

Informe de Laboratorio 08

Tema: HashMap

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
08	HashMap	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 Laboratorio8
- **Actividad 2:** Usted deberá crear las dos clases Soldado.java y VideoJuego5.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 (usar HashMaps). 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 (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 (indicar conclusiones respecto a este ordenamiento de HashMaps). Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla). Hacerlo como 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.
- HashMap 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 08 en el Repositorio GitHub.
- <https://github.com/cmestasz/fp2-23b/tree/main/fase02/lab08>


4. Actividades con el repositorio GitHub


commits.bash

```
1 # CREACION DE CARPETAS Y ARCHIVOS COMO PLANTILLA
2 $ mkdir lab07
3 $ code lab08/Soldado.java
4 $ code lab08/VideoJuego5.java
5 $ git add fase02/lab08/Soldado.java
6 $ git add fase02/lab08/VideoJuego5.java
7 $ git commit -m "Plantilla en main y metodo simularBatalla()"
8 [main cf15a45] Plantilla en main y metodo simularBatalla()
9 2 files changed, 145 insertions(+)
10 create mode 100644 fase02/lab08/Soldado.java
11 create mode 100644 fase02/lab08/VideoJuego5.java
12 $ git add fase02/lab08/VideoJuego5.java
13 # METODO QUE INICIALIZA LOS SOLDADOS
14 $ git commit -m "Metodo inicializarSoldados()"
15 [main a89a2a0] Metodo inicializarSoldados()
16 1 file changed, 16 insertions(+), 1 deletion(-)
17 $ git add fase02/lab08/VideoJuego5.java
18 # METODO QUE IMPRIME EL TABLERO CON EL NOMBRE Y EL EQUIPO DE CADA SOLDADO
19 $ git commit -m "Metodo imprimirTablero() y auxiliares"
20 [main 97d9c09] Metodo imprimirTablero() y auxiliares
21 1 file changed, 35 insertions(+)
22 $ git add fase02/lab08/VideoJuego5.java
```


```
23 # METODO QUE RETORNA EL SOLDADO CON MAYOR VIDA DE UN EJERCITO
24 $ git commit -m "Metodo soldadoMayorVida()"
25 [main 84b1c6b] Metodo soldadoMayorVida()
26 1 file changed, 7 insertions(+)
27 $ git add fase02/lab08/VideoJuego5.java
28 # METODO QUE RETORNA EL PROMEDIO DE PUNTOS DE VIDA DE UN EJERCITO
29 $ git commit -m "Metodo promedioPuntosVida() y auxiliar"
30 [main ba62eee] Metodo promedioPuntosVida() y auxiliar
31 1 file changed, 6 insertions(+)
32 $ git add fase02/lab08/VideoJuego5.java
33 # METODO QUE IMPRIME LA LISTA DE SOLDADOS DE UN EJERCITO
34 $ git commit -m "Metodo imprimirHashMap()"
35 [main fe60a09] Metodo imprimirHashMap()
36 1 file changed, 2 insertions(+)
37 $ git add fase02/lab08/VideoJuego5.java
38 # METODO QUE COPIA LA LISTA DE SOLDADOS DE UN EJERCITO
39 $ git commit -m "Metodo imprimirArrayList()"
40 [main b32f9d2] Metodo imprimirArrayList()
41 1 file changed, 2 insertions(+)
42 $ git add fase02/lab08/VideoJuego5.java
43 # METODO QUE COPIA EL MAPA DE SOLDADOS DE UN EJERCITO
44 $ git commit -m "Metodo copiarHashMap()"
45 [main 614a758] Metodo copiarHashMap()
46 1 file changed, 2 insertions(+)
47 $ git add fase02/lab08/VideoJuego5.java
48 # METODO QUE ORDENA LOS SOLDADOS DE UN EJERCITO USANDO EL ORDENAMIENTO BURBUJA
49 $ git commit -m "Metodo ordenarSoldadosBurbuja() y auxiliar"
50 [main cea0f61] Metodo ordenarSoldadosBurbuja() y auxiliar
51 1 file changed, 15 insertions(+)
52 $ git add fase02/lab08/VideoJuego5.java
53 # METODO QUE ORDENA LOS SOLDADOS DE UN EJERCITO USANDO EL ORDENAMIENTO SELECCION
54 $ git commit -m "Metodo ordenarSoldadosSeleccion()"
55 [main dab2537] Metodo ordenarSoldadosSeleccion()
56 1 file changed, 14 insertions(+)
57 $ git add fase02/lab08/VideoJuego5.java
58 # METODO QUE IMPRIME EL EJERCITO GANADOR DE LA BATALLA
59 $ git commit -m "Metodo imprimirGanador()"
60 [main 92cb4d3] Metodo imprimirGanador()
61 1 file changed, 8 insertions(+)
62 # PUSH FINAL
63 $ git push
64 Enumerating objects: 58, done.
65 Counting objects: 100% (58/58), done.
66 Delta compression using up to 4 threads
67 Compressing objects: 100% (56/56), done.
68 Writing objects: 100% (56/56), 6.21 KiB | 1.24 MiB/s, done.
69 Total 56 (delta 32), reused 0 (delta 0), pack-reused 0
70 remote: Resolving deltas: 100% (32/32), completed with 2 local objects.
71 To https://github.com/cmestasz/fp2-23b.git
72 86677b2..92cb4d3 main -> main
```

Plantilla en main y metodo simularBatalla()

 main

 cmestas committed 16 minutes ago


1 parent 86677b2 commit cf15a45


 Showing 2 changed files with 145 additions and 0 deletions.

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

Primer Commit.

Metodo inicializarSoldados()

 main

 cmestas committed 16 minutes ago


1 parent cf15a45 commit a89a2a0


Showing 1 changed file with 16 additions and 1 deletion.

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

Segundo Commit.

Metodo imprimirTablero() y auxiliares

 main

 cmestas committed 14 minutes ago


1 parent a89a2a0 commit 97d9c09


Showing 1 changed file with 35 additions and 0 deletions.

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

Tercer Commit.

Metodo soldadoMayorVida()

 main

 cmestas committed 14 minutes ago


1 parent 97d9c09 commit 84b1c6b


Showing 1 changed file with 7 additions and 0 deletions.

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

Cuarto Commit.

Metodo promedioPuntosVida() y auxiliar

 main

 cmestas committed 13 minutes ago


1 parent 84b1c6b commit ba62eee


Showing 1 changed file with 6 additions and 0 deletions.

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

Quinto Commit.

Metodo imprimirHashMap()

 main

 cmestas committed 13 minutes ago


1 parent ba62eee commit fe60a09


Showing 1 changed file with 2 additions and 0 deletions.

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

Sexto Commit.

Metodo imprimirArrayList()

 main

 cmestasz committed 13 minutes ago


1 parent fe60e09 commit b32f9d2


[Browse files](#)

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

Septimo Commit.

Metodo copiarHashMap()

 main

 cmestasz committed 13 minutes ago


1 parent b32f9d2 commit 614a758


[Browse files](#)

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

Octavo Commit.

Metodo ordenarSoldadosBurbuja() y auxiliar

 main

 cmestasz committed 12 minutes ago


1 parent 614a758 commit cea0f61


[Browse files](#)

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

Noveno Commit.

Metodo ordenarSoldadosSeleccion()

 main

 cmestasz committed 12 minutes ago


1 parent cea0f61 commit dab2537


[Browse files](#)

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

Decimo Commit.

Metodo imprimirGanador()

 main

 cmestasz committed 12 minutes ago

1 parent dab2537 commit 92cb4d3

[Browse files](#)

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

Decimo Primer Commit.

5. Código desarrollado

Soldado.java

```
1 package fase02.lab08;
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.

VideoJuego5.java

```
1 package fase02.lab08;
2
3 import java.util.Scanner;
4 import java.util.ArrayList;
5 import java.util.HashMap;
6 import java.util.Map.Entry;
7 import java.util.Random;
8
9 public class VideoJuego5 {
10     public static void main(String[] args) {
11         Scanner sc = new Scanner(System.in);
12         String ans = "s";
13         while (ans.equalsIgnoreCase("s")) {
14             simularBatalla();
15             System.out.print("Desea simular otra batalla? (S/N): ");
16             ans = sc.nextLine();
17         }
18     }
19
20     public static void simularBatalla() {
21         HashMap<String, Soldado> soldados1 = new HashMap<String, Soldado>();
22         HashMap<String, Soldado> soldados2 = new HashMap<String, Soldado>();
23         inicializarSoldados(soldados1, 1);
24         inicializarSoldados(soldados2, 2);
25         imprimirTablero(soldados1, soldados2);
26         System.out.printf("Soldado con mayor vida del ejercito 1: %s%n",
27             soldadoMayorVida(soldados1));
28         System.out.printf("Promedio de puntos de vida del ejercito 1: %f%n",
29             promedioPuntosVida(soldados1));
30         System.out.println("\nSoldados:");
31         imprimirHashMap(soldados1);
32         HashMap<String, Soldado> soldados1a = new HashMap<String, Soldado>();
33         HashMap<String, Soldado> soldados1b = new HashMap<String, Soldado>();
34         copiarHashMap(soldados1, soldados1a);
35         copiarHashMap(soldados1, soldados1b);
36         System.out.println("\nSoldados ordenados por burbuja");
37         ArrayList<Soldado> soldados1a0 = ordenarSoldadosBurbuja(soldados1a);
38         imprimirArrayList(soldados1a0);
39         System.out.println("\nSoldados ordenados por seleccion");
40         ArrayList<Soldado> soldados1b0 = ordenarSoldadosSeleccion(soldados1b);
41         imprimirArrayList(soldados1b0);
42         System.out.println();
43         System.out.printf("Soldado con mayor vida del ejercito 2: %s%n",
44             soldadoMayorVida(soldados2));
45         System.out.printf("Promedio de puntos de vida del ejercito 2: %f%n",
46             promedioPuntosVida(soldados2));
47         System.out.println("\nSoldados:");
48         imprimirHashMap(soldados2);
49         HashMap<String, Soldado> soldados2a = new HashMap<String, Soldado>();
50         HashMap<String, Soldado> soldados2b = new HashMap<String, Soldado>();
51         copiarHashMap(soldados2, soldados2a);
52         copiarHashMap(soldados2, soldados2b);
53         System.out.println("\nSoldados ordenados por burbuja");
54         ArrayList<Soldado> soldados2a0 = ordenarSoldadosBurbuja(soldados2a);
55         imprimirArrayList(soldados2a0);
```

```
52     System.out.println("\nSoldados ordenados por seleccion");
53     ArrayList<Soldado> soldados2b0 = ordenarSoldadosSeleccion(soldados2b);
54     imprimirArrayList(soldados2b0);
55     System.out.println();
56     imprimirGanador(soldados1, soldados2);
57 }
58
59 public static void inicializarSoldados(HashMap<String, Soldado> soldados,
60     int equipo) {
61     Random r = new Random();
62     int cantidad = r.nextInt(10) + 1;
63     for (int i = 0; i < cantidad; i++) {
64         String nombre = "Soldado" + i + "X" + equipo;
65         int vida = r.nextInt(5) + 1;
66         int fila, columna;
67         do {
68             fila = r.nextInt(10);
69             columna = r.nextInt(10);
70         } while (soldados.containsKey(fila + "," + columna));
71         Soldado soldado = new Soldado(nombre, vida, equipo);
72         soldados.put(fila + "," + columna, soldado);
73     }
74 }
75
76 public static void imprimirTablero(HashMap<String, Soldado> soldados1, HashMap<String,
77     Soldado> soldados2) {
78     System.out.print(generarEncabezado());
79     String separacion = generarSeparacion();
80     for (int i = 0; i < 10; i++) {
81         System.out.print(separacion);
82         System.out.print(generarFila(soldados1, soldados2, i));
83     }
84     System.out.print(separacion);
85 }
86
87 public static String generarEncabezado() {
88     String encabezado = "\t";
89     for (int i = 0; i < 10; i++)
90         encabezado += (" " + intToChar(i + 1) + " ");
91     encabezado += " \n";
92     return encabezado;
93 }
94
95 public static String generarSeparacion() {
96     String fila = "\t";
97     for (int i = 0; i < 10; i++)
98         fila += "-----";
99     fila += "-\n";
100     return fila;
101 }
102
103 public static String generarFila(HashMap<String, Soldado> soldados1, HashMap<String,
104     Soldado> soldados2, int f) {
105     String fila = (f + 1) + "\t";
106     for (int i = 0; i < 10; i++) {
107         fila += "| ";
```



```
106         Soldado soldado = null;
107         String llave = f + "," + i;
108         if (soldados1.containsKey(llave))
109             soldado = soldados1.get(llave);
110         else if (soldados2.containsKey(llave))
111             soldado = soldados2.get(llave);
112         else
113             fila += " ";
114         if (soldado != null)
115             fila += soldado.getNombre().substring(soldado.getNombre().length() - 3);
116         fila += " ";
117     }
118     fila += "\n";
119     return fila;
120 }
121
122 public static Soldado soldadoMayorVida(HashMap<String, Soldado> soldados) {
123     String llave = null;
124     for (Entry<String, Soldado> entrySet : soldados.entrySet()) {
125         if (llave == null || entrySet.getValue().getVida() > soldados.get(llave).getVida())
126             llave = entrySet.getKey();
127     }
128     return soldados.get(llave);
129 }
130
131 public static double promedioPuntosVida(HashMap<String, Soldado> soldados) {
132     int suma = sumaPuntosVida(soldados);
133     return 1.0 * suma / soldados.size();
134 }
135
136 public static void imprimirHashMap(HashMap<String, Soldado> soldados) {
137     for (Entry<String, Soldado> entrySet : soldados.entrySet())
138         System.out.println(entrySet.getValue());
139 }
140
141 public static void imprimirArrayList(ArrayList<Soldado> soldados) {
142     for (Soldado soldado : soldados)
143         System.out.println(soldado);
144 }
145
146 public static void copiarHashMap(HashMap<String, Soldado> original, HashMap<String,
147     Soldado> copia) {
148     for (Entry<String, Soldado> entrySet : original.entrySet())
149         copia.put(entrySet.getKey(), entrySet.getValue());
150 }
151
152 public static ArrayList<Soldado> ordenarSoldadosBurbuja(HashMap<String, Soldado>
153     soldados) {
154     ArrayList<Soldado> arreglo = new ArrayList<Soldado>();
155     for (Entry<String, Soldado> entrySet : soldados.entrySet())
156         arreglo.add(entrySet.getValue());
157     for (int i = 0; i < arreglo.size() - 1; i++) {
158         for (int j = 0; j < arreglo.size() - i - 1; j++) {
159             int vida1 = arreglo.get(j).getVida();
160             int vida2 = arreglo.get(j + 1).getVida();
161             if (vida1 < vida2)
```

```
160         intercambiar(arreglo, j, j + 1);
161     }
162 }
163 return arreglo;
164 }
165
166 public static ArrayList<Soldado> ordenarSoldadosSeleccion(HashMap<String, Soldado>
    soldados) {
167     ArrayList<Soldado> arreglo = new ArrayList<Soldado>();
168     for (Entry<String, Soldado> entrySet : soldados.entrySet())
169         arreglo.add(entrySet.getValue());
170     for (int i = 0; i < arreglo.size() - 1; i++) {
171         int idx = i;
172         for (int j = i + 1; j < arreglo.size(); j++) {
173             int vida1 = arreglo.get(j).getVida();
174             int vida2 = arreglo.get(idx).getVida();
175             if (vida1 > vida2)
176                 idx = j;
177         }
178         intercambiar(arreglo, i, idx);
179     }
180     return arreglo;
181 }
182
183 public static void imprimirGanador(HashMap<String, Soldado> soldados1, HashMap<String,
    Soldado> soldados2) {
184     int suma1 = sumaPuntosVida(soldados1);
185     int suma2 = sumaPuntosVida(soldados2);
186     if (suma1 == suma2)
187         System.out.printf("Hay un empate con %d puntos de vida!\n", suma1);
188     else if (suma1 > suma2)
189         System.out.printf("Gana el ejercito 1 con %d a %d puntos de vida!\n", suma1,
            suma2);
190     else
191         System.out.printf("Gana el ejercito 2 con %d a %d puntos de vida!\n", suma2,
            suma1);
192 }
193
194 public static int sumaPuntosVida(HashMap<String, Soldado> soldados) {
195     int suma = 0;
196     for (Entry<String, Soldado> entrySet : soldados.entrySet())
197         suma += entrySet.getValue().getVida();
198     return suma;
199 }
200
201 public static void intercambiar(ArrayList<Soldado> soldados, int i, int j) {
202     Soldado t = soldados.get(i);
203     soldados.set(i, soldados.get(j));
204     soldados.set(j, t);
205 }
206
207 public static char intToChar(int n) {
208     return (char) (n + 'A' - 1);
209 }
210 }
```

- 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 hashmaps 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 numero 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.
- Métodos `imprimirHashMap()` e `imprimirArrayList()` imprimen los soldados del ejército.
- Se crean dos copias del arreglo de soldados usando el método `copiarHashMap()` 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      | 3X1 |      |      |      |      |      |      | 2X1 |      |
4      -----
5  2      |      |      |      |      |      |      |      | 4X1 |      |
6      -----
7  3      |      | 5X1 |      |      |      |      |      | 0X1 |      |
8      -----
9  4      |      |      | 1X1 |      |      |      | 0X2 | 6X1 | 7X1 |      |
10     -----
11 5      | 9X1 |      |      |      |      |      |      |      |      |
12     -----
13 6      |      |      |      |      |      |      |      |      |      |
14     -----
15 7      |      |      |      |      |      |      |      |      |      |
16     -----
17 8      | 1X2 |      |      |      |      |      |      |      |      |
18     -----
19 9      |      |      | 8X1 |      |      |      |      |      |      |
20     -----
21 10     |      |      |      |      |      |      |      |      |      |
22     -----
23 Soldado con mayor vida del ejercito 1: Soldado1X1 - 5
24 Promedio de puntos de vida del ejercito 1: 2.800000
25
26 Soldados:
27 Soldado3X1 - 2
28 Soldado5X1 - 3
29 Soldado9X1 - 4
30 Soldado1X1 - 5
31 Soldado2X1 - 1
32 Soldado4X1 - 4
33 Soldado0X1 - 2
34 Soldado6X1 - 2
35 Soldado8X1 - 2
36 Soldado7X1 - 3
37
38 Soldados ordenados por burbuja
39 Soldado1X1 - 5
40 Soldado9X1 - 4
41 Soldado4X1 - 4
42 Soldado5X1 - 3
43 Soldado7X1 - 3
44 Soldado3X1 - 2
45 Soldado0X1 - 2
46 Soldado6X1 - 2
47 Soldado8X1 - 2
48 Soldado2X1 - 1
49
50 Soldados ordenados por seleccion
51 Soldado1X1 - 5
52 Soldado9X1 - 4

```

```

53 Soldado4X1 - 4
54 Soldado5X1 - 3
55 Soldado7X1 - 3
56 Soldado3X1 - 2
57 Soldado0X1 - 2
58 Soldado6X1 - 2
59 Soldado8X1 - 2
60 Soldado2X1 - 1
61
62 Soldado con mayor vida del ejercito 2: Soldado1X2 - 4
63 Promedio de puntos de vida del ejercito 2: 3.000000
64
65 Soldados:
66 Soldado1X2 - 4
67 Soldado0X2 - 2
68
69 Soldados ordenados por burbuja
70 Soldado1X2 - 4
71 Soldado0X2 - 2
72
73 Soldados ordenados por seleccion
74 Soldado1X2 - 4
75 Soldado0X2 - 2
76
77 Gana el ejercito 1 con 28 a 6 puntos de vida!
78 Desea simular otra batalla? (S/N): S
79
80      A      B      C      D      E      F      G      H      I      J
81 1  |  |  | 0X2 |  |  |  | 0X1 |  |  |  |  |
82 -----
83 2  |  |  |  |  |  |  |  |  |  |  |  |
84 -----
85 3  |  |  |  |  |  |  |  |  |  |  |  |
86 -----
87 4  |  |  |  |  |  |  |  |  |  |  |  |
88 -----
89 5  |  |  |  |  |  |  |  |  |  |  |  |
90 -----
91 6  |  |  |  |  |  |  |  |  |  |  |  |
92 -----
93 7  |  |  |  |  |  |  |  |  |  |  |  |
94 -----
95 8  |  | 1X1 |  |  |  |  |  |  |  |  |
96 -----
97 9  |  |  |  |  |  |  |  |  |  |  |  |
98 -----
99 10 |  |  |  |  |  |  |  |  |  |  |  |
100 -----
101 Soldado con mayor vida del ejercito 1: Soldado1X1 - 4
102 Promedio de puntos de vida del ejercito 1: 3.000000
103
104 Soldados:
105 Soldado0X1 - 2
106 Soldado1X1 - 4
107
108 Soldados ordenados por burbuja

```

```
109 Soldado1X1 - 4
110 Soldado0X1 - 2
111
112 Soldados ordenados por seleccion
113 Soldado1X1 - 4
114 Soldado0X1 - 2
115
116 Soldado con mayor vida del ejercito 2: Soldado0X2 - 2
117 Promedio de puntos de vida del ejercito 2: 2.000000
118
119 Soldados:
120 Soldado0X2 - 2
121
122 Soldados ordenados por burbuja
123 Soldado0X2 - 2
124
125 Soldados ordenados por seleccion
126 Soldado0X2 - 2
127
128 Gana el ejercito 1 con 6 a 2 puntos de vida!
129 Desea simular otra batalla? (S/N): N
```

7. Estructura de laboratorio 08

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

```
lab08/  
|--- Soldado.java  
|--- VideoJuego5.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

Informe	
Latex	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

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

Rúbrica para contenido del Informe y demostración

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	3	
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	1.5	
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 estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
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	4	
Total		20		18	

9. Referencias

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