

# Informe de Laboratorio 07

## Tema: Arreglos Estandar y ArrayList

**Nota**

Estudiante	Escuela	Asignatura
Jhonatan David Arias Quispe jariasq@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de Programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
07	Arreglos Estandar y ArrayList	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Octubre 2023	Al 23 Octubre 2023

### 1. Actividades

- Cree un Proyecto llamado Laboratorio7
- Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
- Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.
- Tendrá 2 Ejércitos (utilizar la estructura de datos más adecuada). Inicializar el tablero con n soldados aleatorios entre 1 y 10 para cada Ejército. Cada soldado tendrá un nombre autogenerado: Soldado0X1, Soldado1X1, etc., un valor de puntos de vida autogenerado aleatoriamente [1..5], la fila y columna también autogenerados aleatoriamente (no puede haber 2 soldados en el mismo cuadrado). Se debe mostrar el tablero con todos los soldados creados y sus puntos de vida (usar caracteres como y otros y distinguir los de un ejército de los del otro ejército). Además de los datos del Soldado con mayor vida de cada ejército, el promedio de puntos de vida de todos los soldados creados por ejército, los datos de todos los soldados por ejército en el orden que fueron creados y un ranking de poder de todos los soldados creados por ejército (del que tiene más nivel de vida al que tiene menos) usando 2 diferentes algoritmos de ordenamiento. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacer el programa iterativo.

## 2. Equipos, materiales y temas utilizados

- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- OpenJDK 64-Bit 20.0.1
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- Programación Orientada a Objetos.
- Creacion de programas con CLI
- Biblioteca Graphics (origen propio)

## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/JhonatanDczel/fp2-23b.git>
- URL para el laboratorio 07 en el Repositorio GitHub.
- <https://github.com/JhonatanDczel/fp2-23b/tree/main/fase02/lab07>

## 4. Orden de archivos

Este es el arbol de directorios final

```
>>> lab07 git:(main) ✕ tree
.
├── graphics
│   ├── GPicture.class
│   ├── GPicture.java
│   ├── Graphics.class
│   ├── Graphics.java
│   ├── Picture$PIterator.class
│   ├── Picture.class
│   └── Picture.java
├── latex
│   ├── acts
│   │   ├── a1.tex
│   │   └── main.tex
│   ├── foot
│   │   ├── rubricas.aux
│   │   └── rubricas.tex
│   ├── head
│   │   ├── codeFormat.tex
│   │   ├── dataTable.tex
│   │   ├── format.tex
│   │   ├── labData.tex
│   │   └── pkgs.tex
│   ├── img
│   │   ├── logo_abet.png
│   │   ├── logo_episunsa.png
│   │   └── logo_unsa.jpg
│   ├── informe-latex.aux
│   ├── informe-latex.log
│   ├── informe-latex.out
│   ├── informe-latex.pdf
│   └── informe-latex.tex
├── Soldado.class
├── Soldado.java
├── VideoJuego.class
└── VideoJuego.java

7 directories, 28 files
```

## 5. Actividades iniciales: Crea un proyecto llamado Laboratorio 7

Se creó el directorio lab07, y se copió a él los archivos de los anteriores laboratorios para reciclar partes de código.

- Se importaron las clases Soldado y Videojuego
- El soldado tiene las especificaciones requeridas como atributos
  - Nombre
  - Puntos de vida
  - Fila
  - Columna
- Para el tablero, se sigue utilizando la biblioteca graphics, la única indicación es que se use la estructura de datos más adecuada, lo que me sugiere que es a libre elección, por lo tanto se usó un array bidimensional simple para el tablero

### 5.1. Regresando el modelo de ArrayList a array simple

- Como dije anteriormente, importamos las anteriores clases de videojuego, y en la clase que importamos, el tablero está representado con un ArrayList
- Para trabajar mejor se hizo el pase de ArrayList a array simple
- Se logró eso, cambiando los lugares donde trabaja ArrayList

## 6. Actividad principal

A continuación se cubrirán los requerimientos principales de este laboratorio

### 6.1. Iniciando 2 ejércitos

En este punto, se inicializan dos ejércitos con nombres autogenerados, para eso, se modifica el código de inicialización de ejércitos, para permitir ingresar un parámetro extra (Un número entero que indica el número de ejército)

Listing 1: Código fuente, Videojuego.java

```
1 public static void main(String[] args){
2     Soldado[] army1 = initializeArmy(0, true);
3     Soldado[] army2 = initializeArmy(1, true);
4     ...
5     public static void displayArmy(Soldado[] army, String str){
6         System.out.println("\n==== " + str + " =====");
7         for(Soldado soldier : army){
8             displaySoldier(soldier);
9         }
10    }
```

- Como vemos, los ejércitos se generan con un número que será lo que contendrá su nombre

## 6.2. Modificando los atributos en Soldado.java

Se agrego un nuevo atributo en soldado, que contendra la coloracion del soldado, en este primer caso solo tendra dos opciones de color

- Se agrego el atributo booleano negro
- Se agrego el metodo setter setNegro()
- Se agrego el metodo getter isNegro(), inicialmente se planteo llamarlo getNegro, pero al ser un valor booleano, y siguiendo las buenas practicas de programacion, se cambio a isNegro, para poder trabajar mas facilmente con su valor

Listing 2:Codigo fuente, Videojuego.java

```
1 public class Soldado{
2     public String name;
3     public int life;
4     public int row;
5     public int column;
6     public boolean negro = false;
7
8     ...
9     ...
10    ...
11
12
13    //SECCION DE SETERS
14
15    public void setNegro(boolean n){
16        this.negro = n;
17    }
18
19    ...
20    ...
21    ...
22
23    //SECCION DE GETERS
24
25    public boolean isNegro(){
26        return this.negro;
27    }
}
```

## 6.3. Implementacion del color del ejercito

- Una vez que ya tenemos una forma de distinguir un soldado, se pueden colorear de diferentes tonos
- Para eso se usa el metodo invertir(), incluido en graphics, que cambia o invierte el color de una ficha/pieza
- Al momento de inicializar un ejercito se tendra que especificar el color del mismo

Listing 3:Codigo fuente, Videojuego.java

```
1
2 public static void makeGBoard(){
3     for(int i = 0; i < 10; i++){
4         Picture fila = null;
5         for(int j = 0; j < 10; j++){
```

```
6   Picture c = Picture.casilleroBlanco();
7   if(board[i][j] != null){
8       c = Picture.soldier().superponer(c);
9       if(board[i][j].isNegro())
10          c = Picture.soldier().invertir().superponer(c);
11   }
```

- En esa porción de código, vemos el método que dibuja a los soldados por pantalla
- Se verifica si alguno tiene el método de `isNegro` como `true`, en caso de ser cierto, se invierte el color de un soldado y se agrega al tablero

## 6.4. Muestra de datos por consola

- La parte final de la actividad es básicamente lo que hicimos por 3 laboratorios seguidos, mostrar los datos de vida, ordenar por algún criterio, generar un ranking de soldados y mostrar sus niveles de vida así como el promedio y la vida más grande
- Por lo tanto solo mostraremos el código sin explicarnos en su funcionamiento

Listing 4: Código fuente, Videojuego.java

```
1   public static void main(String[] args){
2       Soldado[] army1 = initializeArmy(0, true);
3       Soldado[] army2 = initializeArmy(1, true);
4       displayArmy(army1, "Ejercito 1");
5       displayArmy(army2, "Ejercito 2");
6       System.out.println("Soldado con maxima vida:");
7       displaySoldier(maxLife);
8       bubbleSortLife(army1);
9       displayArmy(army1, "Ranking de soldados:");
10      makeGBoard();
11      displayBoard();
12  }
```

- El código pertenece al método principal
- Se inicializan dos ejércitos de soldados
- Se muestran por pantalla, gracias al método `displayArmy`, que mostrara sus niveles de vida, filas, columnas, nombres y puntos
- Se muestra el soldado de mayor vida, usando el método que hicimos dos laboratorios atrás
- Se muestra el ranking de soldados, usando el método que une a los dos ejércitos, los ordena y muestra el resultado por pantalla
- Se utiliza el algoritmo de ordenamiento bubble sort para ordenar los ejércitos
- y finalmente se muestra el tablero gracias al método `makeBoard` y `DisplayBoard` que crea, y muestra el tablero respectivamente

## 7. Ejecución

- A continuación mostraremos la ejecución de Videojuego.java en sus dos formas, la forma por consola, y la forma gráfica

## 7.1. Consola

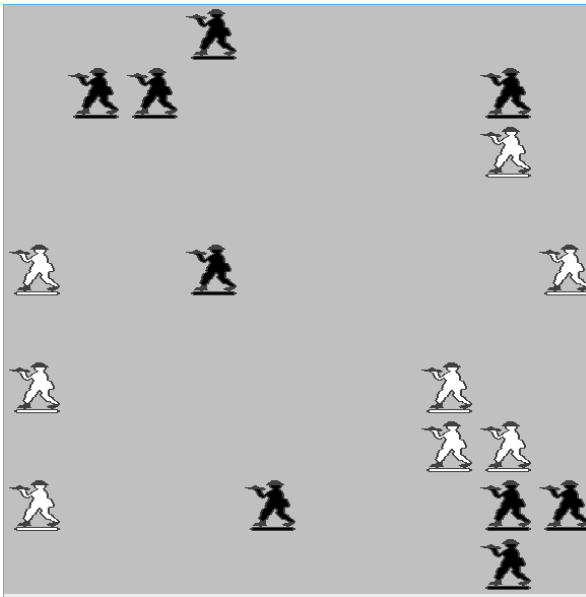
Listing 5: Ejecucion del codigo por consola

```
1
2 ===== Ejercito 1 =====
3 Soldado 0x0:
4   Nivel de vida: 3
5   Fila: 9
6   Columna: 1
7
8 Soldado 0x1:
9   Nivel de vida: 2
10  Fila: 8
11  Columna: 9
12
13 Soldado 0x2:
14  Nivel de vida: 5
15  Fila: 3
16  Columna: 9
17
18 Soldado 0x3:
19  Nivel de vida: 3
20  Fila: 5
21  Columna: 10
22
23 Soldado 0x4:
24  Nivel de vida: 1
25  Fila: 7
26  Columna: 1
27
28 Soldado 0x5:
29  Nivel de vida: 2
30  Fila: 7
31  Columna: 8
32
33 Soldado 0x6:
34  Nivel de vida: 4
35  Fila: 8
36  Columna: 8
37
38 Soldado 0x7:
39  Nivel de vida: 1
40  Fila: 5
41  Columna: 1
42
43
44 ===== Ejercito 2 =====
45 Soldado 1x0:
46  Nivel de vida: 5
47  Fila: 2
48  Columna: 3
49
50 Soldado 1x1:
51  Nivel de vida: 1
52  Fila: 10
53  Columna: 9
54
55 Soldado 1x2:
56  Nivel de vida: 2
57  Fila: 9
58  Columna: 5
59
60 Soldado 1x3:
61  Nivel de vida: 3
```

```
62  Fila: 2
63  Columna: 9
64
65  Soldado 1x4:
66  Nivel de vida: 4
67  Fila: 1
68  Columna: 4
69
70  Soldado 1x5:
71  Nivel de vida: 1
72  Fila: 2
73  Columna: 2
74
75  Soldado 1x6:
76  Nivel de vida: 4
77  Fila: 9
78  Columna: 9
79
80  Soldado 1x7:
81  Nivel de vida: 1
82  Fila: 5
83  Columna: 4
84
85  Soldado 1x8:
86  Nivel de vida: 3
87  Fila: 9
88  Columna: 10
89
90  Soldado con maxima vida:
91  Soldado 0x2:
92  Nivel de vida: 5
93  Fila: 3
94  Columna: 9
95
96
97  ===== Ranking de soldados: =====
98  Soldado 0x4:
99  Nivel de vida: 1
100  Fila: 7
101  Columna: 1
102
103  Soldado 0x1:
104  Nivel de vida: 2
105  Fila: 8
106  Columna: 9
107
108  Soldado 0x5:
109  Nivel de vida: 2
110  Fila: 7
111  Columna: 8
112
113  Soldado 0x7:
114  Nivel de vida: 1
115  Fila: 5
116  Columna: 1
117
118  Soldado 0x0:
119  Nivel de vida: 3
120  Fila: 9
121  Columna: 1
122
123  Soldado 0x3:
124  Nivel de vida: 3
125  Fila: 5
126  Columna: 10
```

```
127
128 Soldado 0x6:
129 Nivel de vida: 4
130 Fila: 8
131 Columna: 8
132
133 Soldado 0x2:
134 Nivel de vida: 5
135 Fila: 3
136 Columna: 9
```

- A continuación mostraremos la salida gráfica del código





## 8. Rúbricas

### 8.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
<b>2. Commits</b>	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
<b>3. Código fuente</b>	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
<b>4. Ejecución</b>	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
<b>5. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
<b>6. Fechas</b>	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
<b>7. Ortografía</b>	El documento no muestra errores ortográficos.	2	X	1.5	
<b>8. Madurez</b>	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	
<b>Total</b>		20		18.5	