

Informe de Laboratorio 05

Tema: Arreglos Bidimensionales de Objetos

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
05	Arreglos Bidimensionales de Objetos	03 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 Laboratorio5
- Usted deberá crear las dos clases Soldado.java y VideoJuego2.java. Puede reutilizar lo desarrollado en Laboratorio 3 y 4.
- 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 arreglo bidimensional de objetos.
- Inicializar el tablero con n soldados aleatorios entre 1 y 10. Cada soldado tendrá un nombre autogenerado: Soldado0, Soldado1, 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.
- Además de los datos del Soldado con mayor vida, el promedio de puntos de vida de todos los soldados creados, el nivel de vida de todo el ejército, los datos de todos los soldados en el orden que fueron creados y un ranking de poder de todos los soldados creados, del que tiene más nivel de vida al que tiene menos (usar al menos 2 algoritmos de ordenamiento).

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
- Arreglos de Objetos.
- Métodos.
- Métodos de Búsqueda y Ordenamiento.

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/lab05>

4. Trabajo del Laboratorio 05

4.1. Actividad 01

4.1.1. Clase Soldado.java

- Haciendo uso de la clase Nave del laboratorio anterior, se usó para adaptarla al ejercicio actual.
 - Primero copiamos el código tal cual a la carpeta del laboratorio actual (lab05).
- 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:
- Se muestran soldados con sus respectivos datos (atributos).

```
Data {
  Name:  Soldier0
  Row:   4
  Column: J
  Status: true
  Health: 4
}

Data {
  Name:  Soldier1
  Row:   9
  Column: F
  Status: true
  Health: 5
}

Data {
  Name:  Soldier2
  Row:   4
  Column: F
  Status: true
  Health: 5
}
```

Figura 1

Listing 1: Commit: El último commit de la clase Soldado.java que concluía con el método toString()

```
$ git add .  
$ git commit -m "Terminado el metodo constructor y toString."  
$ 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 arreglo bidimensional, el cual contendrá a los soldados del ejército donde en caso no se encuentre uno este tendrá `null` en dichas posiciones.
 - Además de un arreglo unidimensional que nos servirá para contener a los soldados creados previamente, de esta manera será más fácil trabajar con ellos en los futuros métodos.

```
6 public class VideoJuego2 {  
7     static Soldado[] [] army = new Soldado[10][10];  
8     static Soldado[] army1D;
```

4.1.3. Método para la creación del ejército

- El método tiene como nombre `createArmy()`.
- Primero se define el número de soldados que contendrá el arreglo, haciendo uso de `Math.random`.
- Según el número se establece el tamaño del arreglo unidimensional de Soldado, además se iterará sobre el for el mismo número de veces.
- Utilizando un do while se crean posiciones aleatorias en el arreglo bidimensional, tomando como condición que dicha posición sea distinta de `null`.
- Finalmente se crea el soldado, se almacena en ambos arreglos y retorna el arreglo unidimensional.

```
33 public static Soldado[] createArmy(Soldado army[] []){  
34     int numSoldiers = (int) (Math.random() * 10) + 1;  
35     army1D = new Soldado[numSoldiers];  
36     for (int i = 0; i < numSoldiers; i++){  
37         int row, col;  
38  
39         do {  
40             row = (int) (Math.random() * 9) + 1;  
41             col = (int) (Math.random() * 9) + 1;  
42         } while (army[row][col] != null);  
43  
44         Soldado s = new Soldado("Soldier" + i, row + 1, (char) (col + 'A'), true, (int)  
45             (Math.random() * 5) + 1);  
46         army1D[i] = s;  
47         army[row][col] = s;  
48     }  
49     return army1D;  
50 }
```

Listing 2: Commit: En el mismo commit donde se terminó el `toString` y el método constructor, se concluyó con el método `createArmy()`, para verificar su funcionalidad se utilizó la impresión en consola

```
$ git add .
$ git commit -m "Terminada la creacion del ejercito y sus soldados, se imprimieron
sus datos para realizar las pruebas"
$ git push -u origin main
```

4.1.4. 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 su nombre de la siguiente manera `Sold#`, siendo `#` el número del soldado.
 - Finalmente se imprime `linesDown` que completaría la línea inferior de cada fila.

```
52 public static void showArmyTable(Soldado[] [] army){
53     String linesUp = " |   |   |   |   |   |   |   |   |   |";
54     |";
55
56     String linesDown = "
57         |-----|-----|-----|-----|-----|-----|-----|-----|";
58
59     System.out.println("      A      B      C      D      E      F      G      H      I
60         J\n"
61         + "
62         -----");
63
64     for (int r = 0; r < army.length; r++){
65         System.out.println(linesUp);
66         System.out.print(r+1 + ((r != 9) ? " | " : " |"));
67         for (int c = 0; c < army[r].length; c++){
68             System.out.print((army[r][c] != null) ? " Sold" + army[r][c].getName().charAt(7) + "
69                 | " : " |");
70         }
71         System.out.println("\n" + linesDown);
72     }
73     System.out.println();
74 }
```

- 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			Sold0							
3										
4										
5										
6				Sold4		Sold1				
7			Sold3						Sold5	
8										
9										
10						Sold2				

Figura 2

Listing 3: Commit:

```
$ git add .
$ git commit -m "Se termino el metodo showArmyTable acomodando los indice de letras
y numero"
$ git push -u origin main
```

4.1.5. 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.

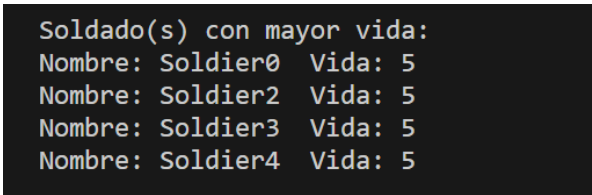
```
71 public static void showArmyData(Soldado[] army1D){
72     for (Soldado s : army1D)
73         System.out.println(s);
74 }
```

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

- El método tiene como nombre `moreHealt()`.
- Primero recorre el arreglo 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`.
- Imprimirá el encabezado ("**Soldado(s) con mayor vida:**")
- Finalmente imprimirá aquellos soldados que tengan la vida igual a `maxHealth`, con un `for` y un `if` que controlará aquello.

```
76 public static void moreHelath(Soldado[] army1D){
77     int maxHealth = -1;
78     for(Soldado s : army1D)
79         if (s.getHealth() > maxHealth)
80             maxHealth = s.getHealth();
81
82     System.out.println("Soldado(s) con mayor vida: ");
83     for (Soldado s : army1D)
84         if (s.getHealth() == maxHealth)
85             System.out.println("Nombre: " + s.getName() + " Vida: " + s.getHealth());
86     System.out.println();
87 }
```

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



```
Soldado(s) con mayor vida:
Nombre: Soldier0 Vida: 5
Nombre: Soldier2 Vida: 5
Nombre: Soldier3 Vida: 5
Nombre: Soldier4 Vida: 5
```

Figura 3

4.1.7. 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(Soldado[] army1D){  
94     int sum = 0;  
95     for (Soldado s : army1D)  
96         sum += s.getHealth();  
97     return sum;  
98 }
```

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

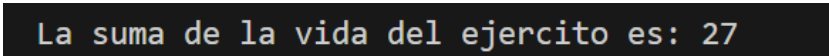


Figura 4

4.1.8. 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.

```
89 public static double averageHealth(Soldado[] army1D){  
90     return sumHealth(army1D) / army1D.length;  
91 }
```

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

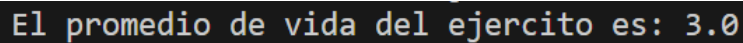


Figura 5

4.1.9. Método de ordenamiento BubbleSort

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

```
100 public static Soldado[] bubbleSort(Soldado[] army1D){
101     Soldado[] army1DCopyBubble = new Soldado[army1D.length];
102     System.arraycopy(army1D, 0, army1DCopyBubble, 0, army1D.length);
103     int n = army1DCopyBubble.length;
104     boolean swapped;
105     for (int i = 0; i < n - 1; i++) {
106         swapped = false;
107         for (int j = 0; j < n - i - 1; j++) {
108             if (army1DCopyBubble[j].getHealth() < army1DCopyBubble[j + 1].getHealth()) {
109                 Soldado temp = army1DCopyBubble[j];
110                 army1DCopyBubble[j] = army1DCopyBubble[j + 1];
111                 army1DCopyBubble[j + 1] = temp;
112                 swapped = true;
113             }
114         }
115         if (!swapped)
116             break;
117     }
118     return army1DCopyBubble;
119 }
```

Listing 4: Commit: Se implementó el método bubbleSort

```
$ git add .
$ git commit -m "Se realizaron modificaciones, ademas de agregar el metodo
    bubbleSort que ordenara el ejercito creado segun la vida de los soldados, este
    se mostrara con el metodo printArmyHealth que aun esta por implementarse"
$ git push -u origin main
```

4.1.10. Método de ordenamiento InsertionSort

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

```
121 public static Soldado[] insertionSort(Soldado[] army1D) {
122     int n = army1D.length;
123     Soldado[] army1DCopyInsertion = new Soldado[n];
124
125     // Copia los elementos del arreglo original al arreglo de copia
126     for (int i = 0; i < n; i++) {
127         army1DCopyInsertion[i] = army1D[i];
128     }
129
130     for (int i = 1; i < n; i++) {
131         Soldado key = army1DCopyInsertion[i];
132         int j = i - 1;
133
134         while (j >= 0 && army1DCopyInsertion[j].getHealth() < key.getHealth()) {
135             army1DCopyInsertion[j + 1] = army1DCopyInsertion[j];
136             j = j - 1;
137         }
138         army1DCopyInsertion[j + 1] = key;
139     }
140     return army1DCopyInsertion;
141 }
```

Listing 5: Commit: Se implementó el método insertionSort

```
$ git add .
$ git commit -m "Se implemento el metodo insertionSort, este ordena el arreglo de
    soldados desde el que tenga mayor vida al menor, ademas de ello se modifiko
    partes del codigo para que al mostrar los distintos datos pedidos sea de una
    forma mas organizada"
$ git push -u origin main
```

4.1.11. Método de ordenamiento SelectionSort

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

```
143 public static Soldado[] selectionSort(Soldado[] army1D) {
144     int n = army1D.length;
145     Soldado[] army1DCopySelection = new Soldado[n];
146
147     for (int i = 0; i < n; i++) {
148         army1DCopySelection[i] = army1D[i];
149     }
150
151     for (int i = 0; i < n - 1; i++) {
152         int min_idx = i;
153
154         for (int j = i + 1; j < n; j++) {
155             if (army1DCopySelection[j].getHealth() > army1DCopySelection[min_idx].getHealth()) {
156                 min_idx = j;
157             }
158         }
159
160         Soldado temp = army1DCopySelection[min_idx];
161         army1DCopySelection[min_idx] = army1DCopySelection[i];
162         army1DCopySelection[i] = temp;
163     }
164
165     return army1DCopySelection;
166 }
```

Listing 6: Commit: Se implementó el método selectionSort

```
$ git add .
$ git commit -m "Se implemento el metodo selectionSort que realizara el mismo
ordenamiento que los dos metodos anteriormente creados, pero usando el algoritmo
de selection sort"
$ git push -u origin main
```

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

- El método tiene como nombre `printArmyHealth()`.
- Este método recibirá un arreglo 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.

```
168 public static void printArmyHealth(Soldado[] armyPrint){
169     for(int i = 0; i < armyPrint.length; i++)
170         System.out.println((i + 1) + ". " + armyPrint[i].getName() + " Vida: " +
171             armyPrint[i].getHealth());
172 }
```

Listing 7: Commit: Se implementará el método `showArmyHealth`

```
$ git add .
$ git commit -m "Se implemento el metodo printArmyHealth, este nos mostrara el
ejercito ordenado segun la vida, sera usado luego de aplicarle los algoritmos de
ordenamiento a nuestro ejercito"
$ git push -u origin main
```

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

```
Ejercito ordenado (bubbleSort) segun la vida:
1. Soldier0 Vida: 5
2. Soldier2 Vida: 5
3. Soldier3 Vida: 5
4. Soldier4 Vida: 5
5. Soldier1 Vida: 4
6. Soldier5 Vida: 1
7. Soldier6 Vida: 1
8. Soldier7 Vida: 1

Ejercito ordenado (insertionSort) segun la vida:
1. Soldier0 Vida: 5
2. Soldier2 Vida: 5
3. Soldier3 Vida: 5
4. Soldier4 Vida: 5
5. Soldier1 Vida: 4
6. Soldier5 Vida: 1
7. Soldier6 Vida: 1
8. Soldier7 Vida: 1

Ejercito ordenado (selectionSort) segun la vida:
1. Soldier0 Vida: 5
2. Soldier2 Vida: 5
3. Soldier3 Vida: 5
4. Soldier4 Vida: 5
5. Soldier1 Vida: 4
6. Soldier5 Vida: 1
7. Soldier6 Vida: 1
8. Soldier7 Vida: 1
PS E:\fp2-23b>
```

Figura 6

4.1.13. 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 el arreglo `army1D` 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.
- Seguidamente se llama al método `moreHealth()`, este nos mostrará aquellos que tengan la mayor vida del ejército.
- Para mostrar la suma y promedio de vida en el ejército, 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 el ejército ordenado según la vida utilizando cada uno según el orden mencionado.

```
9 public static void main(String [] args){
10     army1D = createArmy(army);
11
12     System.out.println("DATOS DE LOS SOLDADOS CREADOS: \n");
13     showArmyData(army1D);
14
15     System.out.println("                TABLA CON LAS UBICACIONES DE LOS SOLDADOS CREADOS: \n");
16     showArmyTable(army);
17
18     moreHelath(army1D);
19
20     System.out.println("La suma de la vida del ejercito es: " + sumHealth(army1D));
21     System.out.println("El promedio de vida del ejercito es: " + averageHealth(army1D));
22
23     System.out.println("\nEjercito ordenado (bubbleSort) segun la vida: ");
24     printArmyHealth(bubbleSort(army1D));
25
26     System.out.println("\nEjercito ordenado (insertionSort) segun la vida: ");
27     printArmyHealth(insertionSort(army1D));
28
29     System.out.println("\nEjercito ordenado (selectionSort) segun la vida: ");
30     printArmyHealth(selectionSort(army1D));
31 }
```

Listing 8: Commit: Último commit, donde se cambió el nombre de la clase a el pedido en el trabajo, el cual es: VideoJuego2.java

```
$ git add .
$ git commit -m "Cambiando el nombre de la clase, pasa de ser DemoBatalla a
    VideoJuego2"
$ git push -u origin main
```

4.2. Estructura de laboratorio 05

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

```
lab05
|  Soldado.java
|  VideoJuego2.java
|
|-----latex
|   Informe_Lab05.pdf
|   Informe_Lab05.tex
|
|-----img
|   averageHealth.png
|   logo_abet.png
|   logo_episunsa.png
|   logo_unsa.jpg
|   moreHealth.png
|   printArmyHealth.png
|   showArmyTable.png
|   sumHealth.png
|   toString.png
|
|-----src
|   Soldado.java
|   VideoJuego2.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/Arrays.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/>