nextInt();  
 *for*(*int* i = 0; i < matriz.length; i++)  
 {  
 *for*(*int* j = 0; j < matriz[i].length; j++)  
 {  
 *if*(matriz[i][j] == num1)  
 {  
 matriz[i][j] = num2;  
 }  
 *else if*(matriz[i][j] == num2)  
 {  
 matriz[i][j] = num1;  
 }  
 }  
 }  
 }  
}

*package* org.example;  
  
*import* java.util.Scanner;  
  
*public class* JuegoDel15 {  
  
 Tablero tablero;  
 *public void* iniciarJuego() {  
 Scanner scanner = *new* Scanner(System.***in***);  
 System.***out***.println("Ingrese el tamaño del tablero (entre 4 y 8):");  
 *int* n = scanner.nextInt();  
 *while* (n < 4 || n > 8) {  
 System.***out***.println("Tamaño inválido. Ingrese un número entre 4 y 8:");  
 n = scanner.nextInt();  
 }  
 tablero = *new* Tablero(n);  
 System.***out***.println("Tablero inicial: (Guia para mover las piezas)");  
 tablero.imprimirTablero();  
 System.***out***.println("Tablero revuelto:");  
 tablero.revolverTablero();  
 tablero.imprimirTablero();  
  
 *while* (!tablero.verificarGanador()) {  
 System.***out***.println("Ingrese la dirección (w,a,s,d) o salir para terminar el juego:");  
 String direccion = scanner.next();  
 *if* (direccion.equals("salir")) {  
 System.***out***.println("Juego terminado");  
 *break*;  
 }  
 tablero.mover(direccion);  
 tablero.imprimirTablero();  
 }  
  
 }  
}

*package* org.example;  
  
*public class* Main {  
 *public static void* main(String[] args)  
 {  
 JuegoDel15 juego = *new* JuegoDel15();  
 juego.iniciarJuego();

return 0;  
 }  
}

**RESULTADOS**

Poniendo primero de ejemplo un tablero 4x4 tenemos el siguiente resultado del tablero ordenado (como debe ser) y el tablero revuelto

Captura de pantalla de computadora

Descripción generada automáticamente

Dando 1 movimiento a la izquierda el 0 cambiaría de posición con el 12

Texto

Descripción generada automáticamente

En la nueva posición del 0, si damos un movimiento hacia abajo, cambiaría de posición con el 7

Texto

Descripción generada automáticamente

Ahora si damos un movimiento hacia la derecha, intercambiaría posición con el 6

Texto, Calendario

Descripción generada automáticamente

Ahora si damos un movimiento hacia arriba, intercambiaría posición con el 12

Texto, Calendario

Descripción generada automáticamente

Ya que confirmamos que el 0 va correctamente a la posición deseada, veamos que pasa si intentamos salirnos de la matriz, por lo que en la posición actual intentaré dar un movimiento a la derecha

Texto

Descripción generada automáticamente

Confirmando que no sucede nada y se queda en su misma posición, comenzamos a resolver el puzle

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Comprobando así que, si funciona correctamente al completar el puzle dando el mensaje ganador al usuario, y lo mismo con dimensiones más grandes

Texto

Descripción generada automáticamente

Calendario

Descripción generada automáticamente

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Calendario

Descripción generada automáticamente

**ANALISIS**

Habiendo hecho bastantes pruebas sobre distintas matrices nxn y distintos movimientos podemos concluir que se cumplió exitosamente con el objetivo y no solo con el objetivo general que era la realización del juego clásico (4x4) sino que pudimos hacerlo dinámico siendo posible una cantidad alta de nxn números, uno d ellos mayores problemas durante la realización del programa fue cómo detectar los bordes ya que el ejemplo en clase se veía una matriz de 5x5 donde las casillas de las orillas eran valores -1, es decir, que ya no se podría acceder a ellos, creí posible que si mi código iba a ser capaz de hacer de distintas nxn, tendría que ahorrarme las orillas de -1, y aunque costó, lo pude resolver simplemente comparando la posición actual del 0 si fuera mayor a 0, también otro problema que tuve fue el cómo localizar el 0 y al final opté por recorrer todos los elementos de la matriz cada movimiento.

**DISCUSION**

Buscando en internet mas soluciones al mismo problema di que, la mayoría utiliza interfaces en java, por lo que la principal diferencia en cuanto a lógica de otras soluciones y mi solución es el código escrito durante el proceso, siendo aun así que la solución y la lógica detrás de los movimientos es exactamente la misma, al igual que la comprobación de bordes, mi programa cumple con los requisitos y expectativas establecidas al inicio del proyecto siendo que veo como área de mejora el implementar una interfaz, así como funciones extra como un contador de tiempo para resolver cada puzle, contabilizador de movimientos y un sistema de juego mas amigable con el usuario.