
	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	<i>Fundamentos de la programación 02</i>				
TÍTULO DE LA PRÁCTICA:	<i>Combinando Arreglos estándar y ArrayList</i>				
NÚMERO DE PRÁCTICA:	<i>07</i>	AÑO LECTIVO:	<i>2024-B</i>	NRO. SEMESTRE:	<i>//</i>
FECHA DE PRESENTACIÓN	<i>15/11/2024</i>	HORA DE PRESENTACIÓN	<i>18:00:00</i>		
INTEGRANTE (s) <i>Mauro Snayder Sullca Mamani</i>				NOTA (0-20)	
DOCENTE(s): <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS	
I. EJERCICIOS RESUELTOS:	<pre> 7 /** 8 * 9 * @author Usuario24B 10 */ 11 public class Soldado { 12 // Creamos los atributos 13 private String nombre= " "; 14 private int vida; 15 private int fila; 16 private int columna; 17 private String color; 18 19 // Creamos los Set y Get de cada atributo 20 public String getNombre() { 21 return nombre; 22 } 23 24 public void setNombre(String nombre) { 25 this.nombre = nombre; 26 } 27 28 //Metodo para saber a que tipo de ejercito pertenece el soldado 29 public void setColor(String color){ 30 this.color=color; 31 } 32 33 public int getVida() { 34 return vida; 35 } 36 37 //Metodo para darle color a "vida" 38 public void printVida(){ 39 System.out.print(color+vida+"\u001B[0m"); 40 } </pre>

```

41
42 public int getFila() {
43     return fila;
44 }
45
46 public int getColumna() {
47     return columna;
48 }
49 //Generamos una posicion aleatoria para el soldado
50 public void aleatorioPosicion(int fila,int columna){
51     this.fila=(int) (Math.random()*fila);
52     this.columna=(int) (Math.random()*columna);
53 }
54 //Generamos la vida del soldado
55 public void aleatorioVida(){
56     this.vida=(int) (Math.random()*5+1);
57 }
58 // Creamos el toString
59 public String toString() {
60     return "Soldado{" + "nombre=" + nombre + ", vida=" + vida + ", fila=" + fila + ", columna=" + columna + "}\n";
61 }
62 }

```

```



8
9  * @author Mauro Snayder
10  */
11  import java.util.*;
12
13  public class VideoJuego4 {
14      public static void main(String[] args) {
15          Scanner scan=new Scanner(System.in);
16
17          while(true){//Creamos un bucle para hacerlo iterativo
18              Soldado[][] tabla=new Soldado[10][10];//Creamos la tabla
19              ArrayList<Soldado> ejercito1=new ArrayList<Soldado>();//Creamos ejercito 1
20              ArrayList<Soldado> ejercito2=new ArrayList<Soldado>();//Creamos ejercito 2
21              for (int i=0;i<tabla.length;i++){//Inicializamos el tablero con valores vacios
22                  for (int j=0;j<tabla[i].length;j++){
23                      tabla[i][j]=new Soldado();
24                  }
25              }
26              inicializarEjercito(tabla,ejercito1,"\033[1;31m");//El ejercito 1 será de color rojo
27              inicializarEjercito(tabla,ejercito2,"\033[1;34m");//El ejercito 2 sea de color azul
28              mostrarEjercitoTabla(tabla);
29              System.out.print("\nEl soldado con mayor vida del ejercito 1 es: "+mayorVida(ejercito1).toString());
30              System.out.print("\nEl soldado con mayor vida del ejercito 2 es: "+mayorVida(ejercito2).toString());
31              System.out.println("\nEl promedio de la vida del ejercito 1 es: "+vidaPromedio(ejercito1));
32              System.out.println("\nEl promedio de la vida del ejercito 2 es: "+vidaPromedio(ejercito2));
33              System.out.println("\nLista de los soldados por orden de creacion: ");
34              mostrarEjercitoOrdenCreacion(ejercito1,1);
35              mostrarEjercitoOrdenCreacion(ejercito2,2);
36              System.out.println("\nEl ranking de los soldados es: ");
37              rankingSoldadosV1(ejercito1,1);
38              rankingSoldadosV2(ejercito2,2);
39              ganador(ejercito1,ejercito2);
40              System.out.println("\nDesea generar otros ejercitos(?) 1=Si 0=No");
41              int opcion=scan.nextInt();
42              if (opcion==0)
43                  break;
44          }
45      }
46      // Método para inicializar una tabla con cierto numeros de soldados
47      public static void inicializarEjercito(Soldado[][] tabla,ArrayList<Soldado> ejercito,String color){
48          int numSoldados=(int) (Math.random()*10+1);//Genemos la cantidad de soldados
49          for (int i=0;i<numSoldados;i++){
50              Soldado persona=new Soldado();//creamos "persona" para luego ponerlo dentro del tablero y del Array del ejercito
51              persona.setColor(color);//ponemos el color al soldado
52              do {
53                  persona.aleatorioPosicion(tabla.length,tabla[0].length);//generamos una posicion
54                  persona.aleatorioVida();//generamos la vida
55                  persona.setNombre("soldado"+" "+persona.getFila()+"X"+persona.getColumna());//Generamos el nombre del soldado
56              }

```

```

57 //El bucle se repite si en una posicion aleatoria ya existe un soldado puesto
58 while(!tabla[persona.getFila()][persona.get columna()].getNombre().equals(""));
59 ejercito.add(persona); //ponemos el soldado dentro del Array del ejercito
60 tabla[persona.getFila()][persona.get columna()]=persona; //ponemos el soldado dentro del tablero
61 }
62 }
63 // Metodo para mostrar la tabla
64 public static void mostrarEjercitoTabla(Soldado[][] tabla){
65     System.out.println("
66     for (int i=0;i<tabla.length;i++){
67         System.out.print("
68     for (int j=0;j<tabla[i].length;j++){
69         if (tabla[i][j].getVida()==0){ //Si la vida es 0, entonces no hay un soldado en esa posicion
70             System.out.print("
71         else { //Caso contrario, si existe un soldado en esa posicion
72             System.out.print("
73             tabla[i][j].printVida(); //imprimimos la vida del soldado
74             System.out.print("
75         }
76     }
77     System.out.println();
78     System.out.println("
79 }
80 }
81 //Metodo para determinar el soldado con mayor vida
82 public static Soldado mayorVida(ArrayList<Soldado> ejercito){
83     Soldado mayor=new Soldado(); //creamos un objeto para almacenar al soldado mayor
84     for (Soldado persona:ejercito){ //Recorremos todo el Array del ejercito
85         if (persona.getVida()>mayor.getVida()) //Buscamos al soldado con mayor vida
86             mayor=persona;
87     }
88     return mayor;
89 }
90 //Metodo para determinar la vida total de todos los soldados
91 public static double vidaPromedio(ArrayList<Soldado> ejercito){
92     double vidaT=0; //vida inicial
93     for (Soldado persona:ejercito){ //Recorremos todo el Array del ejercito
94         vidaT+=persona.getVida(); //Sumamos las vidas de todos los soldados
95     }
96     return vidaT/ejercito.size();
97 }
98 //Metodo para ver las lista de los soldados por el orden de creacion
99 public static void mostrarEjercitoOrdenCreacion(ArrayList<Soldado> ejercito,int tipo){
100     System.out.println("Ejercito "+tipo+" : ");
101     for (Soldado persona:ejercito){ //imprimimos todos los soldados del Array ejercito
102         System.out.print(persona.toString());
103     }
104 }
105 //Metodo para ver el ranking de los soldados version 1(por vida)
106 public static void rankingSoldadosV1(ArrayList<Soldado> ejercito,int tipo){
107     ordenamientoBurbuja(ejercito); //ordenamos el Array Unidimensional
108     System.out.println("Ejercito "+tipo+" : ");
109     for (int i=ejercito.size()-1;i>0;i--){ //imprimimos el Array
110         System.out.print(ejercito.get(i).toString());
111     }
112 }
113 //Metodo para ver el ranking de los soldados version 2(por vida)
114 public static void rankingSoldadosV2(ArrayList<Soldado> ejercito,int tipo){
115     ordenamientoInsercion(ejercito); //ordenamos el Array Unidimensional
116     System.out.println("Ejercito "+tipo+" : ");
117     for (int i=ejercito.size()-1;i>0;i--){ //imprimimos el Array
118         System.out.print(ejercito.get(i).toString());
119     }
120 }
121 //Metodo de ordenamiento Burbuja para la vida de los soldados
122 public static void ordenamientoBurbuja(ArrayList<Soldado> lista){
123     Soldado cambio;
124     for(int i=0;i<lista.size()-1;i++){
125         for(int j=0;j<lista.size()-1-j;j++){
126             if (lista.get(j).getVida()>lista.get(j+1).getVida()){
127                 cambio=lista.get(j);
128                 lista.set(j,lista.get(j+1));
129                 lista.set(j+1,cambio);
130             }
131         }
132     }
133 }

```

	<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: 4</p>

```

134 //Metodo de ordenamiento de insercion para la vida de los soldados
135 public static void ordenamientoInsercion(ArrayList<Soldado> lista) {
136     for (int i=1;i<lista.size();i++){
137         Soldado soldadoActual=lista.get(i);
138         int j=i-1;
139         while (j>=0 && lista.get(j).getVida()>soldadoActual.getVida()){
140             lista.set(j+1,lista.get(j));
141             j--;
142         }
143         lista.set(j+1,soldadoActual);
144     }
145 }
146 //Determinar el ganador de la batalla (por la cantidad de soldados que tiene cada ejercito)
147 //Gana el ejercito que tiene mas soldados.
148 public static void ganador(ArrayList<Soldado> ejercito1,ArrayList<Soldado> ejercito2){
149     if (ejercito1.size()>ejercito2.size())
150         System.out.println("\nGana el ejercito 1.");
151     else if (ejercito1.size()<ejercito2.size())
152         System.out.println("\nGana el ejercito 2.");
153     else
154         System.out.println("\nQuedan empatados los ejércitos.");
155 }
156 }

```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

Comprobé la práctica utilizando diferentes configuraciones de entrada, como cantidades variables de soldados que iban desde 1 hasta 10 por ejército, y vida asignada a los soldados en un rango de 1 a 5. Además, revisé las posiciones generadas aleatoriamente en el tablero de 10x10 para asegurarme de que no se repitieran y que los métodos funcionaran correctamente en diferentes escenarios.

¿Qué resultado esperabas obtener para cada valor de entrada?

Esperaba obtener un tablero con celdas que mostraran correctamente la vida de los soldados y espacios vacíos donde no hubiera soldados. También anticipé que los métodos para calcular el promedio de vida reflejarían con precisión la suma de las vidas dividida por la cantidad de soldados de cada ejército. Además, esperaba que se identificara al soldado con mayor vida de forma correcta y que los rankings de soldados estuvieran ordenados de menor a mayor vida. Finalmente, esperaba que el método de determinación del ganador mostrara correctamente al ejército con más soldados o indicara empate si ambos tenían la misma cantidad.

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

Los comportamientos obtenidos confirmaron mis expectativas. El tablero se generó adecuadamente, mostrando la vida de los soldados y celdas vacías donde correspondía. El cálculo del promedio de vida se hizo con precisión, incluso para diferentes cantidades de soldados. La identificación del soldado con mayor vida fue correcta en todas las pruebas, y los rankings se ordenaron correctamente tanto con el método de burbuja como con el de inserción. El método para determinar al ganador funcionó como se esperaba, indicando al ejército con más soldados o declarando un empate cuando era el caso.

La primera ejecución del programa podemos ver que se generó una tabla 10x10 con 7 soldados en total. En el primer ejercito (rojo) se crearon 5 soldados donde el soldado con mayor vida es el soldado 6x0 y el promedio de vida es de 3.0, mientras que en el segundo ejército (azul) se creó 2 soldados donde el soldado con mayor vida es el soldado 3x3 y el promedio de vida es 3. Y por último nos muestra la lista de los soldados por orden de creación (por ejercito), por orden de vida (por ejercito) y el ganador de la batalla (por la cantidad de soldados por ejercito).

Output - Laboratorios (run)

run:

			4						
	3		4		1		2		
						2			
5									

El soldado con mayor vida del ejercito 1 es: Soldado{nombre=soldado 6X0, vida=5, fila=6, columna=0}
El soldado con mayor vida del ejercito 2 es: Soldado{nombre=soldado 3X3, vida=4, fila=3, columna=3}

El promedio de la vida del ejercito 1 es: 3.0
El promedio de la vida del ejercito 2 es: 3.0

Lista de los soldados por orden de creacion:

Ejercito 1 :

Soldado{nombre=soldado 6X0, vida=5, fila=6, columna=0}
Soldado{nombre=soldado 3X6, vida=1, fila=3, columna=6}
Soldado{nombre=soldado 3X8, vida=2, fila=3, columna=8}
Soldado{nombre=soldado 2X3, vida=4, fila=2, columna=3}
Soldado{nombre=soldado 3X1, vida=3, fila=3, columna=1}

Ejercito 2 :

Soldado{nombre=soldado 5X7, vida=2, fila=5, columna=7}
Soldado{nombre=soldado 3X3, vida=4, fila=3, columna=3}

El ranking de los soldados es:

Ejercito 1 :

Soldado{nombre=soldado 6X0, vida=5, fila=6, columna=0}
Soldado{nombre=soldado 2X3, vida=4, fila=2, columna=3}
Soldado{nombre=soldado 3X1, vida=3, fila=3, columna=1}
Soldado{nombre=soldado 3X8, vida=2, fila=3, columna=8}
Soldado{nombre=soldado 3X6, vida=1, fila=3, columna=6}

Ejercito 2 :

Soldado{nombre=soldado 3X3, vida=4, fila=3, columna=3}
Soldado{nombre=soldado 5X7, vida=2, fila=5, columna=7}

Gana el ejercito 1.

Desea generar otros ejercitos(?) 1=Si 0=No

La segunda ejecución del programa podemos ver que se generó una tabla 10x10 con 19 soldados en total. En el primer ejército (rojo) se crearon 9 soldados donde el soldado con mayor vida es el soldado 7x9 y el promedio de vida es de 2.8, mientras que en el segundo ejército (azul) se creó 10 soldados donde el soldado con mayor vida es el soldado 1x5 y el promedio de vida es 2.9. Y por último nos muestra la lista de los soldados por orden de creación (por ejército), por orden de vida (por ejército) y el ganador de la batalla (por la cantidad de soldados por ejército).

Output - Laboratorios (run)

Desea generar otros ejercitos (?) 1=Si 0=No
1

4				4	5			1	
				5					
			4						
			2	2					
	1	2	3					2	
				4				1	
3		3						2	5
					2				

El soldado con mayor vida del ejercito 1 es: Soldado[nombre=soldado 7X9, vida=5, fila=7, columna=9]
El soldado con mayor vida del ejercito 2 es: Soldado[nombre=soldado 1X5, vida=5, fila=1, columna=5]

El promedio de la vida del ejercito 1 es: 2.888888888888889
El promedio de la vida del ejercito 2 es: 2.9

Lista de los soldados por orden de creacion:

Ejercito 1 :

Soldado[nombre=soldado 7X0, vida=3, fila=7, columna=0]
Soldado[nombre=soldado 6X5, vida=4, fila=6, columna=5]
Soldado[nombre=soldado 4X4, vida=2, fila=4, columna=4]
Soldado[nombre=soldado 6X9, vida=1, fila=6, columna=9]
Soldado[nombre=soldado 7X2, vida=3, fila=7, columna=2]
Soldado[nombre=soldado 7X9, vida=5, fila=7, columna=9]
Soldado[nombre=soldado 5X1, vida=1, fila=5, columna=1]
Soldado[nombre=soldado 8X6, vida=2, fila=8, columna=6]
Soldado[nombre=soldado 2X4, vida=5, fila=2, columna=4]

Ejercito 2 :

Soldado[nombre=soldado 4X3, vida=2, fila=4, columna=3]
Soldado[nombre=soldado 5X8, vida=2, fila=5, columna=8]
Soldado[nombre=soldado 1X8, vida=1, fila=1, columna=8]
Soldado[nombre=soldado 1X0, vida=4, fila=1, columna=0]
Soldado[nombre=soldado 1X5, vida=5, fila=1, columna=5]
Soldado[nombre=soldado 5X2, vida=2, fila=5, columna=2]
Soldado[nombre=soldado 1X4, vida=4, fila=1, columna=4]
Soldado[nombre=soldado 7X8, vida=2, fila=7, columna=8]
Soldado[nombre=soldado 3X3, vida=4, fila=3, columna=3]
Soldado[nombre=soldado 5X3, vida=3, fila=5, columna=3]

El ranking de los soldados es:

Ejercito 1 :

Soldado[nombre=soldado 2X4, vida=5, fila=2, columna=4]
Soldado[nombre=soldado 7X9, vida=5, fila=7, columna=9]
Soldado[nombre=soldado 6X5, vida=4, fila=6, columna=5]
Soldado[nombre=soldado 7X2, vida=3, fila=7, columna=2]
Soldado[nombre=soldado 7X0, vida=3, fila=7, columna=0]
Soldado[nombre=soldado 8X6, vida=2, fila=8, columna=6]
Soldado[nombre=soldado 4X4, vida=2, fila=4, columna=4]
Soldado[nombre=soldado 5X1, vida=1, fila=5, columna=1]
Soldado[nombre=soldado 6X9, vida=1, fila=6, columna=9]

Ejercito 2 :

Soldado[nombre=soldado 1X5, vida=5, fila=1, columna=5]
Soldado[nombre=soldado 3X3, vida=4, fila=3, columna=3]
Soldado[nombre=soldado 1X4, vida=4, fila=1, columna=4]
Soldado[nombre=soldado 1X0, vida=4, fila=1, columna=0]
Soldado[nombre=soldado 5X3, vida=3, fila=5, columna=3]
Soldado[nombre=soldado 7X8, vida=2, fila=7, columna=8]
Soldado[nombre=soldado 5X2, vida=2, fila=5, columna=2]
Soldado[nombre=soldado 5X8, vida=2, fila=5, columna=8]
Soldado[nombre=soldado 4X3, vida=2, fila=4, columna=3]
Soldado[nombre=soldado 1X8, vida=1, fila=1, columna=8]

Gana el ejercito 2.

La tercera ejecución del programa podemos ver que se generó una tabla 10x10 con 11 soldados en total. En el primer ejército (rojo) se crearon 3 soldados donde el soldado con mayor vida es el soldado 9x4 y el promedio de vida es de 3.0, mientras que en el segundo ejército (azul) se creó 8 soldados donde el soldado con mayor vida es el soldado 7x4 y el promedio de vida es 2.5. Y por último nos muestra la lista de los soldados por orden de creación (por ejército), por orden de vida (por ejército) y el ganador de la batalla (por la cantidad de soldados por ejército).

Output - Laboratorios (run)


Desea generar otros ejercitos(?) 1=Si 0=No
1

									2
			3						
							1		
	3		1						
	4								
	1				5				
		1	3						
					5				

El soldado con mayor vida del ejercito 1 es: Soldado{nombre=soldado 9X4, vida=5, fila=9, columna=4}
El soldado con mayor vida del ejercito 2 es: Soldado{nombre=soldado 7X4, vida=5, fila=7, columna=4}

El promedio de la vida del ejercito 1 es: 3.0
El promedio de la vida del ejercito 2 es: 2.5

Lista de los soldados por orden de creacion:



Ejercito 1 :
Soldado{nombre=soldado 8X3, vida=3, fila=8, columna=3}
Soldado{nombre=soldado 7X1, vida=1, fila=7, columna=1}
Soldado{nombre=soldado 9X4, vida=5, fila=9, columna=4}
Ejercito 2 :
Soldado{nombre=soldado 6X1, vida=4, fila=6, columna=1}
Soldado{nombre=soldado 1X3, vida=3, fila=1, columna=3}
Soldado{nombre=soldado 0X8, vida=2, fila=0, columna=8}
Soldado{nombre=soldado 5X3, vida=1, fila=5, columna=3}
Soldado{nombre=soldado 4X7, vida=1, fila=4, columna=7}
Soldado{nombre=soldado 5X1, vida=3, fila=5, columna=1}
Soldado{nombre=soldado 7X4, vida=5, fila=7, columna=4}
Soldado{nombre=soldado 8X2, vida=1, fila=8, columna=2}

El ranking de los soldados es:

Ejercito 1 :
Soldado{nombre=soldado 9X4, vida=5, fila=9, columna=4}
Soldado{nombre=soldado 8X3, vida=3, fila=8, columna=3}
Soldado{nombre=soldado 7X1, vida=1, fila=7, columna=1}
Ejercito 2 :
Soldado{nombre=soldado 7X4, vida=5, fila=7, columna=4}
Soldado{nombre=soldado 6X1, vida=4, fila=6, columna=1}
Soldado{nombre=soldado 5X1, vida=3, fila=5, columna=1}
Soldado{nombre=soldado 1X3, vida=3, fila=1, columna=3}
Soldado{nombre=soldado 0X8, vida=2, fila=0, columna=8}
Soldado{nombre=soldado 8X2, vida=1, fila=8, columna=2}
Soldado{nombre=soldado 4X7, vida=1, fila=4, columna=7}
Soldado{nombre=soldado 5X3, vida=1, fila=5, columna=3}

Gana el ejercito 2.

Desea generar otros ejercitos(?) 1=Si 0=No

	<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>

III. COMMITS:

Entramos a nuestra carpeta donde están nuestros archivos y añadimos los cambios.

```

MINGW64/c/Users/Mauro Snayder/Documents/NetBeansProjects/Laboratorios/src
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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:   Laboratorio_07/Soldado.java
        modified:   Laboratorio_07/VideoJuego4.java

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

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git add .

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Laboratorio_07/Soldado.java
        modified:   Laboratorio_07/VideoJuego4.java

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```

Hacemos un commit

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git commit -m "Culminado"
[master fcf2d03] Culminado
 2 files changed, 12 insertions(+), 10 deletions(-)

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```


Subimos nuestro commit a nuestro repositorio remoto

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 617 bytes | 617.00 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/MauroSullcaMamani/FDLP2_LAB.git
 4e04476..fcf2d03 master -> master
branch 'master' set up to track 'origin/master'.
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ !

```

Verificación de que se hizo el commit

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git log
commit fcf2d035e22207099e3a3f1cb294c0dd1f4cc35d (HEAD -> master, origin/master)
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Fri Nov 15 16:38:20 2024 -0500

    Culminado

commit 4e0447646e6133205d6e90c80810aef118b0009f
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Tue Nov 12 11:34:24 2024 -0500

    terminado

commit c25aa3b2b1f934d02b3a43cff59abfe6212a49f7
Author: MauroSullcaMamani <msullcam@unsa.edu.pe>
Date: Tue Nov 12 10:13:06 2024 -0500

    avance en labs



```

Link de mi repositorio: https://github.com/MauroSullcaMamani/FDLP2_LAB.git

IV. RUBRICA:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	✓	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	✓	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	✓	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	✓	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	✓	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	✓	3	
TOTAL		20		18	

Tabla 2: Rúbrica para contenido del Informe y demostración

	<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: 10</p>

CONCLUSIONES

En conclusión, el uso combinado de arreglos estándar y ArrayList en la práctica permitió manejar eficientemente estructuras de datos complejas. Los arreglos bidimensionales fueron esenciales para representar un tablero de 10x10 con posiciones estáticas, facilitando la visualización y asignación de soldados en ubicaciones específicas. A su vez, los ArrayList brindaron la flexibilidad necesaria para gestionar ejércitos de tamaño variable, permitiendo la adición, manipulación y ordenamiento de soldados de forma dinámica. Esta combinación proporcionó un equilibrio entre estructura fija y adaptabilidad, lo cual resultó fundamental para implementar métodos que calcularan el promedio de vida, identificaran al soldado con mayor vida y ordenaran los ejércitos de manera eficiente.

METODOLOGÍA DE TRABAJO

Lo primero que hice, es leer cada los enunciados de cada ejercicio y también tomar en cuenta las restricciones que nos da para así poder buscar una solución al problema. Después observar el código y entender la funcionalidad de cada uno y completar las partes que están incompletas. Y por último comprobar nuestro código ingresando varias veces valores de prueba para ver que nuestro código está funcionando correctamente.

REFERENCIAS Y BIBLIOGRAFÍA

E. G. Castro Gutiérrez and 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.