

	<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: 1</p>

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación 2				
TÍTULO DE LA PRÁCTICA:	Combinando Arreglos Estándar y ArrayList				
NÚMERO DE PRÁCTICA:	07	AÑO LECTIVO:	2024 B	NRO. SEMESTRE:	II
FECHA DE PRESENTACIÓN	15/11/2024	HORA DE PRESENTACIÓN	18:20:00		
INTEGRANTE (s) Layme Salas Rodrigo Fabricio				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <p style="text-align: center;"><b>CLASE VideoJuego7</b></p> <p style="text-align: center;">(Las explicaciones están en los comentarios dentro del código y hay comentarios extra debajo de las imágenes)</p>

```

1  /*Propósito: simular un tablero 10x10 en el que se desarrolla una batalla con arreglos estándar y ArrayList*/
2  import java.util.*;
3  public class Videojuego7 {
4      public static int vidaTotalAzul = 0; //PARA HALLAR EL PROMEDIO
5      public static int vidaTotalRojo = 0;
6      public static Soldado mayorVidaAzul = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null); // INICIALIZO UN SOLDADO PARA
7      //COMPARACIÓN
8      public static Soldado mayorVidaRojo = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null);
9      public static Soldado[] soldadosUniDimensionalAzul = new Soldado[10]; // ARREGLO ESTÁNDAR UTILIZADO PARA ORDENAMIENTOS
10     public static Soldado[] soldadosUniDimensionalRojo = new Soldado[10]; // ARREGLO ESTÁNDAR UTILIZADO PARA ORDENAMIENTOS
11     Run | Debug
12     public static void main(String[] args) {
13         Scanner scan = new Scanner(System.in);
14         boolean seguir = true;
15         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<ArrayList<Soldado>>(); //INICIALIZO MI ARRAYLIST BIDIMENSIONAL
16         while(seguir){
17             System.out.println("¿Desea ejecutar el programa? (s/n)");
18             String rpt = scan.next();
19             if(rpt.equals("s")){
20                 iniciarPrograma(tablero);
21                 tablero.removeAll(tablero); // Limpia el tablero para un nuevo juego
22                 vidaTotalAzul = 0;
23                 vidaTotalRojo = 0;
24                 mayorVidaAzul = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null);
25                 mayorVidaRojo = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null);
26                 soldadosUniDimensionalAzul = new Soldado[10];
27                 soldadosUniDimensionalRojo = new Soldado[10];
28             }
29             else if(rpt.equals("n"))
30                 seguir = false;
31             else
32                 System.out.println("Esa no es una opción válida");
33         }
34     }
35 }

```

He reducido líneas de código al eliminar métodos innecesarios que estaban en el laboratorio anterior. Agregué un bucle para que el programa sea iterativo y un reinicio de datos.

```

33     public static void iniciarPrograma(ArrayList<ArrayList<Soldado>> tablero){
34         int cantidad = (int)(Math.random() * 10 + 1);
35         int cantidadEnemiga = (int)(Math.random() * 10 + 1);
36         for(int i = 0; i<10; i++){ //INICIALIZAR 10 FILAS
37             tablero.add(new ArrayList<Soldado>());
38             for(int j = 0; j<10; j++){ //INICIALIZAR 10 COLUMNAS
39                 tablero.get(i).add(e:null); // INICIALIZA CON VALOR NULL PARA AHORRAR MEMORIA
40             }
41         }
42         inicializarEjercito(tablero, cantidad, color:"\u001B[44mSoldado", equipo:"azul"); // INICIALIZAR DATOS DE LOS EJERCITOS
43         inicializarEjercito(tablero, cantidadEnemiga, color:"\u001B[41mSoldado", equipo:"rojo");
44         mostrarTabla(tablero); // MUESTRA TABLA
45         hallarSoldadoMayorVida(tablero);
46         System.out.println("El soldado con mayor vida del ejército azul es: " + mayorVidaAzul);
47         System.out.println("El soldado con mayor vida del ejército rojo es: " + mayorVidaRojo);
48         System.out.println("El promedio del ejército azul es: " + vidaTotalAzul/cantidad);
49         System.out.println("El promedio del ejército rojo es: " + vidaTotalRojo/cantidadEnemiga);
50         System.out.println("DATOS DEL EJÉRCITO AZUL POR ORDEN DE INGRESO:");
51         imprimirInformacion(cantidad, soldadosUniDimensionalAzul); // IMPRIME LA INFORMACIÓN CON toString
52         System.out.println("DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:");
53         imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
54         rankingDePoder(cantidad, soldadosUniDimensionalAzul); // ORDENA POR MÉTODO BURBUJA
55         ordenarSelección(cantidadEnemiga, soldadosUniDimensionalRojo); // ORDENA POR MÉTODO SELECCIÓN
56         System.out.println("DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:");
57         imprimirInformacion(cantidad, soldadosUniDimensionalAzul);
58         System.out.println("DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:");
59         imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
60         mostrarGanador(); // MUESTRA EL GANADOR, CRITERIO: VIDA TOTAL DE LOS EJÉRCITOS
61     }

```

*Ahora el programa se inicia mediante un método llamado si el usuario decide ejecutar el programa.*

```
public static void inicializarEjercito(ArrayList<ArrayList<Soldado>> tablero, int cantidad, String color, String equipo) { // INICIALIZA SOLDADOS CON SU INFORMACIÓN
    int contadorIndiceSoldadoAzul = 0; //CONTADORES DIFERENTES PARA EVITAR ERROR NullPointerException
    int contadorIndiceSoldadoRojo = 0;
    while (cantidad > 0) {
        int fila = (int) (Math.random() * 10);
        int columna = (int) (Math.random() * 10);
        if (tablero.get(fila).get(columna) == null) { // SE VERIFICA QUE NO HAYA OTRO SOLDADO EN ESA POSICIÓN, SI LO HAY, BUSCA OTRA
            tablero.get(fila).set(columna, new Soldado(color+fila+"X"+columna + "\u001B[0m", (int) (Math.random() * 5 + 1), fila, columna, equipo)); //COLORES PARA LA DISTINCIÓN
            cantidad--;
            if(equipo.equals(anObject:"azul")){
                vidaTotalAzul += tablero.get(fila).get(columna).getVida();
                soldadosUniDimensionalAzul[contadorIndiceSoldadoAzul] = tablero.get(fila).get(columna);
                contadorIndiceSoldadoAzul++;
            }
            else{
                vidaTotalRojo += tablero.get(fila).get(columna).getVida();
                soldadosUniDimensionalRojo[contadorIndiceSoldadoRojo] = tablero.get(fila).get(columna);
                contadorIndiceSoldadoRojo++;
            }
        }
    }
}
```

*Ahora ya no uso 2 métodos para inicializar los soldados, solo 1.*

```
61 public static void mostrarTabla(ArrayList<ArrayList<Soldado>> tablero) { // GENERA LA TABLA PARA LOS SOLDADOS
62     for (int i = 0; i < tablero.size(); i++) {
63         for (int j = 0; j < tablero.get(i).size(); j++) {
64             if (tablero.get(i).get(j) == null) System.out.print(s:"|      "); // GENERA ESPACIOS EN BLANCO
65             else System.out.print(s:"|" + tablero.get(i).get(j).getNombre()); // CARGA EL NOMBRE DEL SOLDADO SI EXISTE
66         }
67         System.out.println(s:"|"); // ACOMODA EL TABLERO 10X10
68     }
69 }
70 public static void hallarSoldadoMayorVida(ArrayList<ArrayList<Soldado>> tablero) { // MUESTRA AL SOLDADO CON MAYOR VIDA
71     for (int i = 0; i < tablero.size(); i++) {
72         for (int j = 0; j < tablero.get(i).size(); j++) {
73             if (tablero.get(i).get(j) != null && tablero.get(i).get(j).getEquipo().equals(anObject:"azul") && mayorVidaAzul.getVida() < tablero.get(i).get(j).getVida()) // SOLO
74                 SE CUMPLE SI HAY UN SOLDADO, PERTENECE AL EQUIPO AZUL Y ES MAYOR
75                 mayorVidaAzul = tablero.get(i).get(j);
76             if (tablero.get(i).get(j) != null && tablero.get(i).get(j).getEquipo().equals(anObject:"rojo") && mayorVidaRojo.getVida() < tablero.get(i).get(j).getVida()) // SOLO
77                 SE CUMPLE SI LA VIDA ES DIFERENTES DE 0, PERTENECE AL EQUIPO ROJO Y ES MAYOR
78                 mayorVidaRojo = tablero.get(i).get(j);
79         }
80     }
81 }
```

```

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

```

En los ordenamientos uso los arreglos estándar, en el tablero el ArrayList.

```

111 public static void mostrarGanador() { // CRITERIO: CANTIDAD TOTAL DE VIDA
112     if(vidaTotalAzul > vidaTotalRojo)
113         System.out.println("¡El ejercito azul gana por mayor nivel de vida! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
114     else if(vidaTotalAzul < vidaTotalRojo)
115         System.out.println("¡El ejercito rojo gana por mayor nivel de vida! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
116     else
117         System.out.println("¡Ha ocurrido un empate! " + "\nAzul " + vidaTotalAzul + ":" + vidaTotalRojo + " Rojo");
118 }
119 }

```

## PRUEBAS

```

¿Desea ejecutar el programa? (s/n)
s

```

				Soldado0X4				Soldado0X8	
								Soldado1X8	
	Soldado2X1	Soldado2X2							
									Soldado3X9
Soldado4X0		Soldado4X2							
							Soldado5X7		
				Soldado6X4	Soldado6X5			Soldado7X7	
					Soldado7X5				
					Soldado8X5				
	Soldado9X1				Soldado9X5				

**Primero se pregunta si se quiere iniciar el programa, se muestra la tabla con los soldados posicionados diferenciados por el color ROJO-AZUL.**

El soldado con mayor vida del ejército azul es: Nombre: Soldado1X8 | Vida: 5 | Fila: 1 | Columna: 8  
El soldado con mayor vida del ejército rojo es: Nombre: Soldado8X5 | Vida: 5 | Fila: 8 | Columna: 5  
El promedio del ejercito azul es: 2  
El promedio del ejercito rojo es: 2

*Se muestran los datos de los Soldados y el ejército, como su promedio y el mejor y peor soldado.*

DATOS DEL EJÉRCITO AZUL POR ORDEN DE INGRESO:  
SOLDADO 0:  
Nombre: Soldado6X5 | Vida: 3 | Fila: 6 | Columna: 5  
SOLDADO 1:  
Nombre: Soldado7X7 | Vida: 2 | Fila: 7 | Columna: 7  
SOLDADO 2:  
Nombre: Soldado1X8 | Vida: 5 | Fila: 1 | Columna: 8  
SOLDADO 3:  
Nombre: Soldado2X2 | Vida: 1 | Fila: 2 | Columna: 2  
SOLDADO 4:  
Nombre: Soldado0X4 | Vida: 2 | Fila: 0 | Columna: 4  
SOLDADO 5:  
Nombre: Soldado3X9 | Vida: 4 | Fila: 3 | Columna: 9  
SOLDADO 6:  
Nombre: Soldado5X7 | Vida: 1 | Fila: 5 | Columna: 7  
SOLDADO 7:  
Nombre: Soldado0X8 | Vida: 3 | Fila: 0 | Columna: 8  
SOLDADO 8:  
Nombre: Soldado6X4 | Vida: 1 | Fila: 6 | Columna: 4  
SOLDADO 9:  
Nombre: Soldado4X2 | Vida: 3 | Fila: 4 | Columna: 2  
DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:  
SOLDADO 0:  
Nombre: Soldado4X0 | Vida: 2 | Fila: 4 | Columna: 0  
SOLDADO 1:  
Nombre: Soldado8X5 | Vida: 5 | Fila: 8 | Columna: 5  
SOLDADO 2:  
Nombre: Soldado7X5 | Vida: 1 | Fila: 7 | Columna: 5  
SOLDADO 3:  
Nombre: Soldado2X1 | Vida: 1 | Fila: 2 | Columna: 1  
SOLDADO 4:  
Nombre: Soldado9X1 | Vida: 2 | Fila: 9 | Columna: 1  
SOLDADO 5:  
Nombre: Soldado9X5 | Vida: 4 | Fila: 9 | Columna: 5

*Ahora, por orden de ingreso se imprimen los datos de los soldados.*

```
DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:
SOLDADO 0:
Nombre: Soldado1X8 | Vida: 5 | Fila: 1 | Columna: 8
SOLDADO 1:
Nombre: Soldado3X9 | Vida: 4 | Fila: 3 | Columna: 9
SOLDADO 2:
Nombre: Soldado6X5 | Vida: 3 | Fila: 6 | Columna: 5
SOLDADO 3:
Nombre: Soldado0X8 | Vida: 3 | Fila: 0 | Columna: 8
SOLDADO 4:
Nombre: Soldado4X2 | Vida: 3 | Fila: 4 | Columna: 2
SOLDADO 5:
Nombre: Soldado7X7 | Vida: 2 | Fila: 7 | Columna: 7
SOLDADO 6:
Nombre: Soldado0X4 | Vida: 2 | Fila: 0 | Columna: 4
SOLDADO 7:
Nombre: Soldado2X2 | Vida: 1 | Fila: 2 | Columna: 2
SOLDADO 8:
Nombre: Soldado5X7 | Vida: 1 | Fila: 5 | Columna: 7
SOLDADO 9:
Nombre: Soldado6X4 | Vida: 1 | Fila: 6 | Columna: 4
DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:
SOLDADO 0:
Nombre: Soldado8X5 | Vida: 5 | Fila: 8 | Columna: 5
SOLDADO 1:
Nombre: Soldado9X5 | Vida: 4 | Fila: 9 | Columna: 5
SOLDADO 2:
Nombre: Soldado9X1 | Vida: 2 | Fila: 9 | Columna: 1
SOLDADO 3:
Nombre: Soldado4X0 | Vida: 2 | Fila: 4 | Columna: 0
SOLDADO 4:
Nombre: Soldado7X5 | Vida: 1 | Fila: 7 | Columna: 5
SOLDADO 5:
Nombre: Soldado2X1 | Vida: 1 | Fila: 2 | Columna: 1
```

**Con ayuda del algoritmo burbuja y de Selección se ordenan y se muestran**

**¡El ejercito azul gana por mayor nivel de vida!  
Azul 25:15 Rojo**


**Finalmente, se imprime el ejército ganador y los puntajes de acuerdo al criterio de mayor vida total.**

## II. PRUEBAS

*¿Con qué valores comprobaste que tu práctica estuviera correcta?*

*Con valores int y String, 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?*

	<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>

*Esperaba que se almacenara dentro del objeto y atributo que quería; esperaba no tener errores, pero tuve varios al momento de construir un ArrayList con valores nulos, pronto lo resolví con otro enfoque.*

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

*Obtuve una secuencia limpia de los métodos usados en el main y sin ningún error.*

### III. CUESTIONARIO:

#### PRUEBAS DE COMMIT HECHO EN CONSOLA:

```
PS D:\FP2 LABORATORIO\PF2> 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_07/VideoJuego7.java

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\FP2 LABORATORIO\PF2> git add .
PS D:\FP2 LABORATORIO\PF2> 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_07/VideoJuego7.java
```

*Después de los cambios uso los comandos de siempre para trackear mis archivos y añadirlos a la nube.*

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

```
PS D:\FP2 LABORATORIO\PF2> git commit -m "Se hizo iterativo y método para iniciar
[main ad0fca7] Se hizo iterativo y método para iniciar
1 file changed, 22 insertions(+)
PS D:\FP2 LABORATORIO\PF2> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 753 bytes | 753.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/F4brici0L4yme/PF2.git
ee04fc4..ad0fca7 main -> main
PS D:\FP2 LABORATORIO\PF2> |
```

*Ahora mi commit y el git push a mi repositorio.*

 LAYME_SALAS_LABORATORIO_07	Se hizo iterativo y método para iniciar	1 minute ago
---	---	--------------

*Ya está reflejado en mi repositorio.*

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

## CONCLUSIONES

*Fue interesante combinar arrays estándar con ArrayList, para mí los ArrayList son los más versátiles, pero con muchos métodos disponibles, lo que podría generar código largo innecesario. Por otro lado, los arreglos estándar son buenos para mantenerlo simple, los usé para los ordenamientos y resultó más sencillo que hacerlo con ArrayList.*

## 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	<b>X</b>	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	<b>X</b>	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	<b>X</b>	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	<b>X</b>	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El	2	<b>X</b>	2	

	profesor puede preguntar para refrendar calificación).				
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	
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	