
	<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> 1

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	<i>Fundamentos de la programación 02</i>				
<b>TÍTULO DE LA PRÁCTICA:</b>	<i>Arreglos bidimensionales a objetos</i>				
<b>NÚMERO DE PRÁCTICA:</b>	<i>05</i>	<b>AÑO LECTIVO:</b>	<i>2024-B</i>	<b>NRO. SEMESTRE:</b>	<i>//</i>
<b>FECHA DE PRESENTACIÓN</b>	<i>18/10/2024</i>	<b>HORA DE PRESENTACIÓN</b>	<i>18:00:00</i>		
<b>INTEGRANTE (s)</b> <i>Mauro Snayder Sullca Mamani</i>				<b>NOTA (0-20)</b>	
<b>DOCENTE(s):</b> <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS	
<b>I. EJERCICIOS RESUELTOS:</b>	
<pre> 7     /** 8     * 9     * @author Mauro Snayder 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   18        // Creamos los Set y Get de cada atributo 19        public String getNombre() { 20            return nombre; 21        } 22   23        public void setNombre(String nombre) { 24            this.nombre = nombre; 25        } 26   27        public int getVida() { 28            return vida; 29        } 30   31        public int getFila(){ 32            return fila; 33        } 34   35        public int getColumna(){ 36            return columna; 37        } </pre>	

```

38 //Generamos una posicion aleatoria para el soldado
39 public void aleatorioPosicion(int fila,int columna){
40     this.fila=(int) (Math.random()*fila);
41     this.columna=(int) (Math.random()*columna);
42 }
43 //Generamos la vida del soldado
44 public void aleatorioVida(){
45     this.vida=(int) (Math.random()*5+1);
46 }
47 // Creamos el toString
48 public String toString() {
49     return "Soldado{" + "nombre=" + nombre + ", vida=" + vida + ", fila=" + fila + ", columna=" + columna + "}\n";
50 }
51 }

```

```



7 /**
8  *
9  * @author Usuario24B
10 */
11
12 public class Actividad05 {
13     public static void main(String[] args) {
14         // Inicializamos dos ejércitos con un número aleatorio de soldados entre 1 y 5
15         int fila=10;//Generamos la fila de la tabla
16         int columna=10;//Generamos la columna de la tabla
17         int numSoldados=(int) (Math.random()*10+1);//Generamos la cantidad de soldados
18         Soldado[][] ejercito=new Soldado[fila][columna];
19         for (int i=0;i<ejercito.length;i++){//inicializamos el arreglo con soldados sin datos
20             for (int j=0;j<ejercito[i].length;j++){
21                 ejercito[i][j]=new Soldado();
22             }
23             inicializarEjercito(ejercito,numSoldados);
24             mostrarEjercitoTabla(ejercito);
25             System.out.print("\nEl soldado con mayor vida es: "+mayorVida(ejercito).toString());
26             System.out.println("El promedio de la vida del ejercito es: "+vidaTotal(ejercito)/numSoldados);
27             System.out.println("La vida total del ejercito es: "+vidaTotal(ejercito));
28             System.out.println("\nLista de los soldados por orden de creacion: ");
29             mostrarEjercitoOrdenCreacion(ejercito,numSoldados);
30             System.out.println("\nEl ranking de los soldados es: ");
31             rankingSoldados(ejercito,numSoldados);
32         }
33     }
34     // Método para inicializar una tabla con cierto numeros de soldados
35     public static void inicializarEjercito(Soldado[][] ejercito,int numSoldados){
36         for (int i=0;i<numSoldados;i++){
37             Soldado persona=new Soldado();//creamos "persona" para luego ponerlo dentro del arreglo
38             persona.setNombre("soldado"+i);
39             do {
40                 persona.aleatorioPosicion(ejercito.length,ejercito[0].length);//generamos una posicion
41                 persona.aleatorioVida();//generamos la vida
42             }
43             //El bucle se repite si en una posicion aleatoria ya existe un soldado puesto
44             while(!ejercito[persona.getFila()][persona.getColumna()].getNombre().equals(""));
45             ejercito[persona.getFila()][persona.getColumna()]=persona;//ponemos el soldado dentro del arreglo
46         }
47     }
48     // Metodo para mostrar la tabla
49     public static void mostrarEjercitoTabla(Soldado[][] ejercito){
50         System.out.println(" ");
51         for (int i=0;i<ejercito.length;i++){
52             System.out.print("|");
53             for (int j=0;j<ejercito[i].length;j++){
54                 System.out.print(ejercito[i][j].getNombre()+"|");
55             }
56         }
57     }

```

```

56         System.out.println(" | _____ | _____ | _____ | _____ | _____ | _____ | _____ | _____ | ");
57     }
58 }
59 //Metodo para determinar el soldado con mayor vida
60 public static Soldado mayorVida(Soldado[][] ejercito){
61     Soldado mayor=new Soldado();//creamos un objeto para almacenar al soldado mayor
62     for (int i=0;i<ejercito.length;i++){
63         for (int j=0;j<ejercito[i].length;j++){
64             if (ejercito[i][j].getVida()>mayor.getVida())
65                 mayor=ejercito[i][j]; //actualizamos "mayor" si otro soldado tiene mayor vida
66         }
67     }
68     return mayor;
69 }
70 //Metodo para determinar la vida total de todos los soldados
71 public static double vidaTotal(Soldado[][] ejercito){
72     double vidaT=0; //vida inicial
73     for (int i=0;i<ejercito.length;i++){
74         for (int j=0;j<ejercito[i].length;j++){
75             vidaT+=ejercito[i][j].getVida(); //sumamos la vida de cada soldado
76         }
77     }
78     return vidaT;
79 }
80 //Metodo para ver las lista de los soldados por el orden de creacion
81 public static void mostrarEjercitoOrdenCreacion(Soldado[][] ejercito,int numSoldados){
82     Soldado[] list=new Soldado[numSoldados]; //creamos un arreglo unidimensional.
83     int copiar=0;
84     for (int j=0;j<ejercito.length;j++){ //Copiamos el arreglo bidimensional a un unidimensional
85         for (int k=0;k<ejercito[j].length;k++){ //ya que es mas facil ordenarlos
86             if (!ejercito[j][k].getNombre().equals(" ")){
87                 list[copiar]=ejercito[j][k];
88                 copiar++;
89             }
90         }
91     }
92     ordenamientoBurbuja(list); //ordenamos el arreglo unidimensional
93     for (int i=0;i<list.length;i++){ //imprimimos el arreglo
94         System.out.print(list[i].toString());
95     }
96 }
97 //Metodo para ver el ranking de los soldados (por vida)
98 public static void rankingSoldados(Soldado[][] ejercito,int numSoldados){
99     Soldado[] listRankT=new Soldado[numSoldados]; //creamos un arreglo unidimensional.
100    int copiar=0;
101    for (int j=0;j<ejercito.length;j++){ //Copiamos el arreglo bidimensional a un unidimensional
102        for (int k=0;k<ejercito[j].length;k++){ //ya que es mas facil ordenarlos
103            if (!ejercito[j][k].getNombre().equals(" ")){
104                listRankT[copiar]=ejercito[j][k];
105                copiar++;
106            }
107        }
108    }
109    ordenamientoInsercion(listRankT); //ordenamos el arreglo unidimensional
110    for (int i=0,j=listRankT.length-1;j>0;i++,j--){ //imprimimos el arreglo
111        System.out.print((i+1)+" -> "+listRankT[j].toString());
112    }
113 }
114 //Metodo de ordenamiento burbuja para los nombres de los soldados, ya que de esa
115 //manera podemos ver el orden de creacion de los soldados
116 public static void ordenamientoBurbuja(Soldado[] lista){
117     Soldado cambio;
118     for(int i=0;i<lista.length-1;i++){
119         for(int j=0;j<lista.length-i-1;j++){
120             if(lista[j].getNombre().compareTo(lista[j+1].getNombre())>0){
121                 cambio=lista[j];
122                 lista[j]=lista[j+1];
123                 lista[j+1]=cambio;
124             }
125         }
126     }
127 }

```

	<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> 4

```

127 //Metodo de ordenamiento de insercion para la vida de los soldados
128 public static void ordenamientoInsercion(Soldado[] lista) {
129     for (int i=1;i<lista.length;i++) {
130         Soldado soldadoActual=lista[i];
131         int j=i-1;
132         while (j>=0 && lista[j].getVida()>soldadoActual.getVida()){
133             lista[j+1]=lista[j];
134             j--;
135         }
136         lista[j+1]=soldadoActual;
137     }
138 }
139

```

## II. PRUEBAS

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

*Se utilizaron una tabla 10x10, así como el número de soldados, variando la posición entre 1 y 5 para las filas y columnas y entre 1 y 10 para la cantidad de soldados. Esto permitió evaluar cómo el programa manejaba diferentes configuraciones y asegurarse de que los soldados se asignaran correctamente a las posiciones en la tabla.*

*¿Qué resultado esperabas obtener para cada valor de entrada?*

*Se esperaba que, dependiendo de los valores de entrada, la tabla se llenara correctamente sin solapamientos de soldados. Además, se anticipaba que el programa identificara el soldado con mayor vida, calculase la vida total y el promedio de vida, y que mostrara la lista de soldados en el orden de creación, así como su ranking por vida.*

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

*Al ejecutar el programa, se observó que los resultados variaban por la aleatoriedad en las posiciones y vidas de los soldados. La lista de soldados se presentó en el orden de creación, y el ranking se mostró correctamente, indicando que el programa funcionó de acuerdo con lo esperado en general.*

La primera ejecución del programa podemos ver que se generó una tabla 10x10 y 4 soldados. El soldado con mayor vida es el “soldado2”, el promedio de vida es “3.75” y la vida total es “15”. Y por último nos muestra la lista de los soldados por orden de creación y por orden de vida.

```
Output - Mauro (run)
run:
[10x10 grid showing soldier positions]

El soldado con mayor vida es: Soldado[nombre=soldado2, vida=5, fila=5, columna=6]
El promedio de la vida del ejercito es: 3.75
La vida total del ejercito es: 15.0

Lista de los soldados por orden de creacion:
Soldado[nombre=soldado0, vida=2, fila=7, columna=1]
Soldado[nombre=soldado1, vida=4, fila=7, columna=3]
Soldado[nombre=soldado2, vida=5, fila=5, columna=6]
Soldado[nombre=soldado3, vida=4, fila=9, columna=4]

El ranking de los soldados es:
1 -> Soldado[nombre=soldado2, vida=5, fila=5, columna=6]
2 -> Soldado[nombre=soldado3, vida=4, fila=9, columna=4]
3 -> Soldado[nombre=soldado1, vida=4, fila=7, columna=3]
4 -> Soldado[nombre=soldado0, vida=2, fila=7, columna=1]
BUILD SUCCESSFUL (total time: 0 seconds)
```

La segunda ejecución del programa podemos ver que se generó una tabla 10x10 y 2 soldados. El soldado con mayor vida es el “soldado0”, el promedio de vida es “2.5” y la vida total es “5”. Y por último nos muestra la lista de los soldados por orden de creación y por orden de vida.

```
Output - Mauro (run)
run:
[10x10 grid showing soldier positions]

El soldado con mayor vida es: Soldado[nombre=soldado0, vida=3, fila=9, columna=0]
El promedio de la vida del ejercito es: 2.5
La vida total del ejercito es: 5.0

Lista de los soldados por orden de creacion:
Soldado[nombre=soldado0, vida=3, fila=9, columna=0]
Soldado[nombre=soldado1, vida=2, fila=9, columna=3]

El ranking de los soldados es:
1 -> Soldado[nombre=soldado0, vida=3, fila=9, columna=0]
2 -> Soldado[nombre=soldado1, vida=2, fila=9, columna=3]
BUILD SUCCESSFUL (total time: 0 seconds)
```

La tercera ejecución del programa podemos ver que se generó una tabla 10x10 y 10 soldados. El soldado con mayor vida es el “soldado6”, el promedio de vida es “3.0” y la vida total es “30”. Y por último nos muestra la lista de los soldados por orden de creación y por orden de vida.

Output - Mauro (run)

run:

									soldado8
					soldado6		soldado0		
	soldado1						soldado3		
	soldado7								
soldado9									
					soldado2			soldado4	
				soldado5					

El soldado con mayor vida es: Soldado[nombre=soldado6, vida=5, fila=2, columna=5]  
El promedio de la vida del ejercito es: 3.0  
La vida total del ejercito es: 30.0



Lista de los soldados por orden de creacion:

Soldado[nombre=soldado0, vida=1, fila=2, columna=7]  
Soldado[nombre=soldado1, vida=4, fila=3, columna=1]  
Soldado[nombre=soldado2, vida=2, fila=6, columna=5]  
Soldado[nombre=soldado3, vida=5, fila=3, columna=7]  
Soldado[nombre=soldado4, vida=1, fila=6, columna=9]  
Soldado[nombre=soldado5, vida=3, fila=7, columna=4]  
Soldado[nombre=soldado6, vida=5, fila=2, columna=5]  
Soldado[nombre=soldado7, vida=3, fila=4, columna=1]  
Soldado[nombre=soldado8, vida=4, fila=0, columna=9]  
Soldado[nombre=soldado9, vida=2, fila=5, columna=0]

El ranking de los soldados es:

1 -> Soldado[nombre=soldado3, vida=5, fila=3, columna=7]  
2 -> Soldado[nombre=soldado6, vida=5, fila=2, columna=5]  
3 -> Soldado[nombre=soldado1, vida=4, fila=3, columna=1]  
4 -> Soldado[nombre=soldado8, vida=4, fila=0, columna=9]  
5 -> Soldado[nombre=soldado5, vida=3, fila=7, columna=4]  
6 -> Soldado[nombre=soldado7, vida=3, fila=4, columna=1]  
7 -> Soldado[nombre=soldado2, vida=2, fila=6, columna=5]  
8 -> Soldado[nombre=soldado9, vida=2, fila=5, columna=0]  
9 -> Soldado[nombre=soldado4, vida=1, fila=6, columna=9]  
10 -> Soldado[nombre=soldado0, vida=1, fila=2, columna=7]

BUILD SUCCESSFUL (total time: 0 seconds)

	<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;"><b>Formato:</b> 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>

### 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_05/Actividad05.java
        modified:   Laboratorio_05/Soldado.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_05/Actividad05.java
        modified:   Laboratorio_05/Soldado.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 "acabado"
[master 3253f3e] acabado
 2 files changed, 124 insertions(+), 37 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), 2.26 KiB | 2.26 MiB/s, done.
Total 5 (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/MauroSullcaMamani/FDLP2_LAB.git
 9305b36..3253f3e master -> master
branch 'master' set up to track 'origin/master'.
```



Link de mi repositorio: [https://github.com/MauroSullcaMamani/FDLP2\\_LAB.git](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;"><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>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 9</p>

## CONCLUSIONES

*En conclusión, la utilización de arreglos bidimensionales de objetos en el programa, junto con técnicas de inicialización, búsqueda y ordenamiento, permite manejar colecciones de soldados de manera eficiente y organizada. Al trabajar con objetos, se pueden gestionar múltiples atributos, como nombre y vida, de forma estructurada, lo que facilita la manipulación y el acceso a la información. Las funcionalidades implementadas para identificar al soldado con mayor vida, calcular la vida total y promedio, y ordenar soldados por su creación y vida, mejoran la accesibilidad y optimizan el rendimiento del programa, proporcionando una base sólida para futuras mejoras.*

## 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.*