

Informe de Laboratorio 06

Tema: ArrayList

Nota

Estudiante	Escuela	Asignatura
Hernan Andy Choquehuanca Zapana hchoquehuanca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación II Semestre: II Código: 20232191

Laboratorio	Tema	Duración
06	ArrayList	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 11 Octubre 2023	Al 16 Octubre 2023

1. Tarea

- Cree un Proyecto llamado Laboratorio6
- Usted deberá crear las dos clases Soldado.java y VideoJuego3.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. Pero ahora el tablero debe ser un ArrayList bidimensional.
- Tendrá 2 Ejércitos. 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. Finalmente, que muestre qué ejército ganará la batalla (indicar la métrica usada para decidir al ganador de la batalla).

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- VIM 9.0.
- Visual Studio Code.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- Variables Simples
- Métodos.
- Métodos de Búsqueda y Ordenamiento.
- ArrayList

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/hernanchoquehuanca/fp2-23b.git>
- URL para el laboratorio 04 en el Repositorio GitHub.
- <https://github.com/hernanchoquehuanca/fp2-23b/tree/main/fase02/lab06>

4. Trabajo del Laboratorio 06

4.1. Actividad 01

4.1.1. Clase Soldado.java

- Haciendo uso de la clase Soldado.java del laboratorio anterior (lab05), se adaptó al ejercicio actual.
 - Primero copiamos el código tal cual a la carpeta del laboratorio actual (lab06).
- Nuestra clase Soldado tendrá los siguientes atributos:
 - name (Nombre).
 - row (Fila).
 - column (Columna).
 - status (Estado).
 - health (Vida).

```
2 private String name;  
3 private int row;  
4 private char column;  
5 private boolean status;  
6 private int health;
```

- Además creamos el método constructor, teniendo en cuenta que tiene que tener el mismo nombre que la clase y considerando sus atributos como parámetros:
 - Se utilizó el puntero `this` para evitar ambigüedades.

```
8 public Soldado(String name, int row, char column, boolean status, int health) {
9     this.name = name;
10    this.row = row;
11    this.column = column;
12    this.status = status;
13    this.health = health;
14 }
```

- Luego de ello también se implementaron los getters y setters para cada atributo de nuestra clase Soldado.

- Setters:

```
16 public void setName(String n){
17     name = n;
18 }
19 public void setRow(int r){
20     row = r;
21 }
22 public void setColumn(char c){
23     column = c;
24 }
25 public void setStatus(boolean s){
26     status = s;
27 }
28 public void setHealth(int h){
29     health = h;
30 }
```

- Getters:

```
32 public String getName(){
33     return name;
34 }
35 public int getRow(){
36     return row;
37 }
38 public char getColumn(){
39     return column;
40 }
41 public boolean getStatus(){
42     return status;
43 }
44 public int getHealth(){
45     return health;
46 }
```

- Y finalizando con esta clase, se creó el método `toString()`.
 - Se agregó la anotación `@Override` para indicar que se está reemplazando el método de su clase padre `Objetct`.
 - En este método se considera los atributos de la clase (`name`, `row`, `column`, `status`, `health`).
 - Se utilizaron saltos de con ayuda del `\n`.

```
47 @Override
48 public String toString() {
49     return "Data { " +
50         "\n Name: " + name +
51         "\n Row: " + row +
52         "\n Column: " + column +
53         "\n Status: " + status +
54         "\n Health: " + health +
55         "\n}\n";
56 }
```

- Un ejemplo de como se muestra en la siguiente imagen:

```
EJERCITO "A"
Data {
  Name:  Soldier0XA
  Row:   9
  Column: I
  Status: true
  Health: 4
}

Data {
  Name:  Soldier1XA
  Row:   9
  Column: F
  Status: true
  Health: 2
}

Data {
  Name:  Soldier2XA
  Row:   5
  Column: J
  Status: true
  Health: 2
}
```

Figura 1

Listing 1: Commit: En el primer commit se reutilizaba la clase Soldado.java del laboratorio anterior

```
$ git add .
$ git commit -m "Se reutilizo las clases del laboratorio anterior, para en este caso
    realizarlo con arrayList"
$ git push -u origin main
```

4.1.2. Clase VideoJuego.java

- Primero se comenzó con la creación de la clase.
- Posterior a ello se crearon dos variables de clase:
 - Un ArrayList bidimensional, el cual contendrá a los soldados del ejército.
 - Además de un ArrayList unidimensional que nos servirá para contener a los soldados creados previamente, de esta manera será más fácil trabajar con ellos en los futuros métodos.

```
7 public class VideoJuego3 {
8     static ArrayList<ArrayList<Soldado>> army = new ArrayList<>();
9     static ArrayList<Soldado> army1DA = new ArrayList<>();
10    static ArrayList<Soldado> army1DB = new ArrayList<>();
```

4.1.3. Método para la creación de los ejército

- El método tiene como nombre `createArmy()`.
- Primero se define el número de soldados que contendrá cada ejército, haciendo uso de `Math.random`.
- Luego utilizando un doble bucle for, se inicializa el ArrayList bidimensional de soldados en `null`.
- Ahora se llama dos veces al método `createArmyTeam` para realizar la creación de los dos ejércitos a partir de los tamaños ya establecidos anteriormente.

```
55 public static void createArmy(){
56     int numSoldiersA = (int) (Math.random() * 10) + 1;
57     int numSoldiersB = (int) (Math.random() * 10) + 1;
58
59     for (int i = 0; i < 10; i++){
60         army.add(new ArrayList<>());
61         for (int j = 0; j < 10; j++)
62             army.get(i).add(null);
63     }
64
65     army1DA = createArmyTeam(numSoldiersA, army1DA, "A");
66     army1DB = createArmyTeam(numSoldiersB, army1DB, "B");
67 }
```

Listing 2: Commit: Se implementó el método `createArmy()`

```
$ git add .
$ git commit -m "Se realizaron modificaciones para que ahora el programa trabaje con
    2 ejércitos, se adapto el metodo createArmy con la ayuda de otro metodo que
    trabajara con cada ejercito llamado createArmyTeam"
$ git push -u origin main
```

4.1.4. Método para la creación de ejércitos

- El método tiene como nombre `createArmyTeam()`.
- Recibe como parámetros un entero que es el número de soldados del ejército, el `ArrayList` de soldados a utilizar y un `String` que es el char que va al final del nombre de los soldados, esto último nos ayuda a identificarlos mejor.
- Utilizando un `do while` se crean posiciones aleatorias en el `ArrayList` bidimensional, tomando como condición que dicha posición sea distinta de `null`.
- Finalmente se crea el soldado, se almacena en ambos arreglos y retorna el `ArrayList` unidimensional con los soldados creados.

```
69 public static ArrayList <Soldado> createArmyTeam(int numSoldiers, ArrayList <Soldado>
    army1D, String t){
70     for (int i = 0; i < numSoldiers; i++){
71         int row, col;
72
73         do {
74             row = (int) (Math.random() * 9) + 1;
75             col = (int) (Math.random() * 9) + 1;
76         } while (army.get(row).get(col) != null);
77
78         Soldado s = new Soldado("Soldier" + i + "X" + t, row + 1, (char) (col + 'A'), true,
            (int) (Math.random() * 5) + 1);
79         army1D.add(i, s);
80         army.get(row).set(col, s);
81     }
82     return army1D;
83 }
```

4.1.5. Método para mostrar la tabla con el ejército

- El método tiene como nombre `showArmyTable()`.
- La funcionalidad es simple:
 - Se crean dos `String` (`linesUp`, `linesDown`), estos son usados para la impresión.
 - Primeramente se imprime la parte superior, donde se encuentran las letras que indica las columnas. Seguido de una línea que representa la parte superior de la tabla.
 - Se utilizó dos bucles `for`, el primero para las filas y el segundo para las columnas.
 - En el primero, inicia imprimiendo `linesUp` y luego de un salto de línea imprime el número de fila, se agregó una condicional que nos sirve para darle un toque de simetría a esta enumeración de fila, y así evitar que al momento de imprimir el 10 recorra un espacio.
 - Dentro del segundo, se evalúa si contiene un `Soldado` o `null`; en caso de ser así, imprimirá “—”, en caso de contener a un soldado completará el cuadrado de la tabla incluyendo ejército (A o B), y su nombre simplificado (S + número generado).
 - Finalmente se imprime `linesDown` que completaría la línea inferior de cada fila.

```

85 public static void showArmyTable(ArrayList <ArrayList <Soldado>> army){
86     String linesUp = " | | | | | | | | | |
87     |";
87     String linesDown = "
88     |-----|-----|-----|-----|-----|-----|-----|-----|";
88     System.out.println("      A      B      C      D      E      F      G      H      I
89     J\n"
90     + "
91     -----");
90
91     for (int r = 0; r < army.size(); r++){
92         System.out.println(linesUp);
93         System.out.print(r+1 + ((r != 9) ? " |" : " |"));
94         for (int c = 0; c < army.get(r).size(); c++){
95             System.out.print(" " + (isTeam(army.get(r).get(c), army1DA)? "'A\'" :
96             isTeam(army.get(r).get(c), army1DB)? "'B\'" : ""
97             + ((army.get(r).get(c) != null) ? "S"
98             + army.get(r).get(c).getName().charAt(7) + " |" : " |"));
99
100         System.out.println("\n" + linesDown);
101     }
101     System.out.println();
102 }

```

- Un ejemplo de como se muestra en la siguiente imagen:

TABLA CON LAS UBICACIONES DE LOS SOLDADOS CREADOS:

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										'A'S2
6										
7										'A'S3
8			'B'S0							
9						'A'S1			'A'S0	
10										

Figura 2

Listing 3: Commit: Se adaptó e implementó el método para mostrar la tabla con los soldados de ambos ejércitos

```
$ git add .
$ git commit -m "Se modifiko el metodo showArmyTable, con ayuda de un metodo que
sirve al momento de graficar, este verifica a que team pertenece y segun eso
sera mostrado en la tabla"
$ git push -u origin main
```

4.1.6. Método para verificar si pertenece a un ejército

- El método tiene como nombre `isTeam()`.
- Simplemente dentro del método se recorrerá el `ArrayList` de soldados, en cada iteración se evaluará si pertenece al ejército dado.
- En caso se encuentre uno igual, se retornará `true`, y si es que termina el bucle `for` y no encontró ninguno, retornará `false`.

```
104 public static boolean isTeam(Soldado s, ArrayList <Soldado> army1DT){
105     for (Soldado sA : army1DT)
106         if (sA.equals(s))
107             return true;
108     return false;
109 }
```

4.1.7. Método para mostrar los datos de los soldados de un ejército

- El método tiene como nombre `showArmyData()`.
- Este usa un `for each` para recorrer el arreglo unidimensional de `Soldado`, y luego mostrar sus datos con el `System.out.println`, que a su vez este sigue el formato que se estableció en el método `toString()` de la clase `Soldado.java`.
- La impresión en consola será la misma que la figura 1.

```
111 public static void showArmyData(ArrayList <Soldado> army1D){
112     for (Soldado s : army1D)
113         System.out.println(s);
114 }
```

Listing 4: Commit: Se adaptó e implementó el método para mostrar los datos de los soldados de un ejército

```
$ git add .
$ git commit -m "Se cambio el parametro del metodo showArmyData a un ArrayList, de
esta manera recibira los ArrayList de ambos ejercitos"
$ git push -u origin main
```


4.1.8. Método para mostrar aquellos soldados con más vida de un ejército

- El método tiene como nombre `moreHealth()`.
- Primero recorre el `ArrayList` unidimensional de soldados haciendo uso de un bucle `for`, de esta manera obtendrá el máximo de vida del ejército, el cual será almacenado en un entero `maxHealth`.
- Finalmente imprimirá aquellos soldados que tengan la vida igual a `maxHealth`, con un `for` y un `if` que controlará aquello.

```
116 public static void moreHealth(ArrayList <Soldado> army1MH){
117     int maxHealth = -1;
118     for(Soldado s : army1MH)
119         if (s.getHealth() > maxHealth)
120             maxHealth = s.getHealth();
121
122     for (Soldado s : army1MH)
123         if (s.getHealth() == maxHealth)
124             System.out.println("Nombre: " + s.getName() + " Vida: " + s.getHealth());
125     System.out.println();
126 }
```

- Un ejemplo de como se muestra en la siguiente imagen:

```
Soldado(s) con mayor vida del Ejercito A:
Nombre: Soldier4XA Vida: 5
Nombre: Soldier6XA Vida: 5

Soldado(s) con mayor vida del Ejercito B:
Nombre: Soldier2XB Vida: 5
Nombre: Soldier3XB Vida: 5
```

Figura 3

Listing 5: Commit: Se agregó el método `moreHealth()` que imprimirá aquellos soldados que tengan la mayor vida dentro de su ejército

```
$ git add .
$ git commit -m "Se adapto el metodo moreHealth para trabajar con los ArrayList de
    soldados, este sera aplicado para cada ejercito"
$ git push -u origin main
```

4.1.9. Método para hallar la suma de vida en un ejército

- El método tiene como nombre `sumHealth()`.
- Se utiliza un entero inicializado en 0 para mientras que se recorre el arreglo unidimensional de Soldado con un bucle for each, este entero (sum) va almacenando la vida de todos los soldados.
- Finalmente se retorna `sum` para que sea utilizado en el main.

```
93 public static int sumHealth(ArrayList <Soldado> army1DSH){  
94     int sum = 0;  
95     for (Soldado s : army1DSH)  
96         sum += s.getHealth();  
97     return sum;  
98 }
```

- Un ejemplo de como se muestra en la siguiente imagen:




Figura 4

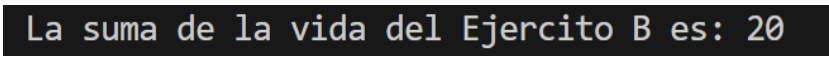


Figura 5

4.1.10. Método para hallar el promedio de vida en un ejército

- El método tiene como nombre `averageHealth()`.
- De manera breve como el método, este retorna una división entre la suma de la vida del ejército, haciendo uso del método `sumHealth()` y dividiendo entre el tamaño del ejército.

```
128 public static double averageHealth(ArrayList <Soldado> army1DAH){  
129     return sumHealth(army1DAH) / army1DAH.size();  
130 }
```

- Un ejemplo de como se muestra en la siguiente imagen:




Figura 6




Figura 7

4.1.11. Método de ordenamiento BubbleSort

- El método tiene como nombre `bubbleSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de Soldado haciendo uso del algoritmo BubbleSort, tomando en cuenta la vida de los soldados que contiene dicho `ArrayList`. Este algoritmo se extrajo de Geeksforgeeks y fue adaptado a este proyecto.

```
139 public static ArrayList <Soldado> bubbleSort(ArrayList <Soldado> army1DBS){
140     ArrayList <Soldado> army1DCopyBubble = new ArrayList<>(army1DBS);
141     int n = army1DCopyBubble.size();
142     boolean swapped;
143     for (int i = 0; i < n - 1; i++) {
144         swapped = false;
145         for (int j = 0; j < n - i - 1; j++)
146             if (army1DCopyBubble.get(j).getHealth() < army1DCopyBubble.get(j+1).getHealth()) {
147                 Soldado temp = army1DCopyBubble.get(j);
148                 army1DCopyBubble.set(j, army1DCopyBubble.get(j+1));
149                 army1DCopyBubble.set(j+1, temp);
150                 swapped = true;
151             }
152         if (!swapped)
153             break;
154     }
155     return army1DCopyBubble;
156 }
```

Listing 6: Commit: Se implementó el método `bubbleSort()` para ordenar los soldados de mayor a menor vida

```
$ git add .
$ git commit -m "Se adapto el algoritmo de ordenamiento en el metodo bubbleSort para
trabajar ahora con los ArrayList de soldados"
$ git push -u origin main
```

4.1.12. Método de ordenamiento InsertionSort

- El método tiene como nombre `insertionSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de Soldado haciendo uso del algoritmo InsertionSort, tomando en cuenta la vida de los soldados que contiene dicho `ArrayList`. Este algoritmo se extrajo de Geeksforgeeks y fue adaptado a este proyecto.

```
158 public static ArrayList <Soldado> insertionSort(ArrayList <Soldado> army1DIS) {
159     int n = army1DIS.size();
160     ArrayList <Soldado> army1DCopyInsertion = new ArrayList<>(army1DIS);
161
162     for (int i = 1; i < n; i++) {
163         Soldado key = army1DCopyInsertion.get(i);
164         int j = i - 1;
165
166         while (j >= 0 && army1DCopyInsertion.get(j).getHealth() < key.getHealth()) {
167             army1DCopyInsertion.set(j+1, army1DCopyInsertion.get(j));
168             j = j - 1;
169         }
170         army1DCopyInsertion.set(j+1, key);
171     }
172     return army1DCopyInsertion;
173 }
```

Listing 7: Commit: Se implementó el método `insertionSort()`

```
$ git add .
$ git commit -m "Se adapto el metodo insertionSort para ordenar los ArrayList de
soldados"
$ git push -u origin main
```

4.1.13. Método de ordenamiento SelectionSort

- El método tiene como nombre `selectionSort()`.
- El método tiene como finalidad ordenar el `ArrayList` unidimensional de `Soldado` haciendo uso del algoritmo `SelectionSort`, tomando en cuenta la vida de los soldados que contiene dicho `ArrayList`. Este algoritmo se extrajo de Geeksforgeeks y fue adaptado a este proyecto.

```
175 public static ArrayList <Soldado> selectionSort(ArrayList <Soldado> army1DSS) {  
176     int n = army1DSS.size();  
177     ArrayList <Soldado> army1DCopySelection = new ArrayList<>(army1DSS);  
178  
179     for (int i = 0; i < n - 1; i++) {  
180         int min_idx = i;  
181  
182         for (int j = i + 1; j < n; j++)  
183             if (army1DCopySelection.get(j).getHealth() >  
184                 army1DCopySelection.get(min_idx).getHealth())  
185                 min_idx = j;  
186  
187         Soldado temp = army1DCopySelection.get(min_idx);  
188         army1DCopySelection.set(min_idx, army1DCopySelection.get(i));  
189         army1DCopySelection.set(i, temp);  
190     }  
191     return army1DCopySelection;  
}
```

Listing 8: Commit: Se implementó el método `selectionSort()`

```
$ git add .  
$ git commit -m "Se adapto el metodo selectionSort el cual es nuestro ultimo metodo  
de algoritmo de ordenamiento para que trabaje con ArrayList de soldados"  
$ git push -u origin main
```

4.1.14. Método para imprimir los soldados de un ejército, ordenados según su vida

- El método tiene como nombre `printArmyHealth()`.
- Este método recibirá un `ArrayList` unidimensional de `Soldado` (previamente ordenado), lo recorrerá usando un bucle `for` y mostrará los soldados, teniendo en cuenta que primero se mostrarán los de mayor vida hasta los de menor.

```
193 public static void printArmyHealth(ArrayList <Soldado> armyPrint){
194     for(int i = 0; i < armyPrint.size(); i++){
195         System.out.println((i + 1) + ". " + armyPrint.get(i).getName() + " Vida: " +
            armyPrint.get(i).getHealth());
196     }
```

Listing 9: Commit: Se implementará el método `showArmyHealth`

```
$ git add .
$ git commit -m "Se adapto el metodo printArmy cambiandole el parametro y el metodo
    que accede al ArrayList, para de esta manera trabajar con los mismo"
$ git push -u origin main
```

- Un ejemplo de como se muestra utilizando los 3 algoritmos de ordenamiento en la siguiente imagen:

```
Ejercitos ordenados (bubbleSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1

Ejercitos ordenados (insertionSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1

Ejercitos ordenados (selectionSort) segun la vida:
EJERCITO A:
1. Soldier0XA Vida: 4
2. Soldier1XA Vida: 1
EJERCITO B:
1. Soldier0XB Vida: 4
2. Soldier3XB Vida: 4
3. Soldier2XB Vida: 3
4. Soldier1XB Vida: 1
```

Figura 8

4.1.15. Método para mostrar al ejército ganador según la vida de sus soldados

- El método tiene como nombre `armyWinnerHealth()`.
- Este método utilizará la suma de vida de cada ejército, hará comparaciones y si uno de los dos es superior en número de vida, se imprimirá que ganó, en caso de ser iguales mostrará aquello.

```
198 public static void armyWinnerHealth(){
199     System.out.println("(Segun la vida)");
200     if (sumHealth(army1DA) > sumHealth(army1DB))
201         System.out.println("El ejercito ganador es: \'A\'");
202     else if (sumHealth(army1DB) > sumHealth(army1DA))
203         System.out.println("El ejercito ganador es: \'B\'");
204     else
205         System.out.println("La batalla quedo en empate");
206     System.out.println();
207 }
```

Listing 10: Commit: Se implementará el método `armyWinnerHealth`

```
$ git add .
$ git commit -m "Se implemento un metodo armyWinnerHealth, este mostrara al ganador
    en caso de una batalla, la cual da como ganador al que tenga la mayor suma de
    vida en soldados"
$ git push -u origin main
```

4.1.16. Método main, utilización de los métodos creados

- En el método principal (main) se utilizarán los métodos creados anteriormente para cumplir con lo pedido en el trabajo.
- Primero llenaremos los `ArrayList` `army1DA` y `army1DB` haciendo uso del método `createArmy()` que a su vez inicializará el arreglo bidimensional de soldados.
- Luego usaremos el método `showArmyData()` para mostrar los soldados creados, siguiendo ese mismo orden, esto para ambos ejércitos (A y B).
- Seguido se llamará al método `showArmyTable` para mostrar la tabla con los soldados ubicados en la misma.
- Seguidamente se llama al método `moreHealth()`, este nos mostrará aquellos que tengan la mayor vida de cada ejército.
- Para mostrar la suma y promedio de vida en nuestros ejércitos, se utiliza los métodos `sumHealth()` y `averageHealth()` respectivamente, para luego ser mostrados en consola.
- Finalmente haciendo uso de los 3 métodos que contienen los algoritmos de ordenamiento (`bubbleSort()`, `insertionSort()`, `selectionsort()`), además del método `printArmyHealth()` para mostrar los ejércitos ordenados según la vida, utilizando cada uno siguiendo el orden mencionado de nuestros algoritmos.

```
11 public static void main(String [] args){
12     createArmy();
13
14     System.out.println("DATOS DE LOS SOLDADOS CREADOS: \n");
15     System.out.println("EJERCITO \"A\"");
16     showArmyData(array1DA);
17     System.out.println("\nEJERCITO \"B\"");
18     showArmyData(array1DB);
19
20     System.out.println("                TABLA CON LAS UBICACIONES DE LOS SOLDADOS CREADOS: \n");
21     showArmyTable(array);
22
23     System.out.println("Soldado(s) con mayor vida del Ejercito A: ");
24     moreHealth(array1DA);
25     System.out.println("Soldado(s) con mayor vida del Ejercito B: ");
26     moreHealth(array1DB);
27
28     System.out.println("La suma de la vida del Ejercito A es: " + sumHealth(array1DA));
29     System.out.println("El promedio de vida del Ejercito A es: " + averageHealth(array1DA));
30     System.out.println("La suma de la vida del Ejercito B es: " + sumHealth(array1DB));
31     System.out.println("El promedio de vida del Ejercito B es: " + averageHealth(array1DB));
32
33     System.out.println("\nEjercitos ordenados (bubbleSort) segun la vida: ");
34     System.out.println("EJERCITO A: ");
35     printArmyHealth(bubbleSort(array1DA));
36     System.out.println("EJERCITO B: ");
37     printArmyHealth(bubbleSort(array1DB));
38
39     System.out.println("\nEjercitos ordenados (insertionSort) segun la vida: ");
40     System.out.println("EJERCITO A: ");
41     printArmyHealth(insertionSort(array1DA));
42     System.out.println("EJERCITO B: ");
43     printArmyHealth(insertionSort(array1DB));
44
45     System.out.println("\nEjercitos ordenados (selectionSort) segun la vida: ");
46     System.out.println("EJERCITO A: ");
47     printArmyHealth(selectionSort(array1DA));
48     System.out.println("EJERCITO B: ");
49     printArmyHealth(selectionSort(array1DB));
50
51     System.out.println();
52     armyWinnerHealth();
53 }
```

Listing 11: Commit: Último commit, donde se cambió el nombre de la clase a el pedido en el trabajo, el cual es: VideoJuego3.java

```
$ git add .
$ git commit -m "Cambiando el nombre de la clase, pasa de ser DemoBatalla a
VideoJuego3"
$ git push -u origin main
```


4.2. Estructura de laboratorio 06

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

```
lab05
|  VideoJuego3.java
|  Soldado.java
|
|-----latex
|    Informe_Lab06.pdf
|    Informe_Lab06.tex
|
|-----img
|    averageHealth.png
|    averageHealth2.png
|    logo_abet.png
|    logo_episunsa.png
|    logo_unsa.jpg
|    moreHealth.png
|    printArmyHealth.png
|    showArmyTable.png
|    sumHealth.png
|    sumHealth2.png
|    toString.png
|
|-----src
|    Soldado.java
|    VideoJuego3.java
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

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

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplió con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe auto calificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	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

Tabla 3: 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	2	
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 están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	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	X	3	
Total		20		18	

6. Referencias

- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html>
- <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
- <https://docs.oracle.com/javase/tutorial/java/java00/methods.html>
- <https://www.geeksforgeeks.org/selection-sort/>
- <https://www.geeksforgeeks.org/bubble-sort/>
- <https://www.geeksforgeeks.org/insertion-sort/>
- <https://es.stackoverflow.com/questions/108171/>