

## Informe de Laboratorio 07

### Tema: Combinando Arreglos Estandar y ArrayList

Nota

Estudiante	Escuela	Asignatura
Christian Mestas Zegarra cmestasz@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
07	Combinando 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. Tarea

- **Actividad 1:** Cree un Proyecto llamado Laboratorio7
- **Actividad 2:** Usted deberá crear las dos clases Soldado.java y VideoJuego4.java. Puede reutilizar lo desarrollado en Laboratorios anteriores.
- **Actividad 3:** Del Soldado nos importa el nombre, puntos de vida, fila y columna (posición en el tablero).
- **Actividad 4:** El juego se desarrollará en el mismo tablero de los laboratorios anteriores. Para el tablero utilizar la estructura de datos más adecuada.
- **Actividad 5:** 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 Microsoft Windows 10 Pro 64 bits
- Visual Studio Code 1.82.2
- Java Development Kit 17.0.1
- Git 2.41.0.windows.1
- Windows PowerShell 5.1.19041.3031
- Cuenta en GitHub con el correo institucional.
- ArrayList de objetos.
- Arreglo bidimensional de objetos.
- Ordenamientos burbuja y por selección.

## 3. URL de Repositorio Github

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

## 4. Actividades con el repositorio GitHub


commits.bash

```
1 # CREACION DE CARPETAS Y ARCHIVOS COMO PLANTILLA
2 $ mkdir lab07
3 $ cd lab07
4 $ code Soldado.java
5 $ code VideoJuego4.java
6 $ git add fase02/lab07/Soldado.java
7 $ git add fase02/lab07/VideoJuego4.java
8 $ git commit -m "Plantilla en main"
9 [main edcc77a] Plantilla en main
10 2 files changed, 131 insertions(+)
11 create mode 100644 fase02/lab07/Soldado.java
12 create mode 100644 fase02/lab07/VideoJuego4.java
13 # METODO QUE CONTIENE LA BATALLA, PARA PERMITIR EL COMPORTAMIENTO ITERATIVO
14 $ git add fase02/lab07/VideoJuego4.java
15 $ git commit -m "Metodo simularBatalla()"
16 [main c13882a] Metodo simularBatalla()
17 1 file changed, 11 insertions(+)
18 # METODO QUE INICIALIZA LOS SOLDADOS
19 $ git add fase02/lab07/VideoJuego4.java
20 $ git commit -m "Metodo inicializarSoldados()"
21 [main 01bb7b2] Metodo inicializarSoldados()
22 1 file changed, 17 insertions(+), 1 deletion(-)
```

```
23 # METODO QUE IMPRIME EL TABLERO CON EL NOMBRE Y EL EQUIPO DE CADA SOLDADO
24 $ git add fase02/lab07/VideoJuego4.java
25 $ git commit -m "Metodo imprimirTablero() y auxiliares"
26 [main ad06cf5] Metodo imprimirTablero() y auxiliares
27 1 file changed, 30 insertions(+)
28 # METODO QUE RETORNA EL SOLDADO CON MAYOR VIDA DE UN EJERCITO
29 $ git add fase02/lab07/VideoJuego4.java
30 $ git commit -m "Metodo soldadoMayorVida()"
31 [main d9efb19] Metodo soldadoMayorVida()
32 1 file changed, 6 insertions(+)
33 # METODO QUE RETORNA EL PROMEDIO DE PUNTOS DE VIDA DE UN EJERCITO
34 $ git add fase02/lab07/VideoJuego4.java
35 $ git commit -m "Metodo promedioPuntosVida()"
36 [main a6511d1] Metodo promedioPuntosVida()
37 1 file changed, 2 insertions(+)
38 # METODO QUE IMPRIME LA LISTA DE SOLDADOS DE UN EJERCITO
39 $ git add fase02/lab07/VideoJuego4.java
40 $ git commit -m "Metodo imprimirSoldados()"
41 [main 22054ad] Metodo imprimirSoldados()
42 1 file changed, 2 insertions(+)
43 # METODO QUE COPIA LA LISTA DE SOLDADOS DE UN EJERCITO
44 $ git add fase02/lab07/VideoJuego4.java
45 $ git commit -m "Metodo copiarSoldados()"
46 [main 6d0fe11] Metodo copiarSoldados()
47 1 file changed, 2 insertions(+)
48 # METODO QUE ORDENA LOS SOLDADOS DE UN EJERCITO USANDO EL ORDENAMIENTO BURBUJA
49 $ git add fase02/lab07/VideoJuego4.java
50 $ git commit -m "Metodo ordenarSoldadosBurbuja() y auxiliar"
51 [main 6f49c2e] Metodo ordenarSoldadosBurbuja()
52 1 file changed, 8 insertions(+)
53 # METODO QUE ORDENA LOS SOLDADOS DE UN EJERCITO USANDO EL ORDENAMIENTO SELECCION
54 $ git add fase02/lab07/VideoJuego4.java
55 $ git commit -m "Metodo ordenarSoldadosSeleccion()"
56 [main dfdff96] Metodo ordenarSoldadosSeleccion()
57 1 file changed, 10 insertions(+)
58 # METODO QUE IMPRIME EL EJERCITO GANADOR DE LA BATALLA
59 $ git add fase02/lab07/VideoJuego4.java
60 $ git commit -m "Metodo imprimirGanador() y auxiliar"
61 [main 4e3a23c] Metodo imprimirGanador() y auxiliar
62 1 file changed, 12 insertions(+)
63 # PUSH FINAL
64 $ git push
65 Enumerating objects: 58, done.
66 Counting objects: 100% (58/58), done.
67 Delta compression using up to 4 threads
68 Compressing objects: 100% (56/56), done.
69 Writing objects: 100% (56/56), 6.07 KiB | 1.01 MiB/s, done.
70 Total 56 (delta 31), reused 0 (delta 0), pack-reused 0
71 remote: Resolving deltas: 100% (31/31), completed with 1 local object.
72 To https://github.com/cmestasz/fp2-23b.git
73 f98e335..7749073 main -> main
```

;

**Plantilla en main**  
main

 **cmestas** committed 27 minutes ago


1 parent f98e335 commit edcc77a

Showing 2 changed files with 131 additions and 0 deletions.

Split Unified

Primer Commit.

**Metodo simularBatalla()**  
main

 **cmestas** committed 20 minutes ago


1 parent edcc77a commit c13882a

Showing 1 changed file with 11 additions and 0 deletions.

Split Unified

Segundo Commit.

**Metodo inicializarSoldados()**  
main

 **cmestas** committed 19 minutes ago


1 parent c13882a commit 01bb7b2

Showing 1 changed file with 17 additions and 1 deletion.

Split Unified

Tercer Commit.

**Metodo imprimirTablero() y auxiliares**  
main

 **cmestas** committed 17 minutes ago


1 parent 01bb7b2 commit ad96cf5

Showing 1 changed file with 30 additions and 0 deletions.

Split Unified

Cuarto Commit.

**Metodo soldadoMayorVida()**  
main

 **cmestas** committed 17 minutes ago


1 parent ad96cf5 commit d9efb19

Showing 1 changed file with 6 additions and 0 deletions.

Split Unified

Quinto Commit.

**Metodo promedioPuntosVida()**  
main

 **cmestas** committed 16 minutes ago


1 parent d9efb19 commit a6511d1


Showing 1 changed file with 2 additions and 0 deletions.

Split Unified

Sexto Commit.

**Metodo imprimirSoldados()**

 main

 cmestasz committed 16 minutes ago

1 parent a6511d1 commit 22054ad


[Browse files](#)


Showing 1 changed file with 2 additions and 0 deletions.

[Split](#) [Unified](#)

Septimo Commit.

**Metodo copiarSoldados()**

 main

 cmestasz committed 15 minutes ago

1 parent 22054ad commit 6d0fe11


[Browse files](#)


Showing 1 changed file with 2 additions and 0 deletions.

[Split](#) [Unified](#)

Octavo Commit.

**Metodo ordenarSoldadosBurbuja() y auxiliar**

 main

 cmestasz committed 8 minutes ago

1 parent 6d0fe11 commit 33954af


[Browse files](#)


Showing 1 changed file with 11 additions and 0 deletions.

[Split](#) [Unified](#)

Noveno Commit.

**Metodo ordenarSoldadosSeleccion()**

 main

 cmestasz committed 8 minutes ago

1 parent 33954af commit 017e827


[Browse files](#)


Showing 1 changed file with 10 additions and 0 deletions.

[Split](#) [Unified](#)

Decimo Primer Commit.

**Metodo imprimirGanador() y auxiliar**

 main

 cmestasz committed 8 minutes ago

1 parent 017e827 commit 7749073

[Browse files](#)

Showing 1 changed file with 12 additions and 0 deletions.

[Split](#) [Unified](#)

Decimo Segundo Commit.

## 5. Código desarrollado

Soldado.java

```
1 package fase02.lab07;
2
3 public class Soldado {
4     private String nombre;
5     private int vida;
6     private int equipo;
7
8     public Soldado(String nombre, int vida, int equipo) {
9         setNombre(nombre);
10        setVida(vida);
11        setEquipo(equipo);
12    }
13
14    public void setNombre(String nombre) {
15        this.nombre = nombre;
16    }
17
18    public void setVida(int vida) {
19        this.vida = vida;
20    }
21
22    public void setEquipo(int equipo) {
23        this.equipo = equipo;
24    }
25
26    public String getNombre() {
27        return nombre;
28    }
29
30    public int getVida() {
31        return vida;
32    }
33
34    public int getEquipo() {
35        return equipo;
36    }
37
38    public String toString() {
39        return (nombre + " - " + vida);
40    }
41 }
```

- Clase que guarda nombre y vida del soldado.
- Posee tanto setters como getters para todos los atributos.
- Posee el metodo toString() para poder imprimir el objeto.

## VideoJuego4.java

```
1 package fase02.lab07;
2
3 import java.util.ArrayList;
4 import java.util.Random;
5 import java.util.Scanner;
6
7 public class VideoJuego4 {
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        String ans = "s";
11        while (ans.equalsIgnoreCase("s")) {
12            simularBatalla();
13            System.out.print("Desea simular otra batalla? (S/N): ");
14            ans = sc.nextLine();
15        }
16    }
17
18    public static void simularBatalla() {
19        Soldado[][] tablero = new Soldado[10][10];
20        ArrayList<Soldado> soldados1 = new ArrayList<Soldado>();
21        ArrayList<Soldado> soldados2 = new ArrayList<Soldado>();
22        inicializarSoldados(tablero, soldados1, 1);
23        inicializarSoldados(tablero, soldados2, 2);
24        imprimirTablero(tablero);
25        System.out.printf("Soldado con mayor vida del ejercito 1: %s%n",
26            soldadoMayorVida(soldados1));
27        System.out.printf("Promedio de puntos de vida del ejercito 1: %f%n",
28            promedioPuntosVida(soldados1));
29        System.out.println("\nSoldados por orden de creacion:");
30        imprimirSoldados(soldados1);
31        ArrayList<Soldado> soldados1a = new ArrayList<Soldado>();
32        ArrayList<Soldado> soldados1b = new ArrayList<Soldado>();
33        copiarArrayList(soldados1, soldados1a);
34        copiarArrayList(soldados1, soldados1b);
35        System.out.println("\nSoldados ordenados por burbuja");
36        ordenarSoldadosBurbuja(soldados1a);
37        imprimirSoldados(soldados1a);
38        System.out.println("\nSoldados ordenados por seleccion");
39        ordenarSoldadosSeleccion(soldados1b);
40        imprimirSoldados(soldados1b);
41        System.out.println();
42        System.out.printf("Soldado con mayor vida del ejercito 2: %s%n",
43            soldadoMayorVida(soldados2));
44        System.out.printf("Promedio de puntos de vida del ejercito 2: %f%n",
45            promedioPuntosVida(soldados2));
46        System.out.println("\nSoldados por orden de creacion:");
47        imprimirSoldados(soldados2);
48        ArrayList<Soldado> soldados2a = new ArrayList<Soldado>();
49        ArrayList<Soldado> soldados2b = new ArrayList<Soldado>();
50        copiarArrayList(soldados2, soldados2a);
51        copiarArrayList(soldados2, soldados2b);
52        System.out.println("\nSoldados ordenados por burbuja");
53        ordenarSoldadosBurbuja(soldados2a);
54        imprimirSoldados(soldados2a);
55        System.out.println("\nSoldados ordenados por seleccion");
```

```
52     ordenarSoldadosSeleccion(soldados2b);
53     imprimirSoldados(soldados2b);
54     System.out.println();
55     imprimirGanador(soldados1, soldados2);
56 }
57
58 public static void inicializarSoldados(Soldado[] [] tablero, ArrayList<Soldado> soldados,
59     int equipo) {
60     Random r = new Random();
61     int cantidad = r.nextInt(10) + 1;
62     for (int i = 0; i < cantidad; i++) {
63         String nombre = "Soldado" + i + "X" + equipo;
64         int vida = r.nextInt(5) + 1;
65         int fila, columna;
66         do {
67             fila = r.nextInt(10);
68             columna = r.nextInt(10);
69         } while (tablero[fila][columna] != null);
70         Soldado soldado = new Soldado(nombre, vida, equipo);
71         tablero[fila][columna] = soldado;
72         soldados.add(soldado);
73     }
74 }
75
76 public static void imprimirTablero(Soldado[] [] tablero) {
77     System.out.print(generarEncabezado(tablero));
78     String separacion = generarSeparacion(tablero);
79     for (int i = 0; i < tablero.length; i++) {
80         System.out.print(separacion);
81         System.out.print(generarFila(tablero, i));
82     }
83     System.out.print(separacion);
84 }
85
86 public static String generarEncabezado(Soldado[] [] tablero) {
87     String encabezado = "\t";
88     for (int i = 0; i < tablero.length; i++)
89         encabezado += (" " + i + " ");
90     encabezado += "\n";
91     return encabezado;
92 }
93
94 public static String generarSeparacion(Soldado[] [] tablero) {
95     String fila = "\t";
96     for (int i = 0; i < tablero[0].length; i++)
97         fila += "-----";
98     fila += "\n";
99     return fila;
100 }
101
102 public static String generarFila(Soldado[] [] tablero, int f) {
103     String fila = (f + 1) + "\t";
104     for (int i = 0; i < tablero[f].length; i++) {
105         fila += "| ";
106         Soldado soldado = tablero[f][i];
107         if (soldado != null)
```



```
108         fila += soldado.getNombre().substring(soldado.getNombre().length() - 3);
109     else
110         fila += " ";
111     fila += " ";
112 }
113 fila += "\n";
114 return fila;
115 }
116
117 public static Soldado soldadoMayorVida(ArrayList<Soldado> soldados) {
118     int idx = 0;
119     for (int i = 1; i < soldados.size(); i++) {
120         if (soldados.get(i).getVida() > soldados.get(idx).getVida())
121             idx = i;
122     }
123     return soldados.get(idx);
124 }
125
126 public static double promedioPuntosVida(ArrayList<Soldado> soldados) {
127     int suma = sumaPuntosVida(soldados);
128     return 1.0 * suma / soldados.size();
129 }
130
131 public static void imprimirSoldados(ArrayList<Soldado> soldados) {
132     for (Soldado soldado : soldados)
133         System.out.println(soldado);
134 }
135
136 public static void copiarArrayList(ArrayList<Soldado> original, ArrayList<Soldado> copia)
137 {
138     for (Soldado soldado : original)
139         copia.add(soldado);
140 }
141
142 public static void ordenarSoldadosBurbuja(ArrayList<Soldado> soldados) {
143     for (int i = 0; i < soldados.size() - 1; i++) {
144         for (int j = 0; j < soldados.size() - i - 1; j++) {
145             int vida1 = soldados.get(j).getVida();
146             int vida2 = soldados.get(j + 1).getVida();
147             if (vida1 < vida2)
148                 intercambiar(soldados, j, j + 1);
149         }
150     }
151 }
152
153 public static void ordenarSoldadosSeleccion(ArrayList<Soldado> soldados) {
154     for (int i = 0; i < soldados.size() - 1; i++) {
155         int idx = i;
156         for (int j = i + 1; j < soldados.size(); j++) {
157             int vida1 = soldados.get(j).getVida();
158             int vida2 = soldados.get(idx).getVida();
159             if (vida1 > vida2)
160                 idx = j;
161         }
162         intercambiar(soldados, i, idx);
163     }
164 }
```

```
163     }
164
165     public static void imprimirGanador(ArrayList<Soldado> soldados1, ArrayList<Soldado>
        soldados2) {
166         int suma1 = sumaPuntosVida(soldados1);
167         int suma2 = sumaPuntosVida(soldados2);
168         if (suma1 == suma2)
169             System.out.printf("Hay un empate con %d puntos de vida!\n", suma1);
170         else if (suma1 > suma2)
171             System.out.printf("Gana el ejercito 1 con %d a %d puntos de vida!\n", suma1,
                suma2);
172         else
173             System.out.printf("Gana el ejercito 2 con %d a %d puntos de vida!\n", suma2,
                suma1);
174     }
175
176     public static int sumaPuntosVida(ArrayList<Soldado> soldados) {
177         int suma = 0;
178         for (int i = 0; i < soldados.size(); i++)
179             suma += soldados.get(i).getVida();
180         return suma;
181     }
182
183     public static void intercambiar(ArrayList<Soldado> soldados, int i, int j) {
184         Soldado t = soldados.get(i);
185         soldados.set(i, soldados.get(j));
186         soldados.set(j, t);
187     }
188
189     public static char intToChar(int n) {
190         return (char) (n + 'A' - 1);
191     }
192 }
```

- Método `simularBatalla()` contiene la batalla, para permitir el comportamiento iterativo.
- Método `inicializarSoldados()` crea a los soldados, los ubica en el tablero y los guarda en un arreglo de soldados por separado.
- Método `imprimirTablero()` imprime el tablero con ayuda de los métodos auxiliares `generarEncabezado()`, `generarSeparacion()` y `generarFila()`, ubicando a los soldados por su número y ejército.
- Método `soldadoMayorVida()` retorna el soldado con mayor vida de un ejército.
- Método `promedioPuntosVida()` retorna el promedio de los puntos de vida de todos los soldados de un ejército.
- Método `imprimirSoldados()` imprime los soldados del ejército.
- Se crean dos copias del arreglo de soldados usando el método `copiarSoldados()` para demostrar los 2 ordenamientos.
- Método `ordenarSoldadosBurbuja()` ordena los soldados por vida de mayor a menor, usando ordenamiento burbuja.
- Método `ordenarSoldadosSeleccion()` ordena los soldados por vida de mayor a menor, usando ordenamiento selección.

- Se reusan los métodos para mostrar los datos de ambos ejércitos.
- Método imprimirGanador() imprime el ejército ganador de acuerdo a la cantidad total de puntos de vida de cada uno.

## 6. Ejecución del código

VideoJuego3.java

```

1      A    B    C    D    E    F    G    H    I    J
2      -----
3  1      |    |    |    |    |    |    |    |    |
4      -----
5  2      |    | 8X1 |    |    |    |    |    |    |
6      -----
7  3      | 3X1 | 6X1 |    |    |    |    |    |    |
8      -----
9  4      | 1X1 |    |    |    |    |    |    | 7X1 |
10     -----
11 5      |    |    |    |    |    |    |    | 2X1 |
12     -----
13 6      |    |    |    |    |    |    |    | 5X1 |
14     -----
15 7      | 4X1 |    |    |    |    |    |    |    |
16     -----
17 8      |    |    |    |    |    |    |    |    |
18     -----
19 9      | 0X1 |    |    |    | 0X2 |    |    |    |
20     -----
21 10     |    |    |    |    |    |    |    |    |
22     -----
23 Soldado con mayor vida del ejercito 1: Soldado1X1 - 5
24 Promedio de puntos de vida del ejercito 1: 2.888889
25
26 Soldados por orden de creacion:
27 Soldado0X1 - 3
28 Soldado1X1 - 5
29 Soldado2X1 - 2
30 Soldado3X1 - 2
31 Soldado4X1 - 3
32 Soldado5X1 - 5
33 Soldado6X1 - 2
34 Soldado7X1 - 3
35 Soldado8X1 - 1
36
37 Soldados ordenados por burbuja
38 Soldado1X1 - 5
39 Soldado5X1 - 5
40 Soldado0X1 - 3
41 Soldado4X1 - 3
42 Soldado7X1 - 3
43 Soldado2X1 - 2
44 Soldado3X1 - 2
45 Soldado6X1 - 2
46 Soldado8X1 - 1
47
48 Soldados ordenados por seleccion
49 Soldado1X1 - 5
50 Soldado5X1 - 5
51 Soldado4X1 - 3
52 Soldado0X1 - 3

```

```

53 Soldado7X1 - 3
54 Soldado3X1 - 2
55 Soldado6X1 - 2
56 Soldado2X1 - 2
57 Soldado8X1 - 1
58
59 Soldado con mayor vida del ejercito 2: Soldado0X2 - 3
60 Promedio de puntos de vida del ejercito 2: 3.000000
61
62 Soldados por orden de creacion:
63 Soldado0X2 - 3
64
65 Soldados ordenados por burbuja
66 Soldado0X2 - 3
67
68 Soldados ordenados por seleccion
69 Soldado0X2 - 3
70
71 Gana el ejercito 1 con 26 a 3 puntos de vida!
72 Desea simular otra batalla? (S/N): S
73      A      B      C      D      E      F      G      H      I      J
74 -----
75 1      |      |      |      | 0X1 |      |      |      |      |      | 1X2 |
76 -----
77 2      |      |      |      |      |      |      |      |      |      |      |
78 -----
79 3      |      |      |      |      |      |      |      |      |      |      |
80 -----
81 4      |      |      |      |      |      |      | 3X2 |      |      |      |
82 -----
83 5      |      |      | 2X2 |      |      |      |      |      |      | 1X1 |
84 -----
85 6      |      |      |      |      | 5X2 |      | 4X2 |      |      |      |
86 -----
87 7      |      |      |      | 2X1 |      | 3X1 |      |      |      |      |
88 -----
89 8      |      | 0X2 |      |      |      |      | 4X1 |      |      |      |
90 -----
91 9      |      |      |      |      |      |      |      |      |      |      |
92 -----
93 10     |      |      |      |      |      | 6X2 |      |      |      |      |
94 -----
95 Soldado con mayor vida del ejercito 1: Soldado0X1 - 5
96 Promedio de puntos de vida del ejercito 1: 3.800000
97
98 Soldados por orden de creacion:
99 Soldado0X1 - 5
100 Soldado1X1 - 5
101 Soldado2X1 - 3
102 Soldado3X1 - 3
103 Soldado4X1 - 3
104
105 Soldados ordenados por burbuja
106 Soldado0X1 - 5
107 Soldado1X1 - 5
108 Soldado2X1 - 3

```

```
109 Soldado3X1 - 3
110 Soldado4X1 - 3
111
112 Soldados ordenados por seleccion
113 Soldado0X1 - 5
114 Soldado1X1 - 5
115 Soldado2X1 - 3
116 Soldado3X1 - 3
117 Soldado4X1 - 3
118
119 Soldado con mayor vida del ejercito 2: Soldado4X2 - 5
120 Promedio de puntos de vida del ejercito 2: 3.714286
121
122 Soldados por orden de creacion:
123 Soldado0X2 - 4
124 Soldado1X2 - 4
125 Soldado2X2 - 3
126 Soldado3X2 - 4
127 Soldado4X2 - 5
128 Soldado5X2 - 2
129 Soldado6X2 - 4
130
131 Soldados ordenados por burbuja
132 Soldado4X2 - 5
133 Soldado0X2 - 4
134 Soldado1X2 - 4
135 Soldado3X2 - 4
136 Soldado6X2 - 4
137 Soldado2X2 - 3
138 Soldado5X2 - 2
139
140 Soldados ordenados por seleccion
141 Soldado4X2 - 5
142 Soldado1X2 - 4
143 Soldado3X2 - 4
144 Soldado0X2 - 4
145 Soldado6X2 - 4
146 Soldado2X2 - 3
147 Soldado5X2 - 2
148
149 Gana el ejercito 2 con 26 a 19 puntos de vida!
150 Desea simular otra batalla? (S/N): N
```

## 7. Estructura de laboratorio 07

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab07/  
|--- Soldado.java  
|--- VideoJuego4.java  
|--- commits.bash  
|--- ejec01.bash  
|--- Informe.tex  
|--- Informe.pdf  
|--- img  
|   |--- logo_abet.png  
|   |--- logo_episunsa.png  
|   |--- logo_unsa.jpg  
|   |--- commit01.jpg  
|   |--- commit02.jpg  
|   |--- commit03.jpg  
|   |--- commit04.jpg  
|   |--- commit05.jpg  
|   |--- commit06.jpg  
|   |--- commit07.jpg  
|   |--- commit08.jpg  
|   |--- commit09.jpg  
|   |--- commit10.jpg  
|   |--- commit11.jpg
```

## 8. Rúbricas

### 8.1. Entregable Informe

Tipo de Informe

<b>Informe</b>	
<b>Latex</b>	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

### 8.2. 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.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobatoria, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Niveles de desempeño

<b>Puntos</b>	<b>Nivel</b>			
	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



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	1.5	
<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 estan 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	

## 9. Referencias

- Aedo, M. y Castro, E. (2021). FUNDAMENTOS DE PROGRAMACIÓN 2 - Tópicos de Programación Orientada a Objetos. Editorial UNSA.