

Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Práctica de Laboratorio N° 18:
Definición de Clases de Usuario
Herencia

Nombre: Jhonatan Benjamin Mamani Céspedes

CUI: 20232188

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

1. En la historia, los ejércitos estaban conformados por diferentes tipos de soldados, que tenían similitudes, pero también particularidades.
2. Basándose en la clase Soldado crear las clases Espadachín, Arquero y Caballero. Las tres clases heredan de la superclase Soldado, pero aumentan atributos y métodos propios, o sobrescriben métodos heredados.
3. Los espadachines tienen como atributo particular "longitud de espada" y como acción "crear un muro de escudos" que es un tipo de defensa en particular.
4. Los caballeros pueden alternar sus armas entre espada y lanza, además de desmontar (sólo se realiza cuando está montando e implica defender y cambiar de arma a espada), montar (sólo se realiza cuando está desmontado e implica montar, cambiar de arma a lanza y envestir). El caballero también puede envestir, ya sea montando o desmontando, cuando es desmontado equivale a atacar 2 veces, pero cuando está montando implica a atacar 3 veces.
5. Los arqueros tienen un número de flechas disponibles las cuales pueden dispararse y se agotan cuando lo hacen.
6. Basarse en los laboratorios anteriores.
7. Realizar el diagrama de clases de UML
8. Tendrá 2 Ejércitos que pueden ser constituidos sólo por espadachines, caballeros y arqueros (usar una estructura de datos por cada tipo de soldado). Cada ejército tendrá n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Espadachin0X1, Arquero1X1, Caballero2X2, etc., un valor de puntos de vida autogenerado aleatoriamente: Espadachín [3..4], Arquero [1..3] y Caballero [3..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado) y valores autogenerados para el resto de atributos. Mostrar el tablero, distinguiendo los ejércitos y los tipos de soldados creados.
9. 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 tenga el turno en dicha posición y que el movimiento es válido (no puede haber 2 soldados del mismo ejército en el cuadrado), 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 basado en la siguiente métrica: la sumatoria de los niveles de vida de los dos soldados que se enfrentan 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% E2:37.5%) y de acuerdo a dichas probabilidades se decidirá el ganador. El ganador ocupará dicho cuadrado (como bono, se le aumentará el 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 sin soldados. Después de cada movida se deberá mostrar el Mapa en su estado correspondiente. Hacer un programa iterativo.



	A	B	C	D	E	F	G	H	I	J
1							E3			
2		A3		A1						
3							E3			A1
4				L2						
5										
6	C5					L1		L2		
7				E4						
8	C5									
9										
10						E4				A1

Para este laboratorio hice una serie de cambios respecto a los laboratorios anteriores presentados y también una modificación en el diseño general del juego, eliminé atributos y métodos que no corresponden a la consigna como los miembros de clase que no se piden o métodos para conocer detalles sobre el ejército como el ranking para, de esta manera, poder controlar mejor el juego con lo que se solicita en la lista de actividades.

Agregué un método de generar nivel en la clase Soldado debido a la mayoría de aleatorios entre 1 y 5, en el caso de las clases hijas sus niveles de vida se calculan al inicializar ejércitos, este último también tiene nuevos métodos similares a los de un arreglo para su gestión, finalmente, está el método de actualizar mapa que limpia todas las celdas y vuelve a colocar los objetos Soldados según sus ubicaciones tras cada turno.

Clase Soldado.java:

```
1 public class Soldado {
2     public static final int MAX_SOLDADOS_POR_EJERCITO = 10;
3     private String nombre;
4     private int nivelVida;
5     private int fila;
6     private char columna;
7     private int nivelAtaque;
8     private int nivelDefensa;
9     private int velocidad;
10    private String actitud;
11    private boolean vive;
12    private int ejercito;
13
14    // Constructores sobrecargados del Soldado
15    public Soldado(String nombre, int nivelVida, int fila, char columna, int nivelAtaque, int nivelDefensa, int ejercito) {
16        this.nombre = nombre;
17        this.nivelVida = nivelVida;
18        this.fila = fila;
19        this.columna = columna;
20        this.nivelAtaque = nivelAtaque;
21        this.nivelDefensa = nivelDefensa;
22        this.velocidad = 0;
23        this.actitud = "Defensiva";
24        this.vive = true;
25        this.ejercito = ejercito;
26    }
27    public Soldado(String nombre, int nivelVida, int fila, char columna, int ejercito) {
28        this.nombre = nombre;
29        this.nivelVida = nivelVida;
30        this.fila = fila;
31        this.columna = columna;
32        this.nivelAtaque = generarNivel();
33        this.nivelDefensa = generarNivel();
34        this.velocidad = 0;
35        this.actitud = "Defensiva";
36        this.vive = true;
37        this.ejercito = ejercito;
38    }
39    public Soldado(String nombre, int fila, char columna, int ejercito) {
40        this.nombre = nombre;
41        this.nivelVida = generarNivel();
42        this.fila = fila;
43        this.columna = columna;
44        this.nivelAtaque = generarNivel();
45        this.nivelDefensa = generarNivel();
46        this.velocidad = 0;
47        this.actitud = "Defensiva";
48        this.vive = true;
49        this.ejercito = ejercito;
50    }
51    public Soldado(String nombre, int nivelVida, int ejercito) {
52        this.nombre = nombre;
53        this.nivelVida = nivelVida;
54        this.fila = 0;
55        this.columna = 'A';
56        this.nivelAtaque = generarNivel();
57        this.nivelDefensa = generarNivel();
58        this.velocidad = 0;
59        this.actitud = "Defensiva";
60        this.vive = true;
61        this.ejercito = ejercito;
62    }
63
64    // Getters
65    public String getNombre() {
66        return nombre;
67    }
68    public int getFila() {
69        return fila;
70    }
71    public char getColumna() {
72        return columna;
73    }
74    public int getNivelVida() {
75        return nivelVida;
76    }
```

```

77     public int getNivelAtaque() {
78         return nivelAtaque;
79     }
80     public int getNivelDefensa() {
81         return nivelDefensa;
82     }
83     public int getVelocidad() {
84         return velocidad;
85     }
86     public String getActitud() {
87         return actitud;
88     }
89     public boolean isVivo() {
90         return vive;
91     }
92     public int getEjercito() {
93         return ejercito;
94     }
95
96     private int generarNivel() {
97         return (int) (Math.random() * 5) + 1;
98     }
99
100    // Setters
101    public void setFila(int fila) {
102        this.fila = fila;
103    }
104    public void setColumna(char columna) {
105        this.columna = columna;
106    }
107    public void setNivelVida(int nivelVida) {
108        this.nivelVida = nivelVida;
109    }
110    public void setNivelAtaque(int nivelAtaque) {
111        this.nivelAtaque = nivelAtaque;
112    }
113    public void setNivelDefensa(int nivelDefensa) {
114        this.nivelDefensa = nivelDefensa;
115    }
116    public void setEjercito(int ejercito) {
117        this.ejercito = ejercito;
118    }
119
120    // Métodos del UML de referencia
121    public void atacar() {
122        velocidad += 1;
123        actitud = "Ofensiva";
124        System.out.println(nombre + " ha atacado, su velocidad es ahora " + velocidad);
125    }
126    public void defender() {
127        actitud = "Defensiva";
128        System.out.println(nombre + " está en modo defensivo.");
129    }
130    public void huir() {
131        velocidad += 2;
132        actitud = "Fuga";
133        System.out.println(nombre + " está huyendo, su velocidad es ahora " + velocidad);
134    }
135    public void avanzar() {
136        velocidad += 1;
137        System.out.println(nombre + " avanza, su velocidad es ahora " + velocidad);
138    }
139    public void retroceder() {
140        if (velocidad > 0) {
141            velocidad = 0;
142            actitud = "Defensiva";
143            System.out.println(nombre + " se ha detenido, velocidad actual: " + velocidad);
144        } else {
145            velocidad -= 1;
146            System.out.println(nombre + " ha retrocedido, velocidad negativa: " + velocidad);
147        }
148    }
149    public void serAtacado(int daño) {
150        recibirAtaque(daño);
151    }
152    public void recibirAtaque(int daño) {
153        nivelVida -= daño;
154        if (nivelVida <= 0) {
155            nivelVida = 0;
156            System.out.println(nombre + " ha muerto.");
157        } else {
158            System.out.println(nombre + " ha recibido " + daño + " de daño. Vida restante: " + nivelVida);
159        }
160    }

```

```

161     public void morir() {
162         vive = false;
163         System.out.println(nombre + " ha muerto.");
164     }
165
166     @Override
167     public String toString() {
168         return "Nombre: " + nombre
169             + " | Vida: " + nivelVida
170             + " | Ataque: " + nivelAtaque
171             + " | Defensa: " + nivelDefensa
172             + " | Velocidad: " + velocidad
173             + " | Actitud: " + actitud
174             + " | Posición: " + columna + fila
175             + " | Vive: " + (vive ? "Si" : "No");
176     }
177 }

```

Clase Espadachin.java:

```

1  public class Espadachin extends Soldado {
2      private double longitudEspada;
3
4      public Espadachin(String nombre, int nivelVida, int fila, char columna,
5                          int nivelAtaque, int nivelDefensa, int ejercito, double longitudEspada) {
6          super(nombre, nivelVida, fila, columna, nivelAtaque, nivelDefensa, ejercito);
7          this.longitudEspada = longitudEspada;
8      }
9
10     public Espadachin(String nombre, int nivelVida, int fila, char columna, int ejercito, double longitudEspada) {
11         super(nombre, nivelVida, fila, columna, ejercito);
12         this.longitudEspada = longitudEspada;
13     }
14
15     public void crearMuroDeEscudos() {
16         System.out.println(getNombre() + " ha creado un muro de escudos.");
17     }
18
19     @Override
20     public String toString() {
21         return super.toString() + " | Longitud Espada: " + longitudEspada;
22     }
23 }

```

Clase Caballero.java:

```

1  public class Caballero extends Soldado {
2      private boolean montado;
3      private String arma;
4
5      public Caballero(String nombre, int nivelVida, int fila, char columna, int nivelAtaque, int nivelDefensa, int ejercito) {
6          super(nombre, nivelVida, fila, columna, nivelAtaque, nivelDefensa, ejercito);
7          this.montado = true;
8          this.arma = "lanza";
9      }
10
11     public Caballero(String nombre, int nivelVida, int fila, char columna, int ejercito) {
12         super(nombre, nivelVida, fila, columna, ejercito);
13         this.montado = true;
14         this.arma = "lanza";
15     }
16
17     public void montar() {
18         if (!montado) {
19             montado = true;
20             arma = "lanza";
21             System.out.println(getNombre() + " ha montado y cambiado a " + arma + ".");

```

```

22     } else {
23         System.out.println(getNombre() + " ya está montado.");
24     }
25 }
26
27 public void desmontar() {
28     if (montado) {
29         montado = false;
30         arma = "espada";
31         System.out.println(getNombre() + " ha desmontado y cambiado a " + arma + ".");
32     } else {
33         System.out.println(getNombre() + " ya está desmontado.");
34     }
35 }
36
37 public void envestir() {
38     int ataques = montado ? 3 : 2;
39     System.out.println(getNombre() + " realizó una envestida con " + ataques + " ataques.");
40 }
41
42 @Override
43 public String toString() {
44     return super.toString() + " | Montado: " + montado + " | Arma: " + arma;
45 }
46 }

```

Clase Arquero.java:

```

1  public class Arquero extends Soldado {
2      private int flechasDisponibles;
3
4      public Arquero(String nombre, int nivelVida, int fila, char columna,
5                      int nivelAtaque, int nivelDefensa, int ejercito, int flechasDisponibles) {
6          super(nombre, nivelVida, fila, columna, nivelAtaque, nivelDefensa, ejercito);
7          this.flechasDisponibles = flechasDisponibles;
8      }
9
10     public Arquero(String nombre, int nivelVida, int fila, char columna, int ejercito, int flechasDisponibles) {
11         super(nombre, nivelVida, fila, columna, ejercito);
12         this.flechasDisponibles = flechasDisponibles;
13     }
14
15     public int getFlechasDisponibles() {
16         return flechasDisponibles;
17     }
18
19     public void dispararFlecha() {
20         if (flechasDisponibles > 0) {
21             flechasDisponibles--;
22             System.out.println(getNombre() + " ha disparado una flecha. Flechas restantes: " + flechasDisponibles);
23         } else {
24             System.out.println(getNombre() + " no tiene flechas disponibles para disparar.");
25         }
26     }
27
28     @Override
29     public void atacar() {
30         if (flechasDisponibles > 0)
31             dispararFlecha();
32         else
33             System.out.println(getNombre() + " no puede atacar porque no tiene flechas.");
34     }
35 }

```

Clase Ejército.java:

```
1  import java.util.*;
2
3  public class Ejército {
4      private int numEjercito;
5      private ArrayList<Soldado> soldados = new ArrayList<>();
6      private Set<String> posicionesExistentes = new HashSet<>();
7
8      public Ejército(int numEjercito) {
9          this.numEjercito = numEjercito;
10         generarSoldados();
11     }
12
13     public void addSoldado(Soldado soldado) {
14         soldados.add(soldado);
15     }
16
17     public int getTotalSoldados() {
18         return soldados.size();
19     }
20
21     public ArrayList<Soldado> getSoldados() {
22         return soldados;
23     }
24
25     private void generarSoldados() {
26         int n = (int) (Math.random() * Soldado.MAX_SOLDADOS_POR_EJERCITO) + 1;
27         for (int i = 0; i < n; i++) {
28             Soldado soldado = crearSoldadoAleatorio(i + 1);
29             soldados.add(soldado);
30         }
31     }
32
33     private Soldado crearSoldadoAleatorio(int idx) {
34         String tipo = generarTipoSoldado();
35         String nombre = tipo + numEjercito + "X" + idx;
36
37         int fila, columna;
38         do {
39             fila = (int) (Math.random() * 10);
40             columna = (int) (Math.random() * 10);
41         } while (posicionesExistentes.contains(fila + "," + columna));
42         posicionesExistentes.add(fila + "," + columna);
43
44         switch (tipo) {
45             case "Espadachin":
46                 return new Espadachin(nombre, generarVida(3, 4), fila + 1, (char) ('A' + columna), numEjercito, 1.5);
47             case "Arquero":
48                 return new Arquero(nombre, generarVida(1, 3), fila + 1, (char) ('A' + columna), numEjercito, 10);
49             case "Caballero":
50                 return new Caballero(nombre, generarVida(3, 5), fila + 1, (char) ('A' + columna), numEjercito);
51             default:
52                 throw new IllegalArgumentException("Tipo de soldado desconocido: " + tipo);
53         }
54     }
55
56     private String generarTipoSoldado() {
57         String[] tipos = {"Espadachin", "Arquero", "Caballero"};
58         return tipos[(int) (Math.random() * tipos.length)];
59     }
60
61     private int generarVida(int min, int max) {
62         return (int) (Math.random() * (max - min + 1)) + min;
63     }
64
65     public void mostrarEjercito() {
66         System.out.println("\nEjército #" + numEjercito);
67         for (Soldado soldado : soldados) {
68             System.out.println(" - " + soldado);
69         }
70     }
71
72     public boolean isEmpty() {
73         return soldados.isEmpty();
74     }
75 }
```

```

75
76     public void eliminarSoldado(Soldado soldado) {
77         soldados.remove(soldado);
78         posicionesExistentes.remove(soldado.getFila() + "," + soldado.getColumna());
79     }
80
81     public Boolean contains(Soldado soldado) {
82         return soldados.contains(soldado);
83     }
84 }

```

Clase Mapa.java:

```

1  import java.util.*;
2
3  public class Mapa {
4      private ArrayList<ArrayList<String>> tablero = new ArrayList<>();
5      private Ejercito ej1;
6      private Ejercito ej2;
7
8      public Mapa(Ejercito ej1, Ejercito ej2) {
9          this.ej1 = ej1;
10         this.ej2 = ej2;
11         inicializarTablero();
12         colocarSoldados();
13     }
14
15     public Soldado getSoldado(int fila, char columna) {
16         for (Soldado soldado : ej1.getSoldados())
17             if (soldado.getFila() == fila && soldado.getColumna() == columna)
18                 return soldado;
19
20         for (Soldado soldado : ej2.getSoldados())
21             if (soldado.getFila() == fila && soldado.getColumna() == columna)
22                 return soldado;
23
24         return null;
25     }
26
27     private void inicializarTablero() {
28         for (int i = 0; i < 10; i++) {
29             ArrayList<String> fila = new ArrayList<>();
30             for (int j = 0; j < 10; j++)
31                 fila.add(" ");
32             tablero.add(fila);
33         }
34     }
35
36     public void limpiarTablero() {
37         for (int i = 0; i < 10; i++) {
38             for (int j = 0; j < 10; j++)
39                 tablero.get(i).set(j, " ");
40         }
41     }
42
43     private void colocarSoldados() {
44         colocarSoldadosPorEjercito(ej1, 1);
45         colocarSoldadosPorEjercito(ej2, 2);
46     }
47
48     private void colocarSoldadosPorEjercito(Ejercito ejercito, int numEjercito) {
49         for (Soldado soldado : ejercito.getSoldados()) {
50             int fila = soldado.getFila() - 1;
51             int columna = soldado.getColumna() - 'A';
52
53             String tipoSoldado = obtenerTipoSoldado(soldado);
54             String representacion = (numEjercito == 1 ? "[" : "<") + tipoSoldado + soldado.getNivelVida() + (numEjercito == 1 ? "]" : ">");
55
56             tablero.get(fila).set(columna, representacion);
57         }
58     }
59
60     private String obtenerTipoSoldado(Soldado soldado) {
61         if (soldado instanceof Arquero)
62             return "A";
63         else if (soldado instanceof Espadachín)
64             return "E";
65         else if (soldado instanceof Caballero)
66             return "C";
67         else
68             return "S"; // Aquí sería un soldado por defecto
69     }
70 }

```



```

70
71 public void mostrarMapa() {
72     System.out.println("\n");
73     System.out.println("      CAMPO DE BATALLA      ");
74     System.out.println(" ");
75     System.out.println("\n* Los soldados del Ejército 1 están representados por '[tipo/Vida]'"
76         + "\n* Los del Ejército 2 por '<tipo/Vida>'\n");
77     System.out.println(" ");
78
79     System.out.println("\n      A  B  C  D  E  F  G  H  I  J");
80     System.out.println();
81     System.out.println("      -----");
82
83     for (int i = 0; i < tablero.size(); i++) {
84         System.out.print(" " + (i + 1) + "\t| ");
85         for (String celda : tablero.get(i)) {
86             if (celda == "") {
87                 celda = " ";
88             }
89             System.out.print("'" + celda + " | ");
90         }
91         System.out.println();
92         System.out.println("      -----");
93     }
94     System.out.println("\n");
95 }
96
97 public void actualizarMapa(Ejercito ej1, Ejercito ej2) {
98     LimpiarTablero();
99     this.ej1 = ej1;
100    this.ej2 = ej2;
101    colocarSoldados();
102 }
103 }

```

Clase VideoJuego.java

```

1 // LABORATORIO N° 18
2 // AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
3 // TIEMPO: 169 MINUTOS
4 import java.text.DecimalFormat;
5 import java.util.*;
6
7 public class VideoJuego {
8     public static void main(String[] args) {
9         Ejercito ej1 = new Ejercito(1);
10        Ejercito ej2 = new Ejercito(2);
11        Mapa m = new Mapa(ej1, ej2);
12
13        juego(m, ej1, ej2);
14    }
15
16    public static boolean juego(Mapa m, Ejercito ej1, Ejercito ej2) {
17        Scanner sc = new Scanner(System.in);
18        boolean turnoEj1 = true;
19
20        System.out.println(" ");
21        System.out.println("      BIENVENIDOS A LA BATALLA");
22        System.out.println(" ");
23        System.out.println(" En este juego, dos ejércitos lucharán por la victoria. ¡Que gane el mejor!\n");
24        System.out.println(" Instrucciones:");
25        System.out.println(" 1. Cada jugador controlará un ejército por turnos.");
26        System.out.println(" 2. Podrás mover tus soldados en el tablero y atacar a los enemigos.");
27        System.out.println(" 3. El objetivo es derrotar a todos los soldados enemigos.\n");
28        System.out.println(" ");
29        System.out.println();
30
31        System.out.println("      + " + " " + "\n"
32            + "      + " + "      ¡INICIA LA BATALLA! " + "\n"
33            + "      + " + " " + "\n");
34        m.mostrarMapa();
35        System.out.println();
36
37        while (!(ej1.isEmpty()) && !(ej2.isEmpty())) {
38            System.out.println("\n");
39            System.out.println(turnoEj1 ? "      TURNO DEL EJÉRCITO 1"
40                : "      TURNO DEL EJÉRCITO 2");
41            System.out.println(" ");
42
43            System.out.print("Ingrese la coordenada del soldado a mover (Ej. C5): ");
44            String coord = sc.nextLine().toUpperCase();
45

```

```

46     Soldado soldadoSeleccionado = buscarSoldado(m, coord);
47     if (soldadoSeleccionado == null || (turnoEj1 ? !ej1.contains(soldadoSeleccionado) : !ej2.contains(soldadoSeleccionado))) {
48         System.out.println("[ERROR] Movimiento inválido: No hay soldado en la posición o es del ejército contrario.");
49         System.out.println("Observa el mapa y vuelve a escribir la posición del soldado.");
50         continue;
51     }
52
53     System.out.print("Ingrese dirección de movimiento (W para arriba, S para abajo, A para izquierda, D para derecha): ");
54     char direccion = sc.next().toUpperCase().charAt(0);
55     sc.nextLine();
56
57     boolean movimientoExitoso = moverSoldado(soldadoSeleccionado, direccion, m, turnoEj1 ? ej1 : ej2, turnoEj1 ? ej2 : ej1);
58     if (!movimientoExitoso) {
59         System.out.println("[ERROR] Movimiento no válido, intente nuevamente.");
60         continue;
61     }
62
63     m.actualizarMapa(ej1, ej2);
64     m.mostrarMapa();
65     turnoEj1 = !turnoEj1;
66
67     try {
68         Thread.sleep(1000); // Pausa entre turnos
69     } catch (InterruptedException e) {
70         Thread.currentThread().interrupt();
71     }
72 }
73
74 if (ej2.isEmpty()) {
75     System.out.println("=====");
76     System.out.println("                        ¡EJÉRCITO 1 GANA!");
77     System.out.println("=====");
78     System.out.println(" El Ejército 1 ha eliminado a todos los enemigos. Enhorabuena!");
79 } else {
80     System.out.println("=====");
81     System.out.println("                        ¡EJÉRCITO 2 GANA!");
82     System.out.println("=====");
83     System.out.println(" El Ejército 2 ha derrotado al ejército enemigo. ¡Honor a los valientes!");
84 }
85 System.out.println("=====");
86
87 return false;
88 }
89
90 public static Soldado buscarSoldado(Mapa m, String coord) {
91     int columna = coord.charAt(0) - 'A';
92     int fila;
93
94     if (coord.length() == 3)
95         fila = Integer.parseInt(coord.substring(1, 3)) - 1;
96     else
97         fila = Character.getNumericValue(coord.charAt(1)) - 1;
98
99     if (fila >= 0 && fila < 10 && columna >= 0 && columna < 10)
100         return m.getSoldado(fila + 1, (char) ('A' + columna));
101
102     return null;
103 }
104
105 public static boolean moverSoldado(Soldado s, char direccion, Mapa m, Ejercito ejAliado, Ejercito ejEnemigo) {
106     int fila = s.getFila() - 1;
107     int columna = s.getColumna() - 'A';
108
109     switch (direccion) {
110         case 'W': fila--; break;
111         case 'S': fila++; break;
112         case 'A': columna--; break;
113         case 'D': columna++; break;
114         default: return false;
115     }
116
117     if (!verificarMovimientoValido(fila, columna))
118         return false;
119
120     if (ejAliado.contains(m.getSoldado(fila + 1, (char) ('A' + columna))))
121         return false;
122
123     Soldado sEnemigo = m.getSoldado(fila + 1, (char) ('A' + columna));
124
125     if (sEnemigo != null && ejEnemigo.contains(sEnemigo))
126         batalla(s, sEnemigo, ejAliado, ejEnemigo, m);
127     else if (sEnemigo == null) {
128         s.setFila(fila + 1);
129         s.setColumna((char) ('A' + columna));
130     } else
131         return false;
132
133     return true;
134 }

```

```

135
136 public static boolean verificarMovimientoValido(int fila, int columna) {
137     return fila >= 0 && fila < 10 && columna >= 0 && columna < 10;
138 }
139
140 public static void batalla(Soldado s, Soldado sEnemigo, Ejercito ejAliado, Ejercito ejEnemigo, Mapa m) {
141     int vidaTotal = s.getNivelVida() + sEnemigo.getNivelVida();
142     double probS = (double) s.getNivelVida() * 100 / vidaTotal;
143     double probsEnemigo = (double) sEnemigo.getNivelVida() * 100 / vidaTotal;
144
145     DecimalFormat df = new DecimalFormat("#.##");
146
147     Random r = new Random();
148     double resultado = r.nextDouble() * 100;
149
150     System.out.println("=====");
151     System.out.println("                        ¡ENFRENTAMIENTO INICIADO!    ");
152     System.out.println("=====");
153     System.out.println(" Soldado 1: " + s.getNombre() + " [Vida: " + s.getNivelVida() + "]");
154     System.out.println(" Soldado 2: " + sEnemigo.getNombre() + " [Vida: " + sEnemigo.getNivelVida() + "]");
155     System.out.println(" Probabilidades de victoria:");
156     System.out.println("   - " + s.getNombre() + ": " + df.format(probS) + "%");
157     System.out.println("   - " + sEnemigo.getNombre() + ": " + df.format(probsEnemigo) + "%");
158     System.out.println("=====");
159     System.out.println("                        Resolviendo la batalla...    ");
160     System.out.println("=====");
161
162     try {
163         Thread.sleep(1000);
164         System.out.println("Pam! Clash! Boom!\n");
165         Thread.sleep(1000);
166     } catch (InterruptedException e) {
167         Thread.currentThread().interrupt();
168     }
169
170     if (resultado < probS) {
171         System.out.println(s.getNombre() + " ha ganado la batalla contra " + sEnemigo.getNombre() + " y ha sumado 1 punto de vida!");
172         ejEnemigo.eliminarSoldado(sEnemigo);
173         s.setNivelVida(s.getNivelVida() + 1);
174
175         s.setFila(sEnemigo.getFila());
176         s.setColumna(sEnemigo.getColumna());
177
178     } else {
179         System.out.println(sEnemigo.getNombre() + " ha ganado la batalla contra " + s.getNombre() + " y ha sumado 1 punto de vida!");
180         ejAliado.eliminarSoldado(s);
181         sEnemigo.setNivelVida(sEnemigo.getNivelVida() + 1);
182     }
183
184     System.out.println("=====");
185 }
186 }

```

Consola:

```

=====
B I E N V E N I D O S   A   L A   B A T A L L A
=====

En este juego, dos ejércitos lucharán por la victoria. ¡Que gane el mejor!

Instrucciones:
1. Cada jugador controlará un ejército por turnos.
2. Podrás mover tus soldados en el tablero y atacar a los enemigos.
3. El objetivo es derrotar a todos los soldados enemigos.

=====

      [ ¡INICIA LA BATALLA! ]

=====

[ ] [ ] [ ] C A M P O   D E   B A T A L L A [ ] [ ]

=====

* Los soldados del Ejército 1 están representados por '[tipo/Vida]'
* Los del Ejército 2 por '<tipo/Vida>'

=====

```

[illegible]

TURNO DEL EJÉRCITO 1

Ingrese la coordenada del soldado a mover (Ej. C5): D7

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

CAMPO DE BATALLA

* Los soldados del Ejército 1 están representados por '[tipo/Vida]'

```
* Los del Ejército 2 por '<tipo/Vida>'
```

TURNO DEL EJÉRCITO 2

Ingrese la coordenada del soldado a mover (Ej. C5): B9

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

TURNO DEL EJÉRCITO 2

Ingrese la coordenada del soldado a mover (Ej. C5): C9

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

¡ENFRENTAMIENTO INICIADO!

Soldado 1: Arquero2X1 [Vida: 3]

Soldado 2: Espadachín1X1 [Vida: 4]

Probabilidades de victoria:

- Arquero2X1: 42.86%
 - Espadachín1X1: 57.14%
-

Resolviendo la batalla...

Pam! Clash! Boom!

Espadachín1X1 ha ganado la batalla contra Arquero2X1 y ha sumado 1 punto de vida!

C A M P O D E B A T A L L A

* Los soldados del Ejército 1 están representados por '[tipo/Vida]'

* Los del Ejército 2 por '<tipo/Vida>'

	A	B	C	D	E	F	G	H	I	J
1										
2										
3									[C5]	
4		[A2]								
5									[E4]	
6										
7										
8										
9				[E5]						
10										

¡EJÉRCITO 1 GANA!

El Ejército 1 ha eliminado a todos los enemigos. Enhorabuena!

Diagrama de clases UML (Con la extensión PlantUML de VS Code):

