

**Universidad Nacional de San Agustín**  
**Escuela Profesional de Ingeniería de Sistemas**  
**Fundamentos de Programación II**  
**Práctica de Laboratorio N° 8:**  
**HashMap**

**Nombre:** Jhonatan Benjamin Mamani Céspedes

**CUI:** 20232188

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

**Ejercicio 1:**

1. Cree un Proyecto llamado Laboratorio8
2. Usted deberá crear las dos clases Soldado.java y VideoJuego5.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
3. Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
4. El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.
5. Tendrá 2 Ejércitos (usar HashMaps). 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). Se debe mostrar el tablero con todos los soldados creados (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 (indicar conclusiones respecto a este ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como 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
7      public Soldado(String nombre, int puntosVida, int fila, char columna) {
8          this.nombre = nombre;
9          this.puntosVida = puntosVida;
10         this.fila = fila;
11         this.columna = columna;
12     }
13
14     public void setNombre(String nombre) {
15         this.nombre = nombre;
16     }
17     public void setPuntosVida(int puntosVida) {
18         this.puntosVida = puntosVida;
19     }
20     public void setFila(int fila) {
21         this.fila = fila;
22     }
23     public void setColumna(char columna) {
24         this.columna = columna;
25     }
26     public String getNombre() {
27         return nombre;
28     }
29     public int getPuntosVida() {
30         return puntosVida;
31     }
32     public int getFila() {
33         return fila;
34     }
35     public char getColumna() {
36         return columna;
37     }
38
39     @Override
40     public String toString() {
41         return "Nombre: " + nombre
42             + " | Puntos de Vida: " + puntosVida
43             + " | Ubicación: " + fila + columna;
44     }
45 }
```

## Clase Videojuego5.java:

```
1 // LABORATORIO Nº 8 - EJERCICIO 1
2 // AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
3 // TIEMPO: 70 MINUTOS
4 import java.util.*;
5 public class VideoJuego5 {
6     public static void main(String[] args) {
7         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<>();
8         HashMap<String, Soldado> e1 = new HashMap<>();
9         HashMap<String, Soldado> e2 = new HashMap<>();
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     }
56
57     public static void inicializarEjercitos(HashMap<String, Soldado> e1, HashMap<String, 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 + 1, (char) ('A' + columna));
71
72                 if (i == 0)
73                     e1.put(nombre, soldado);
```

```

74         else
75             e2.put(nombre, soldado);
76         tablero.get(fila).set(columna, soldado);
77     }
78 }
79 }
80
81 public static void mostrarTablero(ArrayList<ArrayList<Soldado>> tablero, HashMap<String, Soldado> e1,
82                                 HashMap<String, Soldado> e2) {
83     System.out.println("Tablero de la batalla:");
84     + "\nLas unidades del ejército 1 estarán con sus puntos de vida entre corchetes ([x])."
85     + "\nLas del ejército 2 con sus puntos de vida entre signos menor y mayor que (<x>):");
86     System.out.println("\n          A    B    C    D    E    F    G    H    I    J");
87     System.out.println();
88     System.out.println("-----");
89
90     for (int i = 0; i < tablero.size(); i++) {
91         System.out.print(i + 1 + "\t| ");
92         for (int j = 0; j < tablero.get(i).size(); j++) {
93             Soldado soldado = tablero.get(i).get(j);
94             if (soldado == null) {
95                 System.out.print("    | ");
96             } else {
97                 if (e1.containsKey(soldado.getNombre())) {
98                     System.out.print("[ " + soldado.getPuntosVida() + " ] | ");
99                 } else if (e2.containsKey(soldado.getNombre())) {
100                     System.out.print("< " + soldado.getPuntosVida() + " > | ");
101                 }
102             }
103         }
104         System.out.println("\n-----");
105     }
106 }
107
108 public static void mostrarDatosEjercito(HashMap<String, Soldado> ejercito) {
109     for (Soldado s : ejercito.values())
110         System.out.println(s);
111 }
112
113 public static Soldado obtenerSoldadoMayorVida(HashMap<String, Soldado> ejercito) {
114     Soldado mayorVida = null;
115     int maxPuntosVida = Integer.MIN_VALUE;
116     for (Soldado soldado : ejercito.values())
117         if (soldado.getPuntosVida() > maxPuntosVida) {
118             maxPuntosVida = soldado.getPuntosVida();
119             mayorVida = soldado;
120         }
121     return mayorVida;
122 }

```

```

123
124 public static double calcularPromedioVida(HashMap<String, Soldado> ejercito) {
125     double total = 0;
126     for (Soldado soldado : ejercito.values())
127         total += soldado.getPuntosVida();
128     return total / ejercito.size();
129 }
130
131 public static void ordenamientoInsertionSort(HashMap<String, Soldado> ejercito) {
132     List<String> keys = new ArrayList<>(ejercito.keySet());
133     for (int i = 1; i < keys.size(); i++) {
134         String key = keys.get(i);
135         Soldado soldadoActual = ejercito.get(key);
136         int j = i - 1;
137
138         while (j >= 0 && ejercito.get(keys.get(j)).getPuntosVida() < soldadoActual.getPuntosVida()) {
139             ejercito.put(keys.get(j + 1), ejercito.get(keys.get(j)));
140             j--;
141         }
142         ejercito.put(keys.get(j + 1), soldadoActual);
143     }
144 }
145
146 public static void ordenamientoBubbleSort(HashMap<String, Soldado> ejercito) {
147     List<String> keys = new ArrayList<>(ejercito.keySet());
148     boolean swapped;
149     int n = keys.size();
150     for (int i = 0; i < n - 1; i++) {
151         swapped = false;
152         for (int j = 0; j < n - i - 1; j++)
153             if (ejercito.get(keys.get(j)).getPuntosVida() < ejercito.get(keys.get(j + 1)).getPuntosVida()) {

```

```

154         Soldado temp = ejercito.get(keys.get(j));
155         ejercito.put(keys.get(j), ejercito.get(keys.get(j + 1)));
156         ejercito.put(keys.get(j + 1), temp);
157         swapped = true;
158     }
159     if (!swapped) break;
160 }
161 }
162
163 public static String determinarGanador(HashMap<String, Soldado> e1, HashMap<String, Soldado> e2) {
164     int puntosE1 = 0;
165     for (Soldado soldado : e1.values())
166         puntosE1 += soldado.getPuntosVida();
167
168     int puntosE2 = 0;
169     for (Soldado soldado : e2.values())
170         puntosE2 += soldado.getPuntosVida();
171
172     if (puntosE1 > puntosE2)
173         return "El ejército 1 ha ganado la batalla. La suma total de sus puntos de vida es " + puntosE1
174             + " superando en " + (puntosE1 - puntosE2) + " puntos al ejército 2.";
175     else if (puntosE2 > puntosE1)
176         return "El ejército 2 ha ganado la batalla. La suma total de sus puntos de vida es " + puntosE2
177             + " superando en " + (puntosE2 - puntosE1) + " puntos al ejército 1.";
178     else
179         return "La batalla ha terminado en empate. Ambos ejércitos tienen " + puntosE1 + " puntos de vida en total.";
180 }
181 }

```

## Consola:

```

PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos\University\Universidad Nacional de San Agustín\2nd Year\Segundo Semestre\Fundamentos de la Programación 2 - lsInExceptionMessages' '-cp' 'C:\Users\jhona\AppData\Roaming\Code\User\workspaceStorage\c0ff9a4fb5c917b98d6bfc0346fa
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  |   |   |   |   |   | <4> |   | [2] |   |
-----
3  | [2] |   | <2> | [2] |   |   |   |   |   |
-----
4  | [3] |   |   |   |   |   |   |   | <4> |   |
-----
5  |   |   |   |   |   |   |   |   |   |   |
-----
6  |   |   |   |   |   |   |   |   |   |   |
-----
7  |   |   |   |   |   |   |   |   |   |   |
-----
8  |   |   |   |   |   |   |   |   | [5] | <2> |
-----
9  |   |   |   |   |   |   |   |   | [5] |   |
-----
10 | [5] |   |   |   |   | <4> |   | <2> |   |   |
-----

Datos del ejército 1:
Nombre: Soldado1X3 | Puntos de Vida: 2 | Ubicación: 2H
Nombre: Soldado1X4 | Puntos de Vida: 3 | Ubicación: 4A
Nombre: Soldado1X5 | Puntos de Vida: 5 | Ubicación: 10A
Nombre: Soldado1X6 | Puntos de Vida: 2 | Ubicación: 3D
Nombre: Soldado1X1 | Puntos de Vida: 5 | Ubicación: 9H
Nombre: Soldado1X2 | Puntos de Vida: 2 | Ubicación: 3A
Nombre: Soldado1X7 | Puntos de Vida: 5 | Ubicación: 8I

```

```

Datos del ejército 2:
Nombre: Soldado2X2 | Puntos de Vida: 2 | Ubicación: 10H
Nombre: Soldado2X3 | Puntos de Vida: 2 | Ubicación: 3C
Nombre: Soldado2X4 | Puntos de Vida: 4 | Ubicación: 2F
Nombre: Soldado2X5 | Puntos de Vida: 4 | Ubicación: 4I
Nombre: Soldado2X1 | Puntos de Vida: 2 | Ubicación: 8J
Nombre: Soldado2X6 | Puntos de Vida: 4 | Ubicación: 10F

Soldado con mayor vida del ejército 1:
Nombre: Soldado1X5 | Puntos de Vida: 5 | Ubicación: 10A

Soldado con mayor vida del ejército 2:
Nombre: Soldado2X4 | Puntos de Vida: 4 | Ubicación: 2F

Promedio de puntos de vida del ejército 1: 3.4285714285714284
Promedio de puntos de vida del ejército 2: 3.0

Ranking de soldados del ejército 1 (ordenado por puntos de vida de forma decreciente):
Nombre: Soldado1X5 | Puntos de Vida: 5 | Ubicación: 10A
Nombre: Soldado1X1 | Puntos de Vida: 5 | Ubicación: 9H
Nombre: Soldado1X7 | Puntos de Vida: 5 | Ubicación: 8I
Nombre: Soldado1X4 | Puntos de Vida: 3 | Ubicación: 4A
Nombre: Soldado1X3 | Puntos de Vida: 2 | Ubicación: 2H
Nombre: Soldado1X6 | Puntos de Vida: 2 | Ubicación: 3D
Nombre: Soldado1X2 | Puntos de Vida: 2 | Ubicación: 3A

Ranking de soldados del ejército 2 (ordenado por puntos de vida de forma decreciente):
Nombre: Soldado2X4 | Puntos de Vida: 4 | Ubicación: 2F
Nombre: Soldado2X5 | Puntos de Vida: 4 | Ubicación: 4I
Nombre: Soldado2X6 | Puntos de Vida: 4 | Ubicación: 10F
Nombre: Soldado2X2 | Puntos de Vida: 2 | Ubicación: 10H
Nombre: Soldado2X3 | Puntos de Vida: 2 | Ubicación: 3C
Nombre: Soldado2X1 | Puntos de Vida: 2 | Ubicación: 8J

Resultado de la batalla:
El ejército 1 ha ganado la batalla. La suma total de sus puntos de vida es 24 superando en 6 puntos al ejército 2.

```

## Conclusiones sobre el ordenamiento con HashMap:

El ordenamiento de elementos aplicado en el HashMap me resulta menos eficiente y algo más complicado, ya que no hay un orden natural garantizado en el HashMap ni tampoco los índices comunes como en las List.

Para ordenar el HashMap, resulta necesario extraer las claves, realizar el ordenamiento haciendo uso de la lista nueva de las mismas y luego reasignar los elementos, lo que me resultó en un proceso muy evitable al usar ArrayList o simplemente alguna de su colección que tenga la característica de índices numéricos ordenados.