

	<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	15/10/2024	HORA DE PRESENTACIÓN	18:30:00		
INTEGRANTE (s) Layme Salas Rodrigo Fabricio				NOTA (0-20)	
DOCENTE(s): Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p style="text-align: center;">CLASE VideoJuego7</p> <p style="text-align: center;"><i>(Las explicaciones están en los comentarios dentro del código)</i></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 LA COMPARACIÓN
7      public static Soldado mayorVidaRojo = new Soldado(nombre:null, vida:0, fila:0, columna:0, equipo:null);
8      public static Soldado[] soldadosUniDimensionalAzul = new Soldado[10]; // ARREGLO ESTÁNDAR UTILIZADO PARA ORDENAMIENTOS
9      public static Soldado[] soldadosUniDimensionalRojo = new Soldado[10]; // ARREGLO ESTÁNDAR UTILIZADO PARA ORDENAMIENTOS
10
11     public static void main(String[] args) {
12         ArrayList<ArrayList<Soldado>> tablero = new ArrayList<ArrayList<Soldado>>(); //INICIALIZO MI ARRAYLIST BIDIMENSIONAL
13         int cantidad = (int)(Math.random() * 10 + 1);
14         int cantidadEnemiga = (int)(Math.random() * 10 + 1);
15         for(int i = 0; i<10; i++){ //INICIALIZAR 10 FILAS
16             tablero.add(new ArrayList<Soldado>());
17             for(int j = 0; j<10; j++){ //INICIALIZAR 10 COLUMNAS
18                 tablero.get(i).add(e:null); // INICIALIZA CON VALOR NULL PARA AHORRAR MEMORIA
19             }
20             inicializarEjercito(tablero, cantidad, color:"\u001B[44mSoldado", equipo:"azul"); // INICIALIZAR DATOS DE LOS EJERCITOS
21             inicializarEjercito(tablero, cantidadEnemiga, color:"\u001B[41mSoldado", equipo:"rojo");
22             mostrarTabla(tablero); // MUESTRA TABLA
23             hallarSoldadoMayorVida(tablero);
24             System.out.println("El soldado con mayor vida del ejército azul es: " + mayorVidaAzul);
25             System.out.println("El soldado con mayor vida del ejército rojo es: " + mayorVidaRojo);
26             System.out.println("El promedio del ejército azul es: " + vidaTotalAzul/cantidad);
27             System.out.println("El promedio del ejército rojo es: " + vidaTotalRojo/cantidadEnemiga);
28             System.out.println(x:"DATOS DEL EJÉRCITO AZUL POR ORDEN DE INGRESO:");
29             imprimirInformacion(cantidad, soldadosUniDimensionalAzul); // IMPRIME LA INFORMACIÓN CON toString
30             System.out.println(x:"DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:");
31             imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
32             rankingDePoder(cantidad, soldadosUniDimensionalAzul); // ORDENA POR MÉTODO BURBUJA
33             ordenarSelección(cantidadEnemiga, soldadosUniDimensionalRojo); // ORDENA POR MÉTODO SELECCIÓN
34             System.out.println(x:"DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:");
35             imprimirInformacion(cantidad, soldadosUniDimensionalAzul);
36             System.out.println(x:"DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:");
37             imprimirInformacion(cantidadEnemiga, soldadosUniDimensionalRojo);
38             mostrarGanador(); // MUESTRA EL GANADOR, CRITERIO: VIDA TOTAL DE LOS EJÉRCITOS
39     }

```

He reducido líneas de código al eliminar métodos innecesarios que estaban en el laboratorio anterior.

```

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(" " + tablero.get(i).get(j).getNombre()); // CARGA EL NOMBRE DEL SOLDADO SI EXISTE
66         }
67         System.out.println(x:""); // 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 }
95 public static void ordenarSeleccion(int cantidad, Soldado[] soldadosUniDimensional) { // 2DO ALGORITMO DE ORDENAMIENTO
96     for (int i = 0; i < cantidad - 1; i++) { //SE USA LA CANTIDAD PARA EVITAR QUE EL BUCLE SEÑALE A OBJETOS NULL
97         int menor = i;
98         for (int j = i + 1; j < cantidad; j++)
99             if (soldadosUniDimensional[j].getVida() > soldadosUniDimensional[menor].getVida())
100                 menor = j; // Almacena la posición del menor
101         Soldado temp = soldadosUniDimensional[menor]; // Intercambio de elementos
102         soldadosUniDimensional[menor] = soldadosUniDimensional[i];
103         soldadosUniDimensional[i] = temp;
104     }
105 }
106 public static void imprimirInformacion(int cantidad, Soldado[] soldadosUniDimensional){
107     for(int i = 0; i<cantidad; i++){
108         System.out.println("SOLDADO " + i + ":");
109         System.out.println(soldadosUniDimensional[i].toString()); //ME APOYO DE UNA MATRIZ UNIDIMENSIONAL PARA NO
110         // DESPERDIJAR MEMORIA
111     }
112 }

```

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

Soldado1X0	Soldado1X1	Soldado1X2							
						Soldado2X6			
								Soldado4X8	
	Soldado5X1					Soldado5X6			
				Soldado7X4					Soldado6X9
Soldado8X0		Soldado8X2							

Primero se muestra la tabla con los soldados posicionados diferenciados por el color ROJO-AZUL.

```
El soldado con mayor vida del ejército azul es: Nombre: Soldado1X0 | Vida: 5 | Fila: 1 | Columna: 0
El soldado con mayor vida del ejército rojo es: Nombre: Soldado1X2 | Vida: 5 | Fila: 1 | Columna: 2
El promedio del ejército azul es: 3
El promedio del ejército rojo es: 3
```

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: Soldado6X9 | Vida: 3 | Fila: 6 | Columna: 9
SOLDADO 1:
Nombre: Soldado8X2 | Vida: 5 | Fila: 8 | Columna: 2
SOLDADO 2:
Nombre: Soldado7X4 | Vida: 5 | Fila: 7 | Columna: 4
SOLDADO 3:
Nombre: Soldado1X0 | Vida: 5 | Fila: 1 | Columna: 0
SOLDADO 4:
Nombre: Soldado8X0 | Vida: 1 | Fila: 8 | Columna: 0
DATOS DEL EJÉRCITO ROJO POR ORDEN DE INGRESO:
SOLDADO 0:
Nombre: Soldado2X6 | Vida: 3 | Fila: 2 | Columna: 6
SOLDADO 1:
Nombre: Soldado1X2 | Vida: 5 | Fila: 1 | Columna: 2
SOLDADO 2:
Nombre: Soldado1X1 | Vida: 4 | Fila: 1 | Columna: 1
SOLDADO 3:
Nombre: Soldado4X8 | Vida: 2 | Fila: 4 | Columna: 8
SOLDADO 4:
Nombre: Soldado5X6 | Vida: 4 | Fila: 5 | Columna: 6
SOLDADO 5:
Nombre: Soldado5X1 | Vida: 5 | Fila: 5 | Columna: 1
```

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

```

DATOS DEL EJÉRCITO AZUL ORDENADO POR NIVEL DE VIDA:
SOLDADO 0:
Nombre: Soldado8X2 | Vida: 5 | Fila: 8 | Columna: 2
SOLDADO 1:
Nombre: Soldado7X4 | Vida: 5 | Fila: 7 | Columna: 4
SOLDADO 2:
Nombre: Soldado1X0 | Vida: 5 | Fila: 1 | Columna: 0
SOLDADO 3:
Nombre: Soldado6X9 | Vida: 3 | Fila: 6 | Columna: 9
SOLDADO 4:
Nombre: Soldado8X0 | Vida: 1 | Fila: 8 | Columna: 0
DATOS DEL EJÉRCITO ROJO ORDENADO POR NIVEL DE VIDA:
SOLDADO 0:
Nombre: Soldado1X2 | Vida: 5 | Fila: 1 | Columna: 2
SOLDADO 1:
Nombre: Soldado5X1 | Vida: 5 | Fila: 5 | Columna: 1
SOLDADO 2:
Nombre: Soldado1X1 | Vida: 4 | Fila: 1 | Columna: 1
SOLDADO 3:
Nombre: Soldado5X6 | Vida: 4 | Fila: 5 | Columna: 6
SOLDADO 4:
Nombre: Soldado2X6 | Vida: 3 | Fila: 2 | Columna: 6
SOLDADO 5:
Nombre: Soldado4X8 | Vida: 2 | Fila: 4 | Columna: 8
¡El ejército rojo gana por mayor nivel de vida!
  
```

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

```

¡El ejército rojo gana por mayor nivel de vida!
Azul 19:23 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?

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.

	<p>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

III. CUESTIONARIO:

PRUEBAS DE COMMIT HECHO EN CONSOLA:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

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/Soldado.java
        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/Soldado.java
        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.

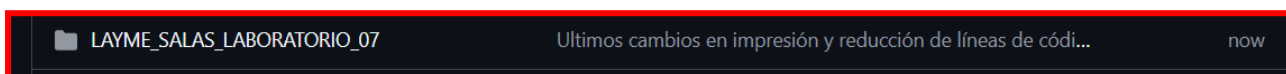
```
PS D:\FP2 LABORATORIO\PF2> git commit -m "Ultimos cambios en impresión y reducción de líneas de código"
[main 52ac16b] Ultimos cambios en impresión y reducción de líneas de código
 2 files changed, 2 insertions(+), 2 deletions(-)
PS D:\FP2 LABORATORIO\PF2> git push
To https://github.com/F4brici0L4yme/PF2.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/F4brici0L4yme/PF2.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
PS D:\FP2 LABORATORIO\PF2> git pull --rebase
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1011 bytes | 112.00 KiB/s, done.
From https://github.com/F4brici0L4yme/PF2
   c9d3237..814591b  main      -> origin/main
Successfully rebased and updated refs/heads/main.
```

	<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>

Al hacer el commit y el push me salió un error porque había una modificaciones que hice en github y no en mi repositorio local. Lo solucioné con git pull –rebase e hice push sin problemas.

```
PS D:\FP2 LABORATORIO\PF2> git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 556 bytes | 185.00 KiB/s, done.
Total 5 (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
 814591b..df44151 main -> main
PS D:\FP2 LABORATORIO\PF2> |
```

Aquí está mi git push después de arreglar el error.



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

	<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: 8</p>

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: 10</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	