

Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Práctica de Laboratorio N° 11:
Definición de Clases de Usuario
Clase Soldado

Nombre: Jhonatan Benjamin Mamani Céspedes

CUI: 20232188

Link de GitHub: <https://github.com/JBenjamin01/fp2-24b/tree/main/Laboratorio>

Usar como base el diagrama de clases UML siguiente:

Soldado
-nombre -nivelAtaque -nivelDefensa -nivelVida -vidaActual -velocidad -actitud -vive
+Soldado() +atacar() +defender() +avanzar() +retroceder() +serAtacado() +huir() +morir() +setVidaActual() +getVidaActual()

1. Crear 3 constructores sobrecargados.
2. La actitud puede ser defensiva, ofensiva, fuga. Dicha actitud varía cuando el soldado defiende, ataca o huye respectivamente.
3. Al atacar el soldado avanza, al avanzar aumenta su velocidad en 1. Al defender el soldado se para. Al huir aumenta su velocidad en 2. Al retroceder, si su velocidad es mayor que 0, entonces primero para y su actitud es defensiva, y si su velocidad es 0 entonces disminuirá a valores negativos. Al ser atacado su vida actual disminuye y puede llegar incluso a morir.
4. Crear los atributos y métodos extra que considere necesarios.
5. Tendrá 2 Ejércitos. Usar la estructura de datos que considere más adecuada. Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida

autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Nivel de ataque y de defensa son aleatorios [1..5]. Se debe mostrar el tablero con todos los soldados creados (usar caracteres como | _ y otros) y distinguir los de un ejército de los del otro ejército.

Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento.

6. El juego es humano contra humano y consistirá en mover un soldado por cada turno de cada jugador. Se puede mover en cualquier dirección, Ud. deberá darle la coordenada del soldado a mover y la dirección de movimiento, el programa deberá verificar que hay un soldado del ejército que corresponda en dicha posición y que el movimiento es válido (no puede haber 2 soldados del mismo ejército en el cuadrado y no se puede ordenar moverse a una posición fuera del tablero), pidiendo ingresar nuevos datos si no es así. Cuando un soldado se mueve a una posición donde hay un soldado rival, se produce una batalla y gana el soldado siguiendo la siguiente métrica: la suma de los 2 niveles de vida actual de los soldados que luchan son el 100% y se le debe dar la probabilidad correspondiente de vencer para cada soldado (ejemplo S1:5 S2:3, las probabilidades de vencer serían S1:62.5% S2:37.5%) y de acuerdo a dichas probabilidades se decidirá el ganador aleatoriamente. El ganador ocupará dicho cuadrado (se le aumentará su nivel de vida actual en 1) y el perdedor desaparecerá. Para cada batalla se deberá explicar por qué ganó uno de los soldados. Gana el juego quien deje al otro ejército vacío. Después de cada movida se deberá mostrar el tablero con su estado actual. Hacer un programa iterativo.
7. Indique su conclusión sobre como la orientación a objetos facilita en el cumplimiento de nuevos requerimientos de un programa

	A	B	C	D	E	F	G	H	I	J
1							1			
2		5		1						
3							1			1
4				1						
5										
6	2					5		2		
7				4						
8	2									
9										
10						1				1

Clase Soldado.java:

```
1  public class Soldado {
2      private String nombre;
3      private int puntosVida;
4      private int fila;
5      private char columna;
6      private int nivelAtaque;
7      private int nivelDefensa;
8      private int velocidad;
9      private String actitud;
10     private boolean vive;
11
12     // Constructores sobrecargados del Soldado
13     public Soldado(String nombre, int puntosVida, int fila, char columna, int nivelAtaque, int nivelDefensa) {
14         this.nombre = nombre;
15         this.puntosVida = puntosVida;
16         this.fila = fila;
17         this.columna = columna;
18         this.nivelAtaque = nivelAtaque;
19         this.nivelDefensa = nivelDefensa;
20         this.velocidad = 0;
21         this.actitud = "Defensiva"; // Estado inicial
22         this.vive = true; // Aquí dejo que por defecto, el soldado está vivo
23     }
24     public Soldado(String nombre, int puntosVida, int fila, char columna) {
25         this.nombre = nombre;
26         this.puntosVida = puntosVida;
27         this.fila = fila;
28         this.columna = columna;
29         this.nivelAtaque = (int)(Math.random() * 5) + 1;
30         this.nivelDefensa = (int)(Math.random() * 5) + 1;
31         this.velocidad = 0;
32         this.actitud = "Defensiva";
33         this.vive = true;
34     }
35     public Soldado(String nombre) {
36         this(nombre, (int)(Math.random() * 5) + 1, (int)(Math.random() * 10), (char)('A' + (int)(Math.random() * 10)),
37             (int)(Math.random() * 5) + 1, (int)(Math.random() * 5) + 1);
38     }
39     public Soldado() {
40         this("Soldado" + (int)(Math.random() * 100));
41     }
42
43     // Getters
44     public String getNombre() {
45         return nombre;
46     }
47
48     public int getFila() {
49         return fila;
50     }
51     public char getColumna() {
52         return columna;
53     }
54     public int getPuntosVida() {
55         return puntosVida;
56     }
57     public int getNivelAtaque() {
58         return nivelAtaque;
59     }
60     public int getNivelDefensa() {
61         return nivelDefensa;
62     }
63     public int getVelocidad() {
64         return velocidad;
65     }
66     public String getActitud() {
67         return actitud;
68     }
69     public boolean isVivo() {
70         return vive;
71     }
72 }
```

```

72 // Setters
73 public void setFila(int fila) {
74     this.fila = fila;
75 }
76 public void setColumna(char columna) {
77     this.columna = columna;
78 }
79 public void setPuntosVida(int puntosVida) {
80     this.puntosVida = puntosVida;
81 }
82
83 // Métodos del UML de referencia
84 public void atacar() {
85     velocidad += 1;
86     actitud = "Ofensiva";
87     System.out.println(nombre + " ha atacado, su velocidad es ahora " + velocidad);
88 }
89 public void defender() {
90     actitud = "Defensiva";
91     System.out.println(nombre + " está en modo defensivo.");
92 }
93 public void huir() {
94     velocidad += 2;
95     actitud = "Fuga";
96     System.out.println(nombre + " está huyendo, su velocidad es ahora " + velocidad);
97 }
98 public void avanzar() {
99     velocidad += 1;
100    System.out.println(nombre + " avanza, su velocidad es ahora " + velocidad);
101 }
102 public void retroceder() {
103     if (velocidad > 0) {
104         velocidad = 0;
105         actitud = "Defensiva";
106         System.out.println(nombre + " se ha detenido, velocidad actual: " + velocidad);
107     } else {
108         velocidad -= 1;
109         System.out.println(nombre + " ha retrocedido, velocidad negativa: " + velocidad);
110     }
111 }
112 public void serAtacado(int daño) {
113     recibirAtaque(daño);
114 }
115 public void recibirAtaque(int daño) {
116     puntosVida -= daño;
117     if (puntosVida <= 0) {
118         puntosVida = 0;
119         System.out.println(nombre + " ha muerto.");
120     } else {
121         System.out.println(nombre + " ha recibido " + daño + " de daño. Vida restante: " + puntosVida);
122     }
123 }
124 public void morir() {
125     vive = false;
126     System.out.println(nombre + " ha muerto.");
127 }
128
129 @Override
130 public String toString() {
131     return "Nombre: " + nombre
132         + " | Vida: " + puntosVida
133         + " | Ataque: " + nivelAtaque
134         + " | Defensa: " + nivelDefensa
135         + " | Velocidad: " + velocidad
136         + " | Actitud: " + actitud
137         + " | Posición: " + columna + fila
138         + " | Vive: " + (vive ? "Sí" : "No");
139 }
140 }

```

Clase VideoJuego.java

```
1 // LABORATORIO N° 11
2 // AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
3 // TIEMPO: 54 MINUTOS
4 import java.text.DecimalFormat;
5 import java.util.*;
6 public class VideoJuego {
7     public static void main(String[] args) {
8         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<>();
9         ArrayList<Soldado> e1 = new ArrayList<>();
10        ArrayList<Soldado> e2 = new ArrayList<>();
11
12        inicializarTablero(tablero);
13        inicializarEjercitos(e1, e2, tablero);
14
15        mostrarTablero(tablero, e1, e2);
16
17        System.out.println("\nDatos del ejército 1:");
18        mostrarDatosEjercito(e1);
19        System.out.println("\nDatos del ejército 2:");
20        mostrarDatosEjercito(e2);
21
22        Soldado soldadoMayorVidaE1 = obtenerSoldadoMayorVida(e1);
23        Soldado soldadoMayorVidaE2 = obtenerSoldadoMayorVida(e2);
24
25        System.out.println("\nSoldado con mayor vida del ejército 1:\n" + soldadoMayorVidaE1);
26        System.out.println("\nSoldado con mayor vida del ejército 2:\n" + soldadoMayorVidaE2);
27
28        double promedioVidaE1 = calcularPromedioVida(e1);
29        double promedioVidaE2 = calcularPromedioVida(e2);
30
31        System.out.println("\nPromedio de puntos de vida del ejército 1: " + String.format("%.3f", promedioVidaE1));
32        System.out.println("Promedio de puntos de vida del ejército 2: " + String.format("%.3f", promedioVidaE2));
33
34        // Aquí hago unas copias de los ejércitos para el ranking, así evito modificar los originales
35        List<Soldado> r1 = new ArrayList<>(e1);
36        List<Soldado> r2 = new ArrayList<>(e2);
37
38        ordenamientoInsertionSort(r1);
39        System.out.println("\nRanking de soldados del ejército 1 (ordenado por puntos de vida de forma decreciente:");
40        mostrarRanking(r1);
41
42        ordenamientoBubbleSort(r2);
43        System.out.println("\nRanking de soldados del ejército 2 (ordenado por puntos de vida de forma decreciente:");
44        mostrarRanking(r2);
45
46
47        // Inicia el ciclo de juego
48        System.out.println("\nInicia la batalla!");
49        juego(tablero, e1, e2);
50    }
51
52    public static void inicializarTablero(ArrayList<ArrayList<Soldado>> tablero) {
53        for (int i = 0; i < 10; i++) {
54            ArrayList<Soldado> fila = new ArrayList<>();
55            for (int j = 0; j < 10; j++)
56                fila.add(null);
57            tablero.add(fila);
58        }
59    }
60
61    public static void inicializarEjercitos(ArrayList<Soldado> e1, ArrayList<Soldado> e2,
62        ArrayList<ArrayList<Soldado>> tablero) {
63        for (int i = 0; i < 2; i++) {
64            int n = (int) (Math.random() * 10) + 1;
65            for (int j = 0; j < n; j++) {
66                int fila, columna;
67                do {
68                    fila = (int) (Math.random() * 10);
69                    columna = (int) (Math.random() * 10);
70                } while (tablero.get(fila).get(columna) != null);
71
72                String nombre = "Soldado" + (i + 1) + "X" + (j + 1);
73                int puntosVida = (int) (Math.random() * 5) + 1;
74                Soldado soldado = new Soldado(nombre, puntosVida, fila + 1, (char) ('A' + columna));
75
76                if (i == 0) {
77                    e1.add(soldado);
78                    tablero.get(fila).set(columna, soldado);
79                } else {
80                    e2.add(soldado);
81                    tablero.get(fila).set(columna, soldado);
82                }
83            }
84        }
85    }
86 }
```

```

87     public static void mostrarTablero(ArrayList<ArrayList<Soldado>> tablero, ArrayList<Soldado> e1,
88                                     ArrayList<Soldado> e2) {
89         System.out.println("Tablero de la batalla:");
90         + "\nLas unidades del ejército 1 están con sus puntos de vida entre corchetes ([x])."
91         + "\nLas del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):");
92         System.out.println("\n      A   B   C   D   E   F   G   H   I   J");
93         System.out.println();
94         System.out.println("-----");
95
96         for (int i = 0; i < tablero.size(); i++) {
97             System.out.print(i + 1 + "\t| ");
98             for (int j = 0; j < tablero.get(i).size(); j++) {
99                 Soldado soldado = tablero.get(i).get(j);
100                 if (soldado == null)
101                     System.out.print("    | ");
102                 else
103                     if (e1.contains(soldado))
104                         System.out.print("[ " + soldado.getPuntosVida() + " ] | ");
105                     else if (e2.contains(soldado))
106                         System.out.print("< " + soldado.getPuntosVida() + " > | ");
107                     else
108                         System.out.print(soldado.getPuntosVida() + " | ");
109             }
110             System.out.println();
111             System.out.println("-----");
112         }
113     }
114
115     public static void mostrarDatosEjercito(ArrayList<Soldado> ejercito) {
116         for (Soldado s : ejercito)
117             System.out.println(s);
118     }
119
120     public static Soldado obtenerSoldadoMayorVida(ArrayList<Soldado> ejercito) {
121         Soldado mayorVida = null;
122         int maxPuntosVida = Integer.MIN_VALUE;
123         for (Soldado soldado : ejercito)
124             if (soldado.getPuntosVida() > maxPuntosVida) {
125                 maxPuntosVida = soldado.getPuntosVida();
126                 mayorVida = soldado;
127             }
128         return mayorVida;
129     }
130
131     public static double calcularPromedioVida(ArrayList<Soldado> ejercito) {
132         double total = 0;
133         for (Soldado soldado : ejercito)
134             total += soldado.getPuntosVida();
135         return total / ejercito.size();
136     }
137
138     public static void ordenamientoInsertionSort(List<Soldado> ejercito) {
139         int n = ejercito.size();
140         for (int i = 1; i < n; ++i) {
141             Soldado key = ejercito.get(i);
142             int j = i - 1;
143
144             while (j >= 0 && ejercito.get(j).getPuntosVida() < key.getPuntosVida()) {
145                 ejercito.set(j + 1, ejercito.get(j));
146                 j = j - 1;
147             }
148             ejercito.set(j + 1, key);
149         }
150     }
151
152     public static void ordenamientoBubbleSort(List<Soldado> ejercito) {
153         int n = ejercito.size();
154         for (int i = 0; i < n - 1; i++)
155             for (int j = 0; j < n - i - 1; j++)
156                 if (ejercito.get(j).getPuntosVida() < ejercito.get(j + 1).getPuntosVida()) {
157                     Soldado temp = ejercito.get(j);
158                     ejercito.set(j, ejercito.get(j + 1));
159                     ejercito.set(j + 1, temp);
160                 }
161     }
162
163     public static void mostrarRanking(List<Soldado> ejercito) {
164         for (int i = 0; i < ejercito.size(); i++) {
165             Soldado soldado = ejercito.get(i);
166             System.out.println((i + 1) + ".- " + soldado);
167         }
168     }
169
170     public static void juego(ArrayList<ArrayList<Soldado>> tablero, ArrayList<Soldado> e1, ArrayList<Soldado> e2) {
171         Scanner sc = new Scanner(System.in);
172         boolean turnoEjercito1 = true;
173         while (!e1.isEmpty() && !e2.isEmpty()) {

```

```

173     System.out.println(turnoEjercito1 ? "\nTurno del Ejército 1 ---> [x]" : "\nTurno del Ejército 2 ---> <x>");
174
175     // Paso 1: Ingresar coordenada y verificarla antes de solicitar la dirección
176     System.out.print("Ingrese coordenada del soldado a mover (Ej. C5): ");
177     String coordenada = sc.nextLine().toUpperCase();
178
179     // Buscar el soldado en la coordenada ingresada
180     Soldado soldadoSeleccionado = buscarSoldado(tablero, coordenada);
181     if (soldadoSeleccionado == null || (turnoEjercito1 ? !e1.contains(soldadoSeleccionado) : !e2.contains(soldadoSeleccionado))) {
182         System.out.println("Movimiento inválido. No hay soldado en la posición o es del ejército contrario.");
183         continue; // El continue deja volver al inicio del ciclo sin solicitar la dirección de movimiento
184     }
185
186     // Paso 2: Solicitar la dirección si la coordenada es válida
187     System.out.print("Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): ");
188     char direccion = sc.next().toUpperCase().charAt(0);
189     sc.nextLine();
190
191     boolean movimientoExitoso = moverSoldado(soldadoSeleccionado, direccion,
192         tablero, turnoEjercito1 ? e1 : e2, turnoEjercito1 ? e2 : e1);
193     if (!movimientoExitoso) {
194         System.out.println("Movimiento no válido, intente nuevamente.");
195         continue;
196     }
197
198     mostrarTablero(tablero, e1, e2);
199     turnoEjercito1 = !turnoEjercito1; // Cambios de turno
200 }
201
202 System.out.println(e1.isEmpty() ? "¡El Ejército 2 ha ganado!" : "¡El Ejército 1 ha ganado!");
203 sc.close();
204 }
205
206 public static Soldado buscarSoldado(ArrayList<ArrayList<Soldado>> tablero, String coordenada) {
207     int columna = coordenada.charAt(0) - 'A';
208
209     // Este paso se hizo para poder verificar el caso de que hayan dos cifras
210     int fila;
211     if (coordenada.length() == 3) {
212         // Aquí son dos dígitos, por ejemplo "H10"
213         fila = Integer.parseInt(coordenada.substring(1, 3)) - 1;
214     } else {
215         // Y este de solo un dígito, por ejemplo "A5"
216         fila = Character.getNumericValue(coordenada.charAt(1)) - 1;
217     }
218
219     if (fila >= 0 && fila < 10 && columna >= 0 && columna < 10)
220         return tablero.get(fila).get(columna);
221
222     return null;
223 }
224
225 public static boolean moverSoldado(Soldado soldado, char direccion, ArrayList<ArrayList<Soldado>> tablero,
226     ArrayList<Soldado> ejercitoAliado, ArrayList<Soldado> ejercitoEnemigo) {
227     int fila = soldado.getFila() - 1;
228     int columna = soldado.getColumna() - 'A';
229
230     switch (direccion) {
231         case 'W': fila--; break; // Arriba
232         case 'S': fila++; break; // Abajo
233         case 'A': columna--; break; // Izquierda
234         case 'D': columna++; break; // Derecha
235         default: return false;
236     }
237
238     // Primero verifico si el movimiento está dentro de los límites del tablero
239     if (!verificarMovimientoValido(fila, columna, tablero))
240         return false;
241
242     Soldado soldadoEnemigo = tablero.get(fila).get(columna);
243
244     // En caso de que la posición contiene un enemigo, se inicia la batalla
245     if (soldadoEnemigo != null && ejercitoEnemigo.contains(soldadoEnemigo)) {
246         batalla(soldado, soldadoEnemigo, ejercitoAliado, ejercitoEnemigo, tablero);
247     } else if (soldadoEnemigo == null) { // Si no hay enemigo, se va a mover el soldado a la posición
248         tablero.get(soldado.getFila() - 1).set(soldado.getColumna() - 'A', null);
249         tablero.get(fila).set(columna, soldado);
250         soldado.setFila(fila + 1);
251         soldado.setColumna((char) ('A' + columna));
252     } else {
253         return false; // Aquí simplemente sería por un movimiento inválido
254     }
255
256     return true;
257 }
258

```

```

259     public static boolean verificarMovimientoValido(int fila, int columna, ArrayList<ArrayList<Soldado>> tablero) {
260         return fila >= 0 && fila < 10 && columna >= 0 && columna < 10;
261     }
262
263     public static void batalla(Soldado s1, Soldado s2, ArrayList<Soldado> ejercitoAliado,
264                               ArrayList<Soldado> ejercitoEnemigo, ArrayList<ArrayList<Soldado>> tablero) {
265         int vidaTotal = s1.getPuntosVida() + s2.getPuntosVida();
266         double probS1 = (double) s1.getPuntosVida() * 100 / vidaTotal;
267         double probS2 = (double) s2.getPuntosVida() * 100 / vidaTotal;
268
269         DecimalFormat df = new DecimalFormat("#.##"); // Aquí estoy usando un objeto para controlar a 3 decimales
270
271         Random random = new Random();
272         double resultado = random.nextDouble() * 100; // Genera un número entre 0 y 1 para decidir el ganador
273
274         System.out.println("\nBatalla entre " + s1.getNombre() + " y " + s2.getNombre());
275         System.out.println("Las probabilidades de victoria son | " + s1.getNombre() + " : " + df.format(probS1) + "% | y | "
276                             + s2.getNombre() + " : " + df.format(probS2) + "% |");
277
278         if (resultado < probS1) {
279             System.out.println(s1.getNombre() + " ha ganado la batalla contra " + s2.getNombre() + " y ha sumado 1 punto de vida.");
280             ejercitoEnemigo.remove(s2);
281             s1.setPuntosVida(s1.getPuntosVida() + 1);
282
283             tablero.get(s2.getFila() - 1).set(s2.getColumna() - 'A', s1);
284
285             tablero.get(s1.getFila() - 1).set(s1.getColumna() - 'A', null);
286             s1.setFila(s2.getFila());
287             s1.setColumna(s2.getColumna());
288         } else {
289             System.out.println(s2.getNombre() + " ha ganado la batalla contra " + s1.getNombre() + " y ha sumado 1 punto de vida.");
290             ejercitoAliado.remove(s1);
291             s2.setPuntosVida(s2.getPuntosVida() + 1);
292
293             tablero.get(s1.getFila() - 1).set(s1.getColumna() - 'A', null);
294         }
295     }

```

Lo que se modificó principalmente fué la métrica de batalla para considerar las probabilidades de vida de cada soldado, es decir, sus puntos sobre el total, y con un valor aleatorio entre 0 y 100 se determina qué soldado ha ganado, además de esto, también se implementó la adición de 1 punto de vida adicional al derrotar al otro soldado teniendo que agregar un nuevo setter para puntosVida de la clase Soldado.

Consola:

```

PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos
mentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos de la Programación 2 - A\Labor
lsInExceptionMessages' '-cp' 'C:\Users\jhona\AppData\Roaming\Code\User\workspaceStorage\6a132b95d1d2b2d12f4208350ee2a932\re
Tablero de la batalla:
Las unidades del ejército 1 están con sus puntos de vida entre corchetes ([x]).
Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

      A   B   C   D   E   F   G   H   I   J
-----
1  |   |   |   |   |   |   |   |   |   |
-----
2  |   |   |   | [5] |   |   |   |   |   | [4] |
-----
3  |   |   | [5] |   |   | <5> |   |   |   |   |
-----
4  |   | <4> | <5> |   |   |   | [2] |   |   |   |
-----
5  |   |   |   |   |   |   |   |   |   |   |
-----
6  |   |   |   |   |   |   |   |   |   |   |
-----
7  |   | [4] |   |   |   | <5> |   |   |   |   |
-----
8  |   |   |   |   |   |   |   | [2] |   |   |
-----
9  |   |   | <1> |   | <3> |   | [4] |   |   |   |
-----
10 |   |   |   |   |   |   |   |   |   |   |
-----

```


Nombre: Soldado1X1	Vida: 4	Ataque: 5	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: J2	Vive: Si
Nombre: Soldado1X2	Vida: 4	Ataque: 3	Defensa: 1	Velocidad: 0	Actitud: Defensiva	Posición: G9	Vive: Si
Nombre: Soldado1X3	Vida: 2	Ataque: 5	Defensa: 2	Velocidad: 0	Actitud: Defensiva	Posición: G4	Vive: Si
Nombre: Soldado1X4	Vida: 2	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: H8	Vive: Si
Nombre: Soldado1X5	Vida: 4	Ataque: 2	Defensa: 3	Velocidad: 0	Actitud: Defensiva	Posición: B7	Vive: Si
Nombre: Soldado1X6	Vida: 5	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: C3	Vive: Si
Nombre: Soldado1X7	Vida: 5	Ataque: 4	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: D2	Vive: Si

Nombre: Soldado2X1	Vida: 4	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: B4	Vive: Si
Nombre: Soldado2X2	Vida: 5	Ataque: 4	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: C4	Vive: Si
Nombre: Soldado2X3	Vida: 3	Ataque: 2	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: E9	Vive: Si
Nombre: Soldado2X4	Vida: 5	Ataque: 2	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: F7	Vive: Si
Nombre: Soldado2X5	Vida: 5	Ataque: 5	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: F3	Vive: Si
Nombre: Soldado2X6	Vida: 1	Ataque: 4	Defensa: 1	Velocidad: 0	Actitud: Defensiva	Posición: C9	Vive: Si

Nombre: Soldado1X6 | Vida: 5 | Ataque: 3 | Defensa: 5 | Velocidad: 0 | Actitud: Defensiva | Posición: C3 | Vive: Si

Nombre: Soldado2X2 | Vida: 5 | Ataque: 4 | Defensa: 4 | Velocidad: 0 | Actitud: Defensiva | Posición: C4 | Vive: Si

Promedio de puntos de vida del ejército 2: 3.833

1.-	Nombre: SoldadoIX6	Vida: 5	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: C3	Vive: Si
2.-	Nombre: SoldadoIX7	Vida: 5	Ataque: 4	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: D2	Vive: Si
3.-	Nombre: SoldadoIX1	Vida: 4	Ataque: 5	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: J2	Vive: Si
4.-	Nombre: SoldadoIX2	Vida: 4	Ataque: 3	Defensa: 1	Velocidad: 0	Actitud: Defensiva	Posición: G9	Vive: Si
5.-	Nombre: SoldadoIX5	Vida: 4	Ataque: 2	Defensa: 3	Velocidad: 0	Actitud: Defensiva	Posición: B7	Vive: Si
6.-	Nombre: SoldadoIX3	Vida: 2	Ataque: 5	Defensa: 2	Velocidad: 0	Actitud: Defensiva	Posición: G4	Vive: Si
7.-	Nombre: SoldadoIX4	Vida: 2	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: H8	Vive: Si

1.- Nombre:	Soldado2X2	Vida:	5	Ataque:	4	Defensa:	4	Velocidad:	0	Actitud:	Defensiva	Posición:	C4	Vive:	Si
2.- Nombre:	Soldado2X4	Vida:	5	Ataque:	2	Defensa:	5	Velocidad:	0	Actitud:	Defensiva	Posición:	F7	Vive:	Si
3.- Nombre:	Soldado2X5	Vida:	5	Ataque:	5	Defensa:	4	Velocidad:	0	Actitud:	Defensiva	Posición:	F3	Vive:	Si
4.- Nombre:	Soldado2X1	Vida:	4	Ataque:	3	Defensa:	5	Velocidad:	0	Actitud:	Defensiva	Posición:	B4	Vive:	Si
5.- Nombre:	Soldado2X3	Vida:	3	Ataque:	2	Defensa:	5	Velocidad:	0	Actitud:	Defensiva	Posición:	E9	Vive:	Si
6.- Nombre:	Soldado2X6	Vida:	1	Ataque:	4	Defensa:	1	Velocidad:	0	Actitud:	Defensiva	Posición:	C9	Vive:	Si

Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): S

Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

[illegible]

Turno del Ejército 2 ---> <x>
 Ingrese coordenada del soldado a mover (Ej. C5): C4
 Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): D
 Tablero de la batalla:
 Las unidades del ejército 1 están con sus puntos de vida entre corchetes ([x]).
 Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

	A	B	C	D	E	F	G	H	I	J
1										
2				[5]						[4]
3						<5>				
4		<4>		<6>			[2]			
5										
6										
7		[4]				<5>				
8								[2]		
9			<1>		<3>		[4]			
10										

Turno del Ejército 1 ---> [x]
 Ingrese coordenada del soldado a mover (Ej. C5): D2
 Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): S
 Tablero de la batalla:
 Las unidades del ejército 1 están con sus puntos de vida entre corchetes ([x]).
 Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

	A	B	C	D	E	F	G	H	I	J
1										
2										[4]
3				[5]		<5>				
4		<4>		<6>			[2]			
5										
6										
7		[4]				<5>				
8								[2]		
9			<1>		<3>		[4]			
10										

Turno del Ejército 1 ---> [x]
 Ingrese coordenada del soldado a mover (Ej. C5): D2
 Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): S
 Tablero de la batalla:
 Las unidades del ejército 1 están con sus puntos de vida entre corchetes ([x]).
 Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

	A	B	C	D	E	F	G	H	I	J
1										
2										[4]
3				[5]		<5>				
4		<4>		<6>			[2]			

```
5 | | | | | | | | | | | |
6 | | | | | | | | | | | |
7 | | [4] | | | | <5> | | | | |
8 | | | | | | | | [2] | | | |
9 | | | <1> | | <3> | | [4] | | | |
10 | | | | | | | | | | | |

Turno del Ejército 2 ---> <x>
Ingrese coordenada del soldado a mover (Ej. C5): D4
Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): W

Batalla entre Soldado2X2 y Soldado1X7
Las probabilidades de victoria son | Soldado2X2 : 54.55% | y | Soldado1X7 : 45.45% |
Soldado1X7 ha ganado la batalla contra Soldado2X2 y ha sumado 1 punto de vida.
Tablero de la batalla:
Las unidades del ejército 1 están con sus puntos de vida entre corchetes ([x]).
Las del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):

    A   B   C   D   E   F   G   H   I   J
-----
1 | | | | | | | | | | | |
2 | | | | | | | | | | [4] |
3 | | | | [6] | | <5> | | | | |
4 | | <4> | | | | | [2] | | | |
5 | | | | | | | | | | | |
6 | | | | | | | | | | | |
7 | | [4] | | | | <5> | | | | |
8 | | | | | | | | [2] | | | |
9 | | | <1> | | <3> | | [4] | | | |
10 | | | | | | | | | | | |

Turno del Ejército 1 ---> [x]
Ingrese coordenada del soldado a mover (Ej. C5): █
```

En cada movimiento que genera un enfrentamiento muestra las probabilidades de cada soldado y al vencedor, también no muestra ningún problema en absoluto con las ubicaciones y actualizaciones.

Diagrama de clases UML:

