

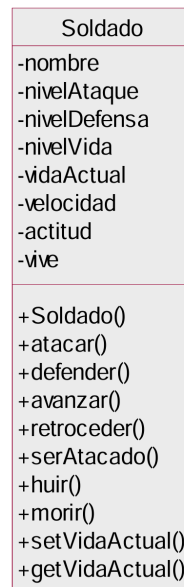
Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Práctica de Laboratorio N° 9:
Definición de Clases de Usuario
Clase Soldado

Nombre: Jhonatan Benjamin Mamani Céspedes

CUI: 20232188

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

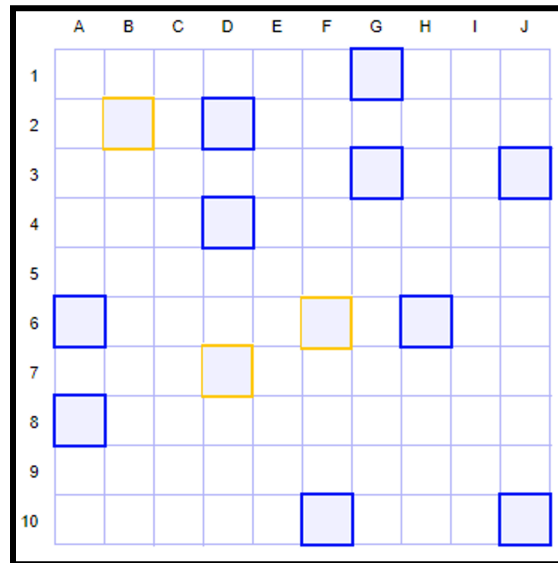
Usar como base el diagrama de clases UML siguiente:



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. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo un programa iterativo.

6. Crear el diagrama de clases UML completo



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; // Aqui 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 = 0;
30         this.nivelDefensa = 0;
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 int getPuntosVida() {
45         return puntosVida;
46     }
47     public int getNivelAtaque() {
48         return nivelAtaque;
49     }
50     public int getNivelDefensa() {
51         return nivelDefensa;
52     }
53     public int getVelocidad() {
54         return velocidad;
55     }
56     public String getActitud() {
57         return actitud;
58     }
59     public boolean isVivo() {
60         return vive;
61     }
62
63     // Métodos del UML de referencia
64     public void atacar() {
65         velocidad += 1;
66         actitud = "Ofensiva";
67         System.out.println(nombre + " ha atacado, su velocidad es ahora " + velocidad);
68     }
69     public void defender() {
70         actitud = "Defensiva";
71         System.out.println(nombre + " está en modo defensivo.");
72     }
73     public void huir() {
74         velocidad += 2;
75         actitud = "Fuga";
76         System.out.println(nombre + " está huyendo, su velocidad es ahora " + velocidad);
77     }
78     public void avanzar() {
79         velocidad += 1;
80         System.out.println(nombre + " avanza, su velocidad es ahora " + velocidad);
81     }

```

```

82     public void retroceder() {
83         if (velocidad > 0) {
84             velocidad = 0;
85             actitud = "Defensiva";
86             System.out.println(nombre + " se ha detenido, velocidad actual: " + velocidad);
87         } else {
88             velocidad -= 1;
89             System.out.println(nombre + " ha retrocedido, velocidad negativa: " + velocidad);
90         }
91     }
92     public void serAtacado(int daño) {
93         recibirAtaque(daño);
94     }
95     public void recibirAtaque(int daño) {
96         puntosVida -= daño;
97         if (puntosVida <= 0) {
98             puntosVida = 0;
99             System.out.println(nombre + " ha muerto.");
100         } else {
101             System.out.println(nombre + " ha recibido " + daño + " de daño. Vida restante: " + puntosVida);
102         }
103     }
104     public void morir() {
105         vive = false;
106         System.out.println(nombre + " ha muerto.");
107     }

```

```

108
109     @Override
110     public String toString() {
111         return "Nombre: " + nombre
112             + " | Vida: " + puntosVida
113             + " | Ataque: " + nivelAtaque
114             + " | Defensa: " + nivelDefensa
115             + " | Velocidad: " + velocidad
116             + " | Actitud: " + actitud
117             + " | Posición: " + fila + columna
118             + " | Vive: " + (vive ? "Si" : "No");
119     }
120 }

```

Clase Videojuego.java:

```

1  // LABORATORIO N° 9
2  // AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
3  // TIEMPO: 79 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: " + promedioVidaE1);
31          System.out.println("Promedio de puntos de vida del ejército 2: " + promedioVidaE2);
32
33          ordenamientoInsertionSort(e1);
34          System.out.println("\nRanking de soldados del ejército 1 (ordenado por puntos de vida de forma decreciente):");
35          mostrarDatosEjercito(e1);
36
37
38          ordenamientoBubbleSort(e2);
39          System.out.println("\nRanking de soldados del ejército 2 (ordenado por puntos de vida de forma decreciente):");
40          mostrarDatosEjercito(e2);
41
42
43          String resultadoBatalla = determinarGanador(e1, e2);
44          System.out.println("\nResultado de la batalla:");
45          System.out.println(resultadoBatalla);
46      }
47
48      public static void inicializarTablero(ArrayList<ArrayList<Soldado>> tablero) {
49          for (int i = 0; i < 10; i++) {
50              ArrayList<Soldado> fila = new ArrayList<>();
51              for (int j = 0; j < 10; j++)
52                  fila.add(null);
53              tablero.add(fila);
54          }
55      }

```

```

57 public static void inicializarEjercitos(ArrayList<Soldado> e1, ArrayList<Soldado> e2,
58     ArrayList<ArrayList<Soldado>> tablero) {
59     for (int i = 0; i < 2; i++) {
60         int n = (int) (Math.random() * 10) + 1;
61         for (int j = 0; j < n; j++) {
62             int fila, columna;
63             do {
64                 fila = (int) (Math.random() * 10);
65                 columna = (int) (Math.random() * 10);
66             } while (tablero.get(fila).get(columna) != null);
67
68             String nombre = "Soldado" + (i + 1) + "X" + (j + 1);
69             int puntosVida = (int) (Math.random() * 5) + 1;
70             Soldado soldado = new Soldado(nombre, puntosVida, fila, (char) ('A' + columna));
71
72             if (i == 0) {
73                 e1.add(soldado);
74                 tablero.get(fila).set(columna, soldado);
75             } else {
76                 e2.add(soldado);
77                 tablero.get(fila).set(columna, soldado);
78             }
79         }
80     }
81 }
82
83 public static void mostrarTablero(ArrayList<ArrayList<Soldado>> tablero, ArrayList<Soldado> e1,
84     ArrayList<Soldado> e2) {
85     System.out.println("Tablero de la batalla:");
86     + "\nLas unidades del ejército 1 estarán con sus puntos de vida entre corchetes ([x])."
87     + "\nLas del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>:).";
88     System.out.println("\n      A    B    C    D    E    F    G    H    I    J");
89     System.out.println();
90     System.out.println("-----");
91
92     for (int i = 0; i < tablero.size(); i++) {
93         System.out.print(i + 1 + "\t| ");
94         for (int j = 0; j < tablero.get(i).size(); j++) {
95             Soldado soldado = tablero.get(i).get(j);
96             if (soldado == null)
97                 System.out.print("    | ");
98             else
99                 if (e1.contains(soldado))
100                     System.out.print("[ " + soldado.getPuntosVida() + " ] | ");
101                 else if (e2.contains(soldado))
102                     System.out.print("< " + soldado.getPuntosVida() + " > | ");
103                 else
104                     System.out.print(soldado.getPuntosVida() + " | ");

```

```

105         }
106         System.out.println();
107         System.out.println("-----");
108     }
109 }
110
111 public static void mostrarDatosEjercito(ArrayList<Soldado> ejercito) {
112     for (Soldado s : ejercito)
113         System.out.println(s);
114 }
115
116 public static Soldado obtenerSoldadoMayorVida(ArrayList<Soldado> ejercito) {
117     Soldado mayorVida = null;
118     int maxPuntosVida = Integer.MIN_VALUE;
119     for (Soldado soldado : ejercito) {
120         if (soldado.getPuntosVida() > maxPuntosVida) {
121             maxPuntosVida = soldado.getPuntosVida();
122             mayorVida = soldado;
123         }
124     }
125     return mayorVida;
126 }
127
128 public static double calcularPromedioVida(ArrayList<Soldado> ejercito) {
129     double total = 0;
130     for (Soldado soldado : ejercito)
131         total += soldado.getPuntosVida();
132     return total / ejercito.size();
133 }
134

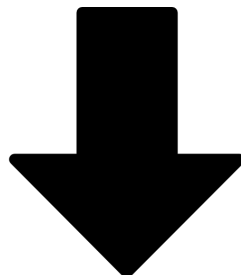
```

```

135     public static void ordenamientoInsertionSort(ArrayList<Soldado> ejercito) {
136         int n = ejercito.size();
137         for (int i = 1; i < n; ++i) {
138             Soldado key = ejercito.get(i);
139             int j = i - 1;
140
141             while (j >= 0 && ejercito.get(j).getPuntosVida() < key.getPuntosVida()) {
142                 ejercito.set(j + 1, ejercito.get(j));
143                 j = j - 1;
144             }
145             ejercito.set(j + 1, key);
146         }
147     }
148     public static void ordenamientoBubbleSort(ArrayList<Soldado> ejercito) {
149         int n = ejercito.size();
150         for (int i = 0; i < n - 1; i++)
151             for (int j = 0; j < n - i - 1; j++)
152                 if (ejercito.get(j).getPuntosVida() < ejercito.get(j + 1).getPuntosVida()){
153                     Soldado temp = ejercito.get(j);
154                     ejercito.set(j, ejercito.get(j + 1));
155                     ejercito.set(j + 1, temp);
156                 }
157     }
158
159     public static String determinarGanador(ArrayList<Soldado> e1, ArrayList<Soldado> e2) {
160         int puntosE1 = 0;
161         int puntosE2 = 0;
162
163         for (Soldado soldado : e1)
164             puntosE1 += soldado.getPuntosVida();
165
166         for (Soldado soldado : e2)
167             puntosE2 += soldado.getPuntosVida();
168
169         if (puntosE1 > puntosE2)
170             return "El ejército 1 ha ganado la batalla. La suma total de sus puntos de vida es " + puntosE1
171                 + " superando en " + (puntosE1 - puntosE2) + " puntos al ejército 2.";
172         else if (puntosE2 > puntosE1)
173             return "El ejército 2 ha ganado la batalla. La suma total de sus puntos de vida es " + puntosE2
174                 + " superando en " + (puntosE2 - puntosE1) + " puntos al ejército 1.";
175         else
176             return "La batalla ha terminado en empate. Ambos ejércitos tienen " + puntosE1 + " puntos de vida en total.";
177     }
178 }

```

Los cambios más importantes hechos en este laboratorio son principalmente adaptar los ejércitos a su forma de ArrayList como en el laboratorio 7 para volver a tener los ordenamientos más eficientes, por otro lado, agregué los nuevos atributos y métodos a la clase Soldado adaptando sus valores por defecto desde los constructores y su posterior muestra desde el toString que se usa también en el mismo videojuego, así se ve a continuación desde la consola:



Consola:

```
● PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamen
ersity\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos de la Programación 2 - A\Laboratorio\P
-cp' 'C:\Users\jhona\AppData\Roaming\Code\User\workspaceStorage\2b402101b08498c28ad9cea985b5fbdb\redhat.java\jdt_ws\Prá
Tablero de la batalla:
Las unidades del ejército 1 estarán 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  |   |   |   |   |   |   |   |   |   |
3  |   | [3] |   |   |   |   |   |   |   |
4  |   |   |   |   |   |   | <5> |   | [2] |   |
5  |   |   |   | [5] |   |   | <3> |   |   |   |
6  |   |   |   |   |   |   |   | <4> |   |   |   |
7  |   | <1> |   |   |   |   | [5] |   |   |   |
8  |   |   |   |   |   | [3] |   |   | [5] | [5] |
9  |   |   |   |   |   |   |   |   |   | <3> |
10 |   |   | [1] |   |   |   |   |   | [2] |   |
-----
```

Datos del ejército 1:

Nombre: Soldado1X1	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 2B	Vive: Si
Nombre: Soldado1X2	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 4D	Vive: Si
Nombre: Soldado1X3	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7I	Vive: Si
Nombre: Soldado1X4	Vida: 2	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 9I	Vive: Si
Nombre: Soldado1X5	Vida: 1	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 9C	Vive: Si
Nombre: Soldado1X6	Vida: 2	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 3I	Vive: Si
Nombre: Soldado1X7	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7F	Vive: Si
Nombre: Soldado1X8	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 6G	Vive: Si
Nombre: Soldado1X9	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7J	Vive: Si

Datos del ejército 2:

Nombre: Soldado2X1	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 3G	Vive: Si
Nombre: Soldado2X2	Vida: 1	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 6B	Vive: Si
Nombre: Soldado2X3	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 4G	Vive: Si
Nombre: Soldado2X4	Vida: 4	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 5H	Vive: Si
Nombre: Soldado2X5	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 8J	Vive: Si

Soldado con mayor vida del ejército 1:

Nombre: Soldado1X2	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 4D	Vive: Si
--------------------	---------	-----------	------------	--------------	--------------------	--------------	----------

Soldado con mayor vida del ejército 2:

Nombre: Soldado2X1	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 3G	Vive: Si
--------------------	---------	-----------	------------	--------------	--------------------	--------------	----------

Promedio de puntos de vida del ejército 1: 3.4444444444444446

Promedio de puntos de vida del ejército 2: 3.2

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

Nombre: Soldado1X2	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 4D	Vive: Si
Nombre: Soldado1X3	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7I	Vive: Si
Nombre: Soldado1X8	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 6G	Vive: Si
Nombre: Soldado1X9	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7J	Vive: Si
Nombre: Soldado1X1	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 2B	Vive: Si
Nombre: Soldado1X7	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 7F	Vive: Si
Nombre: Soldado1X4	Vida: 2	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 9I	Vive: Si
Nombre: Soldado1X6	Vida: 2	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 3I	Vive: Si
Nombre: Soldado1X5	Vida: 1	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 9C	Vive: Si

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

Nombre: Soldado2X1	Vida: 5	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 3G	Vive: Si
Nombre: Soldado2X4	Vida: 4	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 5H	Vive: Si
Nombre: Soldado2X3	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 4G	Vive: Si
Nombre: Soldado2X5	Vida: 3	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 8J	Vive: Si
Nombre: Soldado2X2	Vida: 1	Ataque: 0	Defensa: 0	Velocidad: 0	Actitud: Defensiva	Posición: 6B	Vive: Si

Resultado de la batalla:

El ejército 1 ha ganado la batalla. La suma total de sus puntos de vida es 31 superando en 15 puntos al ejército 2.

Diagrama de clases UML (hecho con Plant UML en VS Code):

