

Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Práctica de Laboratorio N° 10:
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 que tenga mayor nivel de vida actual. 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.

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

```

VideoJuego.java

```
1 // LABORATORIO N° 10
2 // AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
3 // TIEMPO: 94 MINUTOS
4 import java.util.*;
5 public class VideoJuego {
6     public static void main(String[] args) {
7         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<>();
8         ArrayList<Soldado> e1 = new ArrayList<>();
9         ArrayList<Soldado> e2 = new ArrayList<>();
10
11         inicializarTablero(tablero);
12         inicializarEjercitos(e1, e2, tablero);
13
14         mostrarTablero(tablero, e1, e2);
15
16         System.out.println("\nDatos del ejército 1:");
17         mostrarDatosEjercito(e1);
18         System.out.println("\nDatos del ejército 2:");
19         mostrarDatosEjercito(e2);
20
21         Soldado soldadoMayorVidaE1 = obtenerSoldadoMayorVida(e1);
22         Soldado soldadoMayorVidaE2 = obtenerSoldadoMayorVida(e2);
23
24         System.out.println("\nSoldado con mayor vida del ejército 1:\n" + soldadoMayorVidaE1);
25         System.out.println("\nSoldado con mayor vida del ejército 2:\n" + soldadoMayorVidaE2);
26
27         double promedioVidaE1 = calcularPromedioVida(e1);
28         double promedioVidaE2 = calcularPromedioVida(e2);
29
30         System.out.println("\nPromedio de puntos de vida del ejército 1: " + String.format("%.3f", promedioVidaE1));
31         System.out.println("Promedio de puntos de vida del ejército 2: " + String.format("%.3f", promedioVidaE2));
32
33         // Aquí hago unas copias de los ejércitos para el ranking para evitar modificar los originales
34         List<Soldado> r1 = new ArrayList<>(e1);
35         List<Soldado> r2 = new ArrayList<>(e2);
36
37         ordenamientoInsertionSort(r1);
38         System.out.println("\nRanking de soldados del ejército 1 (ordenado por puntos de vida de forma decreciente:");
39         mostrarRanking(r1);
40
41         ordenamientoBubbleSort(r2);
42         System.out.println("\nRanking de soldados del ejército 2 (ordenado por puntos de vida de forma decreciente:");
43         mostrarRanking(r2);
44
45         // Inicia el ciclo de juego
46         System.out.println("\n\nInicia la batalla!");
47         juego(tablero, e1, e2);
48     }
49 }
50
51 public static void inicializarTablero(ArrayList<ArrayList<Soldado>> tablero) {
52     for (int i = 0; i < 10; i++) {
53         ArrayList<Soldado> fila = new ArrayList<>();
54         for (int j = 0; j < 10; j++)
55             fila.add(null);
56         tablero.add(fila);
57     }
58 }
59
60 public static void inicializarEjercitos(ArrayList<Soldado> e1, ArrayList<Soldado> e2,
61     ArrayList<ArrayList<Soldado>> tablero) {
62     for (int i = 0; i < 2; i++) {
63         int n = (int) (Math.random() * 10) + 1;
64         for (int j = 0; j < n; j++) {
65             int fila, columna;
66             do {
67                 fila = (int) (Math.random() * 10);
68                 columna = (int) (Math.random() * 10);
69             } while (tablero.get(fila).get(columna) != null);
70
71             String nombre = "Soldado" + (i + 1) + "X" + (j + 1);
72             int puntosVida = (int) (Math.random() * 5) + 1;
73             Soldado soldado = new Soldado(nombre, puntosVida, fila + 1, (char) ('A' + columna));
74
75             if (i == 0) {
76                 e1.add(soldado);
77                 tablero.get(fila).set(columna, soldado);
78             } else {
79                 e2.add(soldado);
80                 tablero.get(fila).set(columna, soldado);
81             }
82         }
83     }
84 }
85 }
```

```

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

```

```

178 // Buscar el soldado en la coordenada ingresada
179 Soldado soldadoSeleccionado = buscarSoldado(tablero, coordenada);
180 if (soldadoSeleccionado == null || (turnoEjercito1 ? !e1.contains(soldadoSeleccionado) : !e2.contains(soldadoSeleccionado))) {
181     System.out.println("Movimiento inválido. No hay soldado en la posición o es del ejército contrario.");
182     continue; // El continue deja volver al inicio del ciclo sin solicitar la dirección de movimiento
183 }
184
185 // Paso 2: Solicitar la dirección si la coordenada es válida
186 System.out.print("Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): ");
187 char direccion = sc.next().toUpperCase().charAt(0);
188 sc.nextLine();
189
190 boolean movimientoExitoso = moverSoldado(soldadoSeleccionado, direccion,
191     tablero, turnoEjercito1 ? e1 : e2, turnoEjercito1 ? e2 : e1);
192
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 System.out.println(e1.isEmpty() ? "¡El Ejército 2 ha ganado!" : "¡El Ejército 1 ha ganado!");
202 sc.close();
203 }
204
205 public static Soldado buscarSoldado(ArrayList<ArrayList<Soldado>> tablero, String coordenada) {
206     int columna = coordenada.charAt(0) - 'A';
207
208     // Este paso se hizo para poder verificar el caso de que hayan dos cifras
209     int fila;
210     if (coordenada.length() == 3) {
211         // Aquí son dos dígitos, por ejemplo "H10"
212         fila = Integer.parseInt(coordenada.substring(1, 3)) - 1;
213     } else {
214         // Y este de solo un dígito, por ejemplo "A5"
215         fila = Character.getNumericValue(coordenada.charAt(1)) - 1;
216     }
217
218     if (fila >= 0 && fila < 10 && columna >= 0 && columna < 10)
219         return tablero.get(fila).get(columna);
220
221     return null;
222 }
223
224 public static boolean moverSoldado(Soldado soldado, char direccion, ArrayList<ArrayList<Soldado>> tablero,
225     ArrayList<Soldado> ejercito, ArrayList<Soldado> ejercitoEnemigo) {
226     int fila = soldado.getFila() - 1;
227     int columna = soldado.getColumna() - 'A';
228
229     switch (direccion) {
230         case 'W': fila--; break; // Arriba
231         case 'S': fila++; break; // Abajo
232         case 'A': columna--; break; // Izquierda
233         case 'D': columna++; break; // Derecha
234         default: return false;
235     }
236
237     // Primero verifico si el movimiento está dentro de los límites del tablero
238     if (!verificarMovimientoValido(fila, columna, tablero))
239         return false;
240
241     Soldado soldadoEnemigo = tablero.get(fila).get(columna);
242
243     // En caso de que la posición contiene un enemigo, se inicia la batalla
244     if (soldadoEnemigo != null && ejercitoEnemigo.contains(soldadoEnemigo)) {
245         batalla(soldado, soldadoEnemigo, ejercitoEnemigo, tablero);
246     } else if (soldadoEnemigo == null) { // Si no hay enemigo, se va a mover el soldado a la posición
247         tablero.get(soldado.getFila() - 1).set(soldado.getColumna() - 'A', null);
248         tablero.get(fila).set(columna, soldado);
249         soldado.setFila(fila + 1);
250         soldado.setColumna((char) ('A' + columna));
251     } else {
252         return false; // Aquí simplemente sería por un movimiento inválido
253     }
254
255     return true;
256 }
257
258 public static boolean verificarMovimientoValido(int fila, int columna, ArrayList<ArrayList<Soldado>> tablero) {
259     return fila >= 0 && fila < 10 && columna >= 0 && columna < 10;
260 }
261
262 public static void batalla(Soldado soldado, Soldado soldadoEnemigo, ArrayList<Soldado> ejercitoEnemigo,
263     ArrayList<ArrayList<Soldado>> tablero) {
264     if (soldado.getPuntosVida() > soldadoEnemigo.getPuntosVida()) {
265         System.out.println(soldado.getNombre() + " ha derrotado a " + soldadoEnemigo.getNombre() + "\n");
266         ejercitoEnemigo.remove(soldadoEnemigo);
267
268         // Quita al soldado derrotado
269         tablero.get(soldadoEnemigo.getFila() - 1).set(soldadoEnemigo.getColumna() - 'A', soldado);

```

```

270
271 // Actualiza La posición del soldado que ganó
272 tablero.get(soldado.getFila() - 1).set(soldado.getColumna() - 'A', null);
273 soldado.setFila(soldadoEnemigo.getFila());
274 soldado.setColumna(soldadoEnemigo.getColumna());
275
276 } else {
277     System.out.println(soldadoEnemigo.getNombre() + " ha derrotado a " + soldado.getNombre() + "\n");
278     ejercitoEnemigo.remove(soldado);
279
280 // Saca al soldado derrotado
281 tablero.get(soldado.getFila() - 1).set(soldado.getColumna() - 'A', soldadoEnemigo);
282
283 // Actualiza La posición
284 tablero.get(soldadoEnemigo.getFila() - 1).set(soldadoEnemigo.getColumna() - 'A', null);
285 soldadoEnemigo.setFila(soldado.getFila());
286 soldadoEnemigo.setColumna(soldado.getColumna());
287 }
288 }
289 }

```

Consola:

Para poder probar todo el código implementado, estuve compilando repetidamente hasta obtener un caso fácil para poder mostrar la victoria de uno de los dos jugadores tras derrotar a todos los soldados de dicho ejército:

```

PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos
\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\jhona\AppData\Roaming\Code\User\workspaceStorage\21ad

```

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

Datos del ejército 1:

Nombre	Vida	Ataque	Defensa	Velocidad	Actitud	Posición	Vive
Soldado1X1	2	2	2	0	Defensiva	H1	Si
Soldado1X2	4	5	4	0	Defensiva	F5	Si
Soldado1X3	2	5	3	0	Defensiva	F3	Si
Soldado1X4	2	3	4	0	Defensiva	D3	Si
Soldado1X5	5	1	1	0	Defensiva	A1	Si
Soldado1X6	1	3	5	0	Defensiva	F6	Si
Soldado1X7	2	1	5	0	Defensiva	A5	Si

Datos del ejército 2:

Nombre	Vida	Ataque	Defensa	Velocidad	Actitud	Posición	Vive
Soldado2X1	1	3	2	0	Defensiva	F2	Si

Soldado con mayor vida del ejército 1:

Nombre	Vida	Ataque	Defensa	Velocidad	Actitud	Posición	Vive
Soldado1X5	5	1	1	0	Defensiva	A1	Si

Soldado con mayor vida del ejército 2:

Nombre	Vida	Ataque	Defensa	Velocidad	Actitud	Posición	Vive
Soldado2X1	1	3	2	0	Defensiva	F2	Si

Ranking de soldados del ejército 1 (ordenado por puntos de vida de forma decreciente):

1.-	Nombre: Soldado1X5	Vida: 5	Ataque: 1	Defensa: 1	Velocidad: 0	Actitud: Defensiva	Posición: A1	Vive: Si
2.-	Nombre: Soldado1X2	Vida: 4	Ataque: 5	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: F5	Vive: Si
3.-	Nombre: Soldado1X1	Vida: 2	Ataque: 2	Defensa: 2	Velocidad: 0	Actitud: Defensiva	Posición: H1	Vive: Si
4.-	Nombre: Soldado1X3	Vida: 2	Ataque: 5	Defensa: 3	Velocidad: 0	Actitud: Defensiva	Posición: F3	Vive: Si
5.-	Nombre: Soldado1X4	Vida: 2	Ataque: 3	Defensa: 4	Velocidad: 0	Actitud: Defensiva	Posición: D3	Vive: Si
6.-	Nombre: Soldado1X7	Vida: 2	Ataque: 1	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: A5	Vive: Si
7.-	Nombre: Soldado1X6	Vida: 1	Ataque: 3	Defensa: 5	Velocidad: 0	Actitud: Defensiva	Posición: F6	Vive: Si

Ranking de soldados del ejército 2 (ordenado por puntos de vida de forma decreciente):

1.-	Nombre: Soldado2X1	Vida: 1	Ataque: 3	Defensa: 2	Velocidad: 0	Actitud: Defensiva	Posición: F2	Vive: Si
-----	--------------------	---------	-----------	------------	--------------	--------------------	--------------	----------

¡Inicia la batalla!

Turno del Ejército 1 ---> [x]

Ingrese coordenada del soldado a mover (Ej. C5): H1

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

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

Turno del Ejército 2 ---> <x>

Ingrese coordenada del soldado a mover (Ej. C5): F2

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

Turno del Ejército 1 ---> [x]

Ingrese coordenada del soldado a mover (Ej. C5): G1

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

Soldado1X1 ha derrotado a Soldado2X1

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	[5]									
2							[2]			
3				[2]		[2]				
4										
5	[2]					[4]				
6						[1]				
7										
8										
9										
10										

¡El Ejército 1 ha ganado!

Diagrama de clases UML (hecho con la herramienta PlantUML de VS Code):

