Guía Práctica de Laboratorio
Sesión 8: INTRODUCCIÓN A LA PROGRAMACIÓN
MLBC - VACJ - CJFV

CONFIO EN TI es el nuevo juego, en el que cualquier parecido con alguna otra realidad es pura coincidencia.

La historia mas o menos es así: **Thanos** no los mato!!!!, los traslado en el tiempo a otra dimesión. Allí llegaron Vincent y Tony; es una nueva dimensión, cuyo suelo es blanco, blanco tal cual la sal; cada uno esta ubicado en una posición diferente y dentro una zona - por supuesto - no tan amigable, ya que en esta zona Thanos ha ubicado dos poderosos explosivos invisibles en distintos lugares.

Vincent y Tony deben confiar uno en el otro para poder salir de la zona peligrosa. Una vez que los dos están afuera pueden respirar tranquilos; pero si alguno de ellos se encuentra con uno de esos explosivos, éste explota y no solo daña la posición donde está ubicada, sino que tiene una onda expansiva de X posiciones a la redonda, por lo que en el peor de los casos ambos podrían morir; uno por haber pisado el explosivo y el otro por estar al alcance de la onda explosiva.

Como consecuencia de haber sido transportados a esta dimensión, ellos han desarrollado la posibilidad de trasladarse de una posición a otra sin tener que transitar todo el camino.

Debes proveer de un mecanismo por el cual uno de ellos se puede mover a la posición P, pero hay condiciones no negociables:

- Nunca la misma persona se mueve en dos ocasiones seguidas, entonces los movimientos se hacen de manera intercalada
- La excepción a la anterior condición es cuando uno de ellos ha salido de la zona de peligro, entonces el que aun se encuentra en la zona de riesgo es el que se mueve
- Nunca el movimiento es a la misma posición, por ejemplo si Vincent esta en la posición (9, 14) y le toca moverse, y le dices que se mueva a la posición (9, 14), no cuenta como movimiento, por lo que aun le toca turno de moverse.

Una vez que ambos han salido de la zona de riesgo, ya pueden dejar de moverse; es decir termina el juego y ellos ganaron, pero si ambos mueren Thanos gano, en caso de que uno muera y el otro no, pero esté fuera de zona de riesgo, se podría decir que es un empate técnico :D. Se te pide contar cuantos movimientos han sucedido para llegar a la conclusión del juego - si llegas...

Si uno de ellos ha salido de zona de riesgo, éste ya no se mueve, entonces todos los movimientos lo debe hacer sólamente el que quedó en la zona de riesgo. Pero aún asi corre riesgo por la onda expansiva del explosivo :(

Imagina que la región de riesgo es una porción rectangular que esta definida por la esquina superior izquierda y la esquina inferior derecha.

Cada una de las personas esta ubicada inicialmente en la zona de riesgo.

Los explosivos también están en la zona de riesgo; cada explosivo ademas de tener una ubicación denotada por una posición, tiene el dato de la onda expansiva en caso de explotar.

Por ejemplo:

- si se tiene la zona de riesgo determinada por (3,10) y (15,1);
- el explosivo exp1 tiene ubicación (7,6) y onda expansiva 5,
- el explosivo exp2 tiene ubicación (11, 7) y onda expansiva 2,
- la persona per1 en la posición (7, 7)
- la persona per2 en la posición (4, 2)

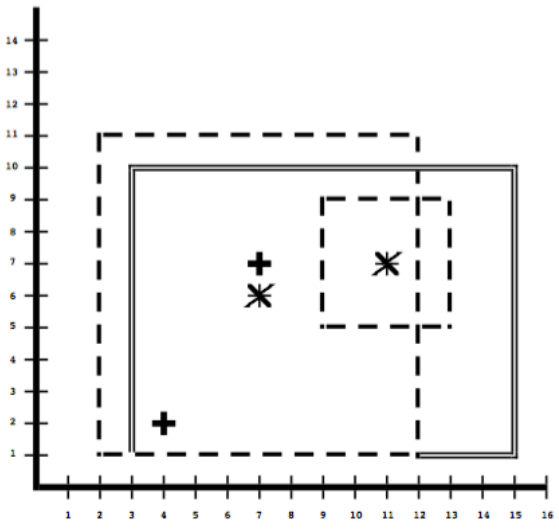

Figura 1: Ejemplo del Juego

En la figura 1 se muestra un ejemplo de la situación descrita.
Aun estando fuera de la zona de riesgo se puede morir :S
Si se solicita mover a la posición (7, 11) la per1 se ubicaría en la (7, 11) para todos los efectos esta fuera de zona de riesgo.
Si se solicita mover a la posición (5, 6), la per2 estaria en la (5,6)
Si se solicita mover a la posición (7, 6) ambas personas mueren por la onda expansiva.
Cada que se hace un movimiento debes informar:

- Si no hubo movimiento
- Si efecto del movimiento, alguien abandono la zona de riesgo
- Si alguien murió.
- Si ambos murieron
- Si ambos están fuera de peligro
- Si hubo movimiento

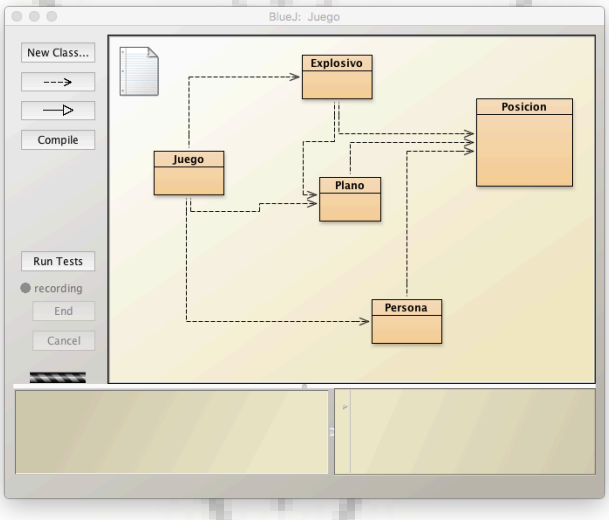

Figura 2: Modelo de la solución

Considera el modelo de la figura 2, como un modelo alternativo de solución, con el código base dado a continuación:

```
1  public class Juego{
2    private Persona   per1, per2;
3    private Plano     zonaRiesgo;
4    private Explosivo exp1, exp2;
5    private int       contadorMov;
6    private boolean   turno;
7                      // true si le toca a la per1
8                      // false si el toca a la per2
9    private boolean   estadoJuego;
10                     // true si el juego sigue activo
11                     // false si el juego termino con cualquier resultado
12
13   public Juego(Posicion esqIzq, Posicion esqDer,
```

```
14                  Persona per1,     Persona per2,
15                  Explosivo exp1,   Explosivo exp2,){
16    }
17    public String mover(Posicion pos){
18        //implementar el mover a una persona a la posicion pos
19        //segun descripcion del problema
20    }
21    public String getEstado(){
22        //indica el estado del juego
23        //si termino y con que resultado
24        //si continua
25    }
26    public int getContadorMov(){
27        //reporta el numero de movimientos en el juego
28    }
29    public void inicializar(){
30        //prepara el juego para reiniciar los movimientos
31        //debes dejarlo como estaba al inicio de creacion
32    }
33  }
34
35  public class Explosivo{
36    private int       onda;
37    private Plano     plano;
38                      //es el plano producido por la onda expansiva
39    private Posicion ubicacion;
40    private boolean   exploto;
41    public Explosivo(Posicion ubicacion, int onda){
42        //implementa
43    }
44    private void definirPlano(){
45        //define el plano de expansion en caso de explosion
46    }
47    public boolean esUbicacion(Posicion posicion){
48        //permite definir si la posicion corresponde a la ubicacion
49    }
50    public boolean estaEnPlano(Posicion posicion){
51        //determina si la posicion esta en el plano de expansion
52        //del explosivo
53    }
54    public void explotar(){
55        //permite explotar el explosivo
56    }
57    public boolean exploto(){
58        //indica si el explosivo exploto
59    }
60  }
61
62  public class Plano{
63    private Posicion esqSupIzq, esqInfDer;
64    public Plano(Posicion pos, int alcance){
65        //construye un plano cuyo punto central es pos
66        //y los contornos definidos a alcance de distancia
67        //del centro
68    }
69    public Plano(Posicion esqIzq, Posicion esqDer){
70        //construye un plano con las dos posiciones
71        //para sus esquinas
72    }
73    public boolean estaDentro(Posicion posicion){
74        //indica si la posicion esta dentro del plano
75    }
76  }
77
78  public class Posicion{
```

```
79    private int posX, posY;
80    public Posicion(int posX, int posY){
81        //implementar
82    }
83    public boolean estaDerAba(Posicion otra){
84        //verifica si la otra posicion esta a la
85        //derecha y abajo de esta posicion
86    }
87    public boolean estaIzqArr(Posicion otra){
88        //verifica si la otra posicion esta a la
89        //izquierda y arriba de esta posicion
90    }
91    public int getX(){
92        //devuelve la posicion en X
93    }
94    public int getY(){
95        //devuelve la posicion en Y
96    }
97    public boolean equals(Posicion otra){
98        //verifica si la otra posicion es igual a esta
99    }
100 }
101
102 public class Persona{
103   private String   nombre;
104   private Posicion ubicacion;
105   private boolean  murio;
106   public Persona(String nombre, Posicion posicion){
107       //implementar
108   }
109   public boolean mover(Posicion pos){
110       //se mueve a la posicion pos
111       //devuelve falso si la pos es la
112       //ubicacion actual de la persona
113       //caso contrario true
114   }
115   public boolean estaDentro(Plano plano){
116       //indica si la ubicacion de la persona
117       // esta dentro el plano
118   }
119   public boolean murio(){
120       //indica si la persona murio
121   }
122   public boolean fueraZona(){
123       //indica si la persona salio de la zona de riesgo
124   }
125   public Posicion getUbicacion(){
126       //devuelve la ubicacion de la persona
127   }
128 }
```

Implementa:

1. Los constructores de las clases

2. Implementa todos los metodos

Considera para ello la descripción del problema, puedes aumentar mas atributos y/o métodos si crees necesario.

Lee bien el modelo y el esqueleto del código para poder entender la propuesta de solución.