**Lógica**

#gui: GUI

#enemigos: [] Enemigo

#jugador: Jugador

#nivelMapa: entero

#mapa: Mapa

*<<Constructor>>*

+Logica(gui:GUI)

*<<Comandos>>*

-generarNuevoMapa()

+moverJugador(carácter:dirección)//I,A,D,B falso si no puede moverse, verdadero si se movio

+**jugadorDispara():boolean**

**Mapa**

#matriz: [][] Celda

<<Constructor>>

+Mapa(archivo: string, gui:GUI)

<<Comandos>>

+concretarMovimientoTanque(vieja,nueva:Celda)

+moverDisparo(fila,columna: entero)

+protegerAguila()

+cantidadFilas():entero

+cantidadColumnas():entero

+getCelda(fila,columna: entero) :Celda

**Celda**

#fila:entero

#columna:entero

#obstaculo: Obstaculo

#bala: Disparo

#powerUp: PowerUp

#tanque:Tanque

<<Constructor>>

+Celda(fila,columna: entero)

<<Comandos>>

+setObstaculo(obsta: Obstaculo)

+setTanque(tanque:Tanque)

+setBala(disparo:Disparo)

+setPower(power: PowerUp)

+piso()

<<Consultas>>

+getColumna():entero

+getFila():entero

+permitidoAvanzarTanque():boolean

+getTanque():Tanque

+getObstaculo(): Obstaculo

+getBala() : Disparo

+getPower(): PowerUp

**ObjetoConImagen\***

#celda: Celda

#tamaño: entero

#imagenes: Icon []

#imagenActual: JLabel

<<Constructor>>

+ObjetoConImagen(tamaño:entero, celda:Celda)

≪Comandos≫

+cambiarImagenActual(índice: entero)

<<Consultas>>

+getTamaño():entero

+getImagenActual(): JLabel

**Tanque \***

#velocidadMovimiento: entero

#velocidadDisparo: entero

#resistencia: entero

#golpesRecibidos: entero

#disparosRealizados: LinkedList<Disparo>

#direccion: char

<<Constructor>>

+Tanque(vm, vd, resis: entero, direc:char,celda:Celda)

*<<Comando>>*

+disparar(): Disparo

+eliminarDisparo(dis:Disparo)

+recibirGolpe():boolean

+setDireccion(dir: char)

*<<Consultas>>*

+getVelocidadMovimiento(): entero

+getVelocidadDisparo(): entero

+getDireccion(): char

**Enemigo\***

#puntos: entero

≪Constructor≫

+Enemigo(puntos:int, celda:Celda)

*<<Consultas>>*

+getPuntos(): entero

**Blindado**

*<<Constructor>>*

+Blindado(celda:Celda)

**DePoder**

*<<Constructor>>*

+DePoder (celda:Celda)

**Rápido**

*<<Constructor>>*

+Rápido(celda:Celda)

**Básico**

*<<Constructor>>*

+Básico(celda:Celda)

**Jugador**

#puntos: entero

#vidas: entero

#nivel: entero

#enemigosDestruidos: entero

#invulnerabilidad: boolean

#disparosSimultaneos: entero

*<<Constructor>>*

+Jugador(celda:Celda)

<<Comandos>>

+reiniciarDestruidos()

+setInvulnerabilidad(estado: boolean)

+aumentarPuntos(puntos: entero)

+aumentarNivel() (cuando agarra la estrellita)

+aumentarVida()

+revivir():boolean

*<<Consultas>>*

+getVidas(): entero

+getNivel(): entero

+getPuntos(): entero

+getInvulnerabilidad(): boolean

+getEnemigosDestruidos(): entero

**Disparo**

#emisor: Tanque

#direccion: char

<<Constructor>>

+Disparo(Tanque: emisor, char: dirección, celda:Celda)

<<Comandos>>

+Destruirse()

<<Consultas>>

+getEmisor(): Tanque

+getDireccion(): char

**PowerUp\***

<<Constructor>>

+PowerUp(celda:Celda)

<<Comandos>>

+actuar(tanque:Jugador):boolean

+actuar(tanque:Enemigo):boolean

**Granada**

<<Constructor>>

+Granada(celda:Celda)

**Casco**

<<Constructor>>

+Casco(celda:Celda)

**Pala**

<<Constructor>>

+Pala(celda:Celda)

**Estrella**

<<Constructor>>

+Estrella (celda:Celda)

**TanquePower**

<<Constructor>>

+Tanque(celda:Celda)

**Timer**

<<Constructor>>

+Timer(celda:Celda)

**Obstaculo\***

#atraviesanTanques: boolean

**#atraviesanDisparos: boolean**

≪Constructor≫

+Obstactulo(at, ad: boolean, celda:Celda)

*<<Comandos>>*

+recibirGolpe():boolean //si le pegue al águila retorna true, si no, false

*<<Consultas>>*

+atraviesanTanques(): boolean

+atraviesanDisparos(): boolean

**Piso**

<<Constructor>>

+Piso(celda:Celda)

**Águila**

<<Constructor>>

+Águila(celda:Celda)

**Ladrillo**

#golpesRecibidos:entero

<<Constructor>>

+Ladrillo(celda:Celda)

**Acero**

<<Constructor>>

+Acero(celda:Celda)

**Árbol**

<<Constructor>>

+Árbol(celda:Celda)

**Agua**

<<Constructor>>

+Agua(celda:Celda)

**PSEUDOCODIGO PARA LA GENERACION DEL MAPA: (sufrio cambios)**

Entrada: Archivo .txt con el siguiente formato:

🡪En la primera línea se indica la cantidad de filas (m)y en la segunda la cantidad de columnas (n).

🡪Habrá m líneas seguidas de nsímbolos separados con un espacio.

🡪Significado de los símbolos:

L: Pared de ladrillo.

A: Pared de acero.

B: Árboles.

H: Agua.

F: Águila.

P: Piso.

int n, m <- números de la primer línea

matriz <- new Celda[m][n]

a <- Nuevo character[n]

Para j de 0 a m-1 hacer

a <- Split de la línea j del archivo

para i de 0 a n-1 hacer

c <- a[i]

si c:

es ‘L’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Ladrillo())

es ‘A’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Acero())

es ‘B’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Árbol())

es ‘H’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Agua())

es ‘F’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Águila())

es ‘P’ entonces matriz[j][i] <- nuevo Celda(j,i, nuevo Piso())

**MOVIMIENTO DEL JUGADOR:**

**EN LA CLASE LÓGICA**

Entrada: un carácter cA,B,D o I

i<- jugador.getFila()

j<- jugador.getColumna()

cantidadColumnas<-mapa.cantidadColumnas()

cantidadFilas<-mapa.cantidadFilas()

Si c es:

I y j-1>=0 y mapa.celda(i,j-1).permitirAvanzar()entonces {

mapa.concretarMovimientoJugador(i,j-1)

}

D y j+1<cantidadColumnas y mapa.celda(i,j+1).permitirAvanzar()entonces {

mapa.concretarMovimientoJugador(i,j+1)

}

B y i+1<cantidadFilas y mapa.celda(i+1,j).permitirAvanzar() entonces {

jugador.setColumna (i+1)

mapa.concretarMovimientoJugador(i+1,)

retornar verdadero

}

A y i-1>=0 y mapa.celda(i-1,j).permitirAvanzar()entonces {

jugador.setColumna (i-1)

mapa.concretarMovimientoJugador(i-1,j)

retornar verdadero

}

Otro caso: retornar falso.

**MOVIMIENTO DEL ENEMIGO:**

**¿Tiene que estar en la clase lógica o en enemigo? logica**

\*\*\*\*Comentario: 1 = oeste, 2 = este, 3 = sur, 4 = norte.

i<- enemigo.getFila()

j<- enemigo.getColumna()

cantidadColumnas<-matriz.length

cantidadFilas<-matriz[0].length

Si enemigo.getDireccion() es:

1 y j-1>=0 y matriz[i][j-1]=nulo entonces {

enemigo.setFila (j-1)

matriz[i][j]<-nulo

matriz[i][j-1]<-enemigo

}

2 y j+1<cantidadColumnas y matriz[i][j+1]=nulo entonces {

enemigo.setFila (j+1)

matriz[i][j]<-nulo

matriz[i][j+1]<-enemigo

}

3 y i+1<cantidadFilas y matriz[i+1][j]=nulo entonces {

enemigo.setColumna (i+1)

matriz[i][j]<-nulo

matriz[i+1][j]<-enemigo

}

4 y i-1>=0 y matriz[i-1][j]=nulo entonces {

enemigo.setColumna (i-1)

matriz[i][j]<-nulo

matriz[i-1][j]<-enemigo

}

Otro caso: {

Int n

Hacer

n<- entero random entre 1 a 4

Mientras que n=enemigo.getDireccion()

enemigo.setDireccion(n)

Volver a llamar al método de movimiento.

}