


```
1 package Laboratorio5.src;
2 import java.util.*;
3 public class Videojuego2 {
4     public static int vidaTotal = 0;
5     public static Soldado[] soldadosUniDimensional = new Soldado[10]; // PARA NO DESPERDICIR TANTA MEMORIA ME APOYO DE UNA MATRIZ
6                                     // UNIDIMENSIONAL PARA LOS ORDENAMIENTOS
7
8     Run | Debug
9     public static void main(String[] args) {
10         int cantidad = (int) (Math.random() * 10 + 1);
11         Soldado[][] tablero = new Soldado[10][10];
12         mostrarTabla(tablero); // MUESTRA LA TABLA VACÍA
13         System.out.println();
14         inicializarEjercito(cantidad, tablero); // GENERA A LOS SOLDADOS
15         mostrarTabla(tablero); // MUESTRA LA TABLA CON NOMBRES DE SOLDADOS EN SUS POSICIONES
16         System.out.println("El soldado con mayor cantidad de vida es el " + mostrarSoldadoMayorVida(tablero).getNombre() + " con " +
17             mostrarSoldadoMayorVida(tablero).getVida() + " puntos de vida!!");
18         System.out.println("El promedio del nivel de vida de los soldados es: " + promedioVida(tablero, cantidad));
19         System.out.println("El nivel de vida de todo el ejercito es: " + vidaTotal);
20         System.out.println(x:"Soldado en el orden que fueron creados");
21         imprimirInformacion(cantidad); // IMPRIME CON EL toString LA INFORMACIÓN DE LOS SOLDADOS DESORDENADOS
22         ordenarPorPuntosSeleccion(cantidad); //ORDENA LA MATRIZ GLOBAL (BURBUJA)
23         rankingDePoder(cantidad); // TAMBIÉN ORDENA LA MATRIZ GLOBAL (SELECCIÓN)
24         System.out.println(x:"Soldado por ranking de poder: ");
25         imprimirInformacion(cantidad); // IMPRIME CON EL toString LA INFORMACIÓN ORDENADA DESCENDENTEMENTE
26     }
27
28     public static void inicializarEjercito(int cantidad, Soldado[][] tablero) { // INICIALIZA SOLDADOS CON SU INFORMACIÓN
29         int contadorNombreSoldado = 0;
30         while (cantidad > 0) {
31             int fila = (int) (Math.random() * 10);
32             int columna = (int) (Math.random() * 10);
33             if (tablero[fila][columna] == null) { // SE VERIFICA QUE NO HAYA OTRO SOLDADO EN ESA POSICIÓN, SI LO HAY, BUSCA OTRA
34                 tablero[fila][columna] = new Soldado(); // CREA UNA NUEVA DIRECCIÓN DEL OBJETO SOLDADO
35                 tablero[fila][columna].setFila(fila);
36                 tablero[fila][columna].setColumna(columna);
37                 tablero[fila][columna].setNombre("Soldado" + contadorNombreSoldado);
38                 tablero[fila][columna].setVida((int) (Math.random() * 5 + 1));
39                 soldadosUniDimensional[contadorNombreSoldado] = tablero[fila][columna];
40                 contadorNombreSoldado++;
41                 cantidad--;
42                 vidaTotal += tablero[fila][columna].getVida();
43             }
44         }
45     }
46 }
```

```
41     }
42 }
43 public static void mostrarTabla(Soldado[][] tablero) { // GENERA A BASE DE | Y _ UN TABLA PARA LOS SOLDADOS
44     for (int i = 0; i < tablero.length; i++) {
45         for (int j = 0; j < tablero[i].length; j++) {
46             if (tablero[i][j] == null) {
47                 System.out.print(s:"|_____"); // GENERA ESPACIOS VACÍOS SI NO HAY UN OBJETO EN ESA DIRECCIÓN
48             } else {
49                 System.out.print("|" + tablero[i][j].getNombre()); // CARGA EL NOMBRE DEL SOLDADO SI EXISTE
50             }
51         }
52         System.out.println(x:"|"); //ACOMODA EL TABLERO 10X10
53     }
54 }
55
56 public static Soldado mostrarSoldadoMayorVida(Soldado[][] tablero) { // MUESTRA AL SOLDADO CON MAYOR VIDA
57     Soldado mayorVida = null; // TRABAJA EN LA MATRIZ BIDIMENSIONAL
58     for (int i = 0; i < tablero.length; i++) {
59         for (int j = 0; j < tablero[i].length; j++) {
60             if (tablero[i][j] != null) {
61                 if (mayorVida == null || mayorVida.getVida() < tablero[i][j].getVida()) {
62                     mayorVida = tablero[i][j];
63                 }
64             }
65         }
66     }
67     return mayorVida;
68 }
69
70 public static double promedioVida(Soldado[][] tablero, int cantidad) { //MUESTRA EL PROMEDIO DE VIDA TRBAJA EN LA
71     double promedioVida = 0; // MATRIZ BIDIMENSIONAL
72     for (int i = 0; i < tablero.length; i++) {
73         for (int j = 0; j < tablero[i].length; j++) {
74             if (tablero[i][j] != null)
75                 promedioVida += tablero[i][j].getVida();
76         }
77     }
78     promedioVida /= cantidad;
79     return promedioVida;
```

```
81 public static Soldado[] rankingDePoder(int cantidad) { //PRIMER ALGORITMO DE ORDENAMIENTO (BURBUJA)
82     boolean intercambio = true;
83     while (intercambio) {
84         intercambio = false; // LO MANTIENE FALSO HASTA QUE SE HAYA UN INTERCAMBIO, SINO SE SALE DEL BUCLE
85         for (int i = 0; i < cantidad - 1; i++)
86             if (soldadosUniDimensional[i].getVida() < soldadosUniDimensional[i + 1].getVida()) {
87                 intercambio = true;
88                 Soldado temp = new Soldado(); //VARIABLE TEMPORAL PARA EL INTERCAMBIO
89                 temp = soldadosUniDimensional[i + 1];
90                 soldadosUniDimensional[i + 1] = soldadosUniDimensional[i];
91                 soldadosUniDimensional[i] = temp;
92             }
93     }
94     return soldadosUniDimensional;
95 }
96
97 public static void ordenarPorPuntosSeleccion(int cantidad) { // 2DO ALGORITMO DE ORDENAMIENTO
98     for (int i = 0; i < cantidad - 1; i++) { //SE USA LA CANTIDAD PARA EVITAR QUE EL BUCLE SEÑALE A OBJETOS NULL
99         int menor = i;
100         for (int j = i + 1; j < cantidad; j++) {
101             if (soldadosUniDimensional[j].getVida() > soldadosUniDimensional[menor].getVida()) {
102                 menor = j; // Almacena la posición del menor
103             }
104         }
105         // Intercambio de elementos
106         Soldado temp = soldadosUniDimensional[menor];
107         soldadosUniDimensional[menor] = soldadosUniDimensional[i];
108         soldadosUniDimensional[i] = temp;
109     }
110 }
111 public static void imprimirInformacion(int cantidad){
112     for(int i = 0; i<cantidad; i++){
113         System.out.println("SOLDADO " + i + ":");
114         System.out.println(soldadosUniDimensional[i].toString()); //ME APOYO DE UNA MATRIZ UNIDIMENSIONAL PARA NO
115     } // DESPERDICIA MEMORIA
116 }
117 }
```

[illegible]

						Soldado3	Soldado0	
		Soldado7						
						Soldado2		
								Soldado9
					Soldado1			
				Soldado8				
				Soldado6		Soldado4	Soldado5	

∴ El soldado con mayor cantidad de vida es el Soldado5 con 5 puntos de vida.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 6

```

jEl soldado con mayor cantidad de vida es el Soldado3 con 5 puntos de vida!!
El promedio del nivel de vida de los soldados es: 2.25
El nivel de vida de todo el ejercito es: 9
Soldado en el orden que fueron creados
SOLDADO 0:
Información:
Nombre: Soldado0
Vida: 2
Fila: 0
Columna: 6
SOLDADO 1:
Información:
Nombre: Soldado1
Vida: 1
Fila: 9
Columna: 9
SOLDADO 2:
Información:
Nombre: Soldado2
Vida: 1
Fila: 2
Columna: 6
SOLDADO 3:
Información:
Nombre: Soldado3
Vida: 5
Fila: 6
Columna: 1

```

Aquí se ejecuta la impresión del
Soldado con mayor vida, promedio y total de vida del ejército. Además de los soldados sin orden.

```

Soldado por ranking de poder:
SOLDADO 0:
Información:
Nombre: Soldado3
Vida: 5
Fila: 6
Columna: 1
SOLDADO 1:
Información:
Nombre: Soldado0
Vida: 2
Fila: 0
Columna: 6
SOLDADO 2:
Información:
Nombre: Soldado2
Vida: 1
Fila: 2
Columna: 6
SOLDADO 3:
Información:
Nombre: Soldado1
Vida: 1
Fila: 9
Columna: 9

```

Aquí se ejecuta el ordenamiento de los
soldados por ambos métodos.

II. PRUEBAS

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 7</p>

¿Con qué valores comprobaste que tu práctica estuviera correcta?

Con valores int, String y boolean, además datos que parecía que el programa aceptaría como caracteres especiales para probar como funciona cada método.

¿Qué resultado esperabas obtener para cada valor de entrada?



Esperaba que se almacenara dentro del objeto y atributo que quería; esperaba no tener errores, pero tuve varios al momento de construir el tablero e idear una forma de hacer los ordenamientos sin usar mucha memoria. Al final solucioné todo.

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Obtuve, al final, el correcto, una secuencia limpia de los métodos usados en el main y sin ningún error.

III. CUESTIONARIO:

PRUEBAS DE COMMIT HECHO EN GIT BASH:

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

```

ASUS@DESKTOP-J2KJOAM MINGW64 ~/PF2/PF2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   LAYME_SALAS_LABORATORIO_05/src/Soldado.java
        modified:   LAYME_SALAS_LABORATORIO_05/src/Videojuego2.java

no changes added to commit (use "git add" and/or "git commit -a")

ASUS@DESKTOP-J2KJOAM MINGW64 ~/PF2/PF2 (main)
$ git add .

ASUS@DESKTOP-J2KJOAM MINGW64 ~/PF2/PF2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   LAYME_SALAS_LABORATORIO_05/src/Soldado.java
        modified:   LAYME_SALAS_LABORATORIO_05/src/Videojuego2.java

ASUS@DESKTOP-J2KJOAM MINGW64 ~/PF2/PF2 (main)
$ git commit -m "Laboratorio05 terminado"
[main 0f37320] Laboratorio05 terminado
 2 files changed, 82 insertions(+), 70 deletions(-)

ASUS@DESKTOP-J2KJOAM MINGW64 ~/PF2/PF2 (main)
$ git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.36 KiB | 2.36 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/F4brici0L4yme/PF2.git
 7b1733e..0f37320  main -> main

```

Hice un git status para ver mis archivos sin trackear, los añadí con git add . (con el . se agregan todos los archivos modificados). Luego el commit con el mensaje de lab terminado y un git push para subir a mi repositorio en la nube. Eso es todo, puede ver mi trabajo por el siguiente link.

LINK A MI REPOSITORIO DE GIT HUB: <https://github.com/F4brici0L4yme/PF2.git>

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

CONCLUSIONES

Los métodos para ordenar los Arrays son bastante útiles, requiere unas líneas más al tratarse de arreglos bidimensionales de objetos, pero siguen funcionando sin problema. Además, bastantes versátiles, pueden ser usados en todo tipo de programa.

METODOLOGÍA DE TRABAJO

Usé las mismas que fui usando durante estos laboratorios, comentar bloques de código para poder concentrarme en una parte y revisando problemas pasados y similares que ya resolví para tener una idea y construir un nuevo método.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez y M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021, pp. 170, ISBN 978-612-5035-20-2.

RÚBRICA DE CALIFICACIÓN DE LABORATORIO

(EN LA SIGUIENTE PÁGINA)

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p align="center">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 11</p>

8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
TOTAL		20		19	