Universidad de los Andes

Ingeniería de Sistemas y Computación

CupiTaller

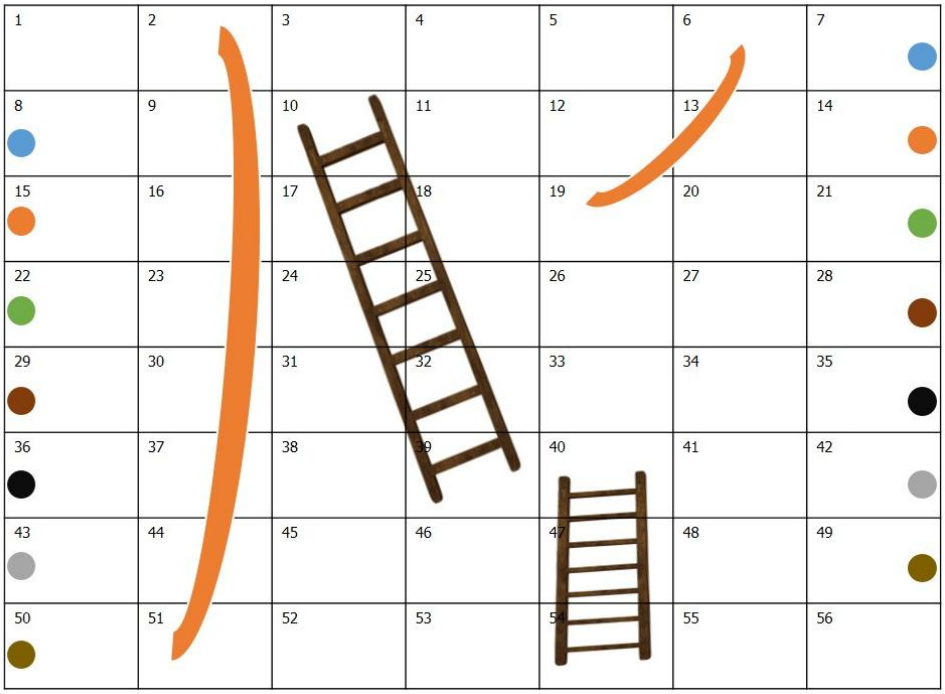
Guía Semana 15.
Noviembre 4 – Noviembre 7

Proyecto Cupi2

Sección 1 – Estructuras contenedoras en 2D (Realizar en hojas blancas)

Escenario:

Suponga el juego de escalera, donde se tiene una ficha y se debe recorrer el tablero hasta llegar a la casilla final. En este juego se tienen escaleras y toboganes, las escaleras ubican al jugador en una posición posterior y los toboganes en una posición anterior a la que se encuentra en ese momento. En una casilla puede haber un inicio de escalera o un inicio de tobogán, también puede que la casilla no tenga ninguno de estos dos elementos. Además, cada casilla tiene el número de guía del recorrido de la ficha (los círculos de colores sólo indican a donde debe saltar la ficha después de esa casilla). El tablero está modelado como una matriz (arreglo en 2D).



Etapa 1: Concepto de arreglos en 2 dimensiones.

1. Complete la siguiente lista que representa algunas de las casillas de la matriz del tablero de juego. La **posición** indica el lugar de esa casilla dentro de la matriz (arreglo en 2D) que representa el tablero:

- **Casilla 1**
  - Número casilla: 1
  - Escalera o tobogán: No
  - Posición: [0] [0]
- **Casilla 2**
  - Número casilla: 2
  - Escalera o tobogán: No
  - Posición: [0] [1]
- **Casilla 10**
  - Número casilla:
  - Escalera o tobogán:
  - Posición: [ ] [ ]
- **Casilla 33**
  - Número casilla: 33
  - Escalera o tobogán: No
  - Posición: [ ] [ ]
- **Casilla 48**
  - Número casilla:
  - Escalera o tobogán:
  - Posición: [ ] [ ]
- **Casilla 51**
  - Número casilla: 51
  - Escalera o tobogán: Inicio tobogán
  - Posición: [ ] [ ]
- **Casilla 56**
  - Número casilla:
  - Escalera o tobogán:
  - Posición: [ ] [ ]

2. Suponiendo un tablero diferente, teniendo en cuenta que el último elemento es el siguiente, ¿cuántas casillas tendría ese tablero?

Casilla XX
- Número casilla: XX
- Escalera o tobogán: No
- Posición: [9] [8]

Etapa 2: Creación e inicialización de matrices en Java

1. Cree el atributo que modela la matriz de casillas en la clase **JuegoEscalera** e inicialícelo. Recuerde también crear las constantes que modelan el tamaño de la matriz. (Las casillas son de la clase **Casilla**)

```java
/**
 * Clase que representa la clase principal del juego de escalera
 *
 */
public class JuegoEscalera
{

	//-----------------------------------------------------------------------
	// Constantes
	//-----------------------------------------------------------------------


	//-----------------------------------------------------------------------
	// Atributos
	//-----------------------------------------------------------------------


	//-----------------------------------------------------------------------
	// Constructores
	//-----------------------------------------------------------------------


	/**
	 * Método constructor de la clase JuegoEscalera
	 */
	private JuegoEscalera()
	{

	}

}
```

Etapa 3: Concepto de ciclos en matrices (arreglos 2D)

Nota: Para los siguientes puntos plantee el ciclo (plantéelo en términos de los índices de recorrido) y describa los pasos que se deben seguir dentro del ciclo. Se recomienda para la realización guiarse del tablero de juego que se encuentra en la descripción del escenario.

1. Describa los pasos que debe seguir para conocer el número de toboganes que tiene el juego.

2. Describa los pasos que debe seguir para conocer donde se encuentra la primera escalera en el tablero.

3. Describa los pasos que debe seguir (en cualquier ubicación) para conocer cuántos toboganes hay delante de la posición de su ficha.

4. Describa los pasos que debe seguir (en cualquier ubicación) para conocer dónde se encuentra la siguiente escalera desde la ubicación de su ficha.

5. Describa los pasos que debe seguir para conocer si su ficha está en una fila peligrosa (fila en la que haya al menos un tobogán).

Sección 2 – Estructuras contenedoras en 2D (Realizar sobre el ejercicio buscaminas)

Agregar una nueva funcionalidad al juego de buscaminas, la cual permita tener casillas con ayudas. De esta manera, cuando el jugador destape una casilla que tenga una ayuda, automáticamente se marcará una casilla que tenga una mina y que no haya sido marcada previamente por el jugador.

Para ello, primero proponga la lista de cambios que deberían hacerse sobre el código y luego implemente dichos cambios en el proyecto, de modo que al final pueda probar el resultado de su trabajo. Si tiene problemas para implementar la nueva funcionalidad, puede revisar la solución propuesta que se indica a continuación:

SOLUCIÓN PROPUESTA (revísela una vez haya intentado su propia solución)

En la clase Casilla:

1. Agregar una constante para identificar el estado AYUDA de una casilla.

```
public static final int AYUDA = 5;
```

2. En el método `public void cambiarEstado( int nuevoEstado )` agregar la opción para que la casilla tenga el nuevo estado AYUDA.

```
case AYUDA:
    estado = AYUDA;
    break;
```

En la clase CampoMinado:

1. Agregar un atributo para guardar el número de ayudas que tendrá el campo de juego.

```
private int numeroAyudas;
```

2. En el método `public CampoMinado( File arch ) throws Exception` leer del archivo de propiedades el número de ayudas que tendrá el campo de juego.

```
String strNumeroAyudas = datos.getProperty( "buscaminas.ayudas" );
numeroAyudas = Integer.parseInt( strNumeroAyudas );
```

3. En el método `public int destapar( int x, int y ) throws Exception` agregar la opción para otorgar una ayuda cuando la casilla destapada tenga el estado AYUDA.

```
if(resultadoJugada == CONTINUA_JUEGO && casillasCampoMinado[ x ][ y ].darEstado( ) == Casilla.AYUDA)
{
    darAyuda();
}
```

4. Crear el método `public void darAyuda()` para recorrer el campo de juego y marcar la primera mina encontrada que no esté marcada.

```
public void darAyuda()
{
    boolean minaMarcada = false;

    // Marcar mina.
    for( int i = 0; i < filas && !minaMarcada; i++ )
    {
        for( int j = 0; j < columnas && !minaMarcada; j++ )
        {
            if( casillasCampoMinado[ i ][ j ].darEstado( ) == Casilla.MINADA )
            {
                try
                {
                    marcar(i, j);
                    minaMarcada = true;
                }
                catch(Exception e)
                {

                }
            }
        }
    }
}
```

5. En el método `public void inicializarJuego( )` distribuir las ayudas en el campo de juego de manera similar como se distribuyen las minas. No se puede ubicar una ayuda en una casilla que ya tenga una mina.

```
int numeroAyudasRestantes = numeroAyudas;
while( numeroAyudasRestantes > 0 )
{
    // Se genera aleatoriamente la fila
    int x = generarNumeroAleatorioEnRango( filas );

    // Se genera aleatoriamente la columna
    int y = generarNumeroAleatorioEnRango( columnas );

    if(casillasCampoMinado[ x ][ y ].darEstado() != Casilla.MINADA)
    {
        casillasCampoMinado[ x ][ y ].cambiarEstado( Casilla.AYUDA );
    }

    numeroAyudasRestantes--;
}
```

6. En el archivo buscaminas.properties, definir el número de ayudas que tendrá el campo de juego.

```
buscaminas.minas=10
buscaminas.filas=9
buscaminas.columnas=9
buscaminas.ayudas=30
```