

Informe de Laboratorio 03

Tema: Arreglos de Objetos

Nota	

${f Estudiante}$	Escuela	${f Asign atura}$
		Fundamentos de la
Christian Mestas Zegarra	Escuela Profesional de	Programación 2
cmestasz@unsa.edu.pe	Ingeniería de Sistemas	Semestre: II
		Código: 1701213

Laboratorio	${f Tema}$	Duración
03	Arreglos de Objetos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 18 Setiembre 2023	Al 25 Setiembre 2023

1. Tarea

- Actividad 1: Analice, complete y pruebe el Código de la clase DemoBatalla.
- Actividad 2: Solucionar la Actividad 4 de la Práctica 1 pero usando arreglo de objetos.
- Actividad 3: Solucionar la Actividad 5 de la Práctica 1 pero usando arreglos de objetos.



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.
- Arreglos de objetos.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- https://github.com/cmestasz/fp2-23b.git
- URL para el laboratorio 03 en el Repositorio GitHub.
- https://github.com/cmestasz/fp2-23b/tree/main/fase01/lab03

4. Actividades con el repositorio GitHub

Creando plantillas

\$ mkdir lab03
\$ cd lab03
\$ code Nave.java
\$ code DemoBatalla.java
\$ code Soldado.java
\$ code Actividad2.java
\$ code Actividad3.java

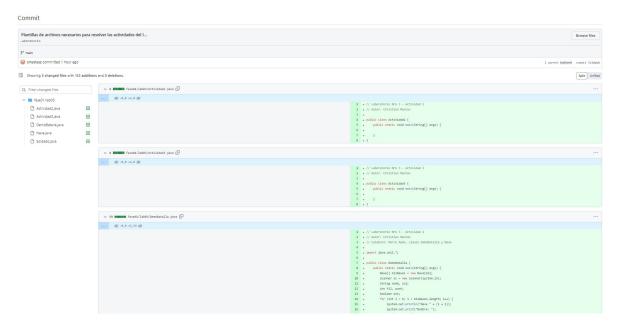
laboratorio"

Primer Commit / Plantillas

- \$ git add .
 \$ git commit -m "Plantillas de archivos necesarios para resolver las actividades del
- \$ git push







Primer Commit.

Actualizando Nave.java y DemoBatalla.java

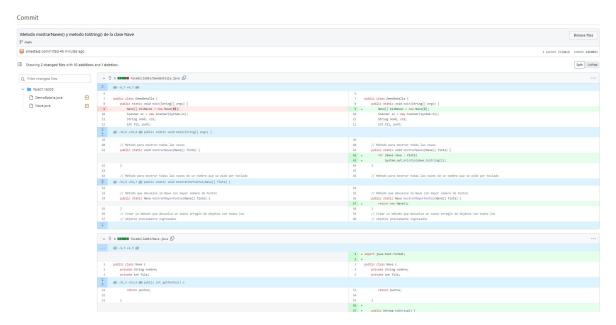
```
$ code Nave.java
$ code DemoBatalla.java
```

Segundo - Sexto Commit / Nave.java y DemoBatalla.java

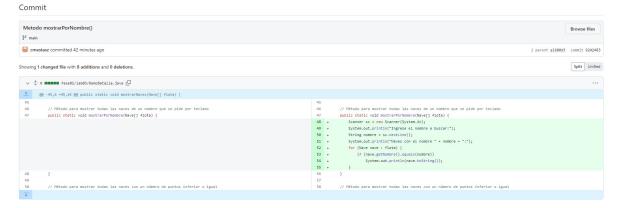
```
$ git add .
$ git commit -m "Metodo mostrarNaves() y metodo toString() de la clase Nave"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarPorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo mostrarMayorPuntos()"
$ code DemoBatalla.java
$ git commit -m "Metodo mostrarMayorPuntos()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo desordenar()"
$ git push
```







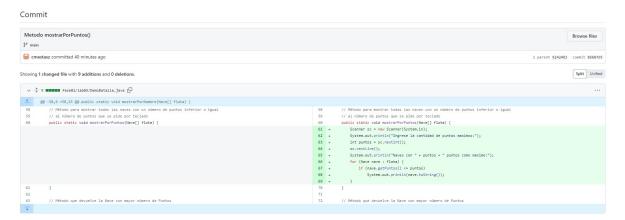
Segundo Commit.



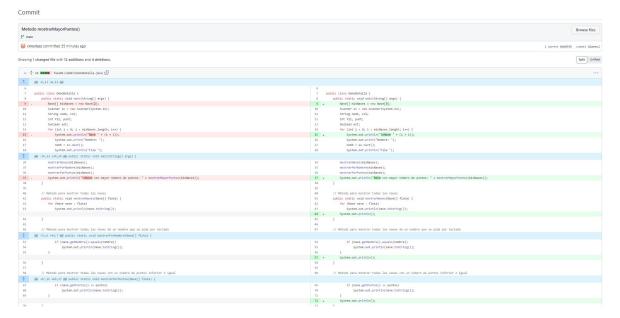
Tercer Commit.







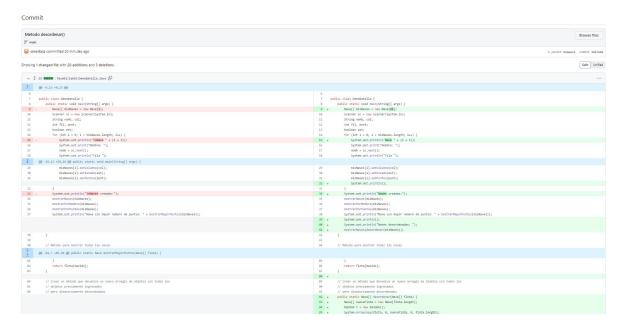
Cuarto Commit.



Quinto Commit.







Sexto Commit.

Actualizando Soldado.java

\$ code Soldado.java

Septimo Commit / Soldado.java

\$ git add Soldado.java
\$ git commit -m "Clase Soldado"
\$ git push





```
Clare Soldado

Prant

Character committed 17 minutes ago

1 parent 194766 consist ee00579

Showing I changed file with 27 additions and 1 deletion.

5 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1 public clare Soldado

2 public clare Soldado

1 public clare Soldado

1
```

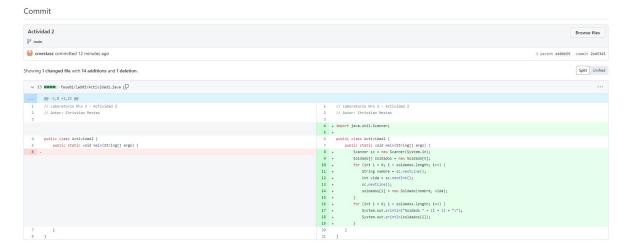
Septimo Commit.

Actualizando Actividad2.java

\$ code Actividad2.java

Octavo Commit / Soldado.java

```
$ git add Actividad2.java
$ git commit -m "Actividad 2"
$ git push
```



Octavo Commit.

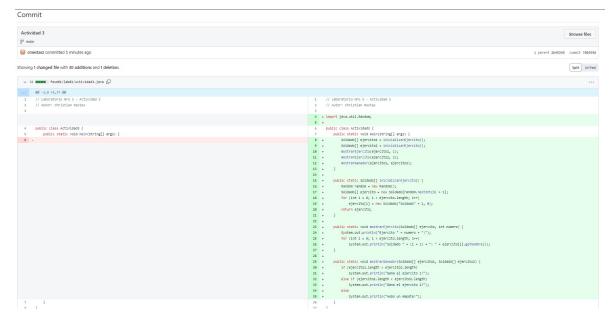


Actualizando Actividad3.java

\$ code Actividad3.java

Noveno Commit / Actividad3.java

```
$ git add Actividad3.java
$ git commit -m "Actividad 3"
$ git push
```



Noveno Commit.





5. Código desarrollado

Nave.java

```
package fase01.lab03;
   public class Nave {
       private String nombre;
       private int fila;
       private String columna;
       private boolean estado;
       private int puntos;
       // Metodos mutadores
       public void setNombre(String n) {
           nombre = n;
12
14
       public void setFila(int f) {
           fila = f;
16
17
18
       public void setColumna(String c) {
19
           columna = c;
20
21
       public void setEstado(boolean e) {
23
           estado = e;
26
       public void setPuntos(int p) {
27
           puntos = p;
28
29
30
31
       // Metodos accesores
       public String getNombre() {
32
33
           return nombre;
34
       }
35
36
       public int getFila() {
           return fila;
39
40
41
       public String getColumna() {
42
           return columna;
43
44
       }
46
       public boolean getEstado() {
47
           return estado;
48
49
       }
50
       public int getPuntos() {
```





- Clase que guarda nombre, fila, columna, estado y puntos de la nave.
- Posee tanto setters como getters para todos los atributos.
- Posee el metodo toString() para poder imprimir el objeto.



DemoBatalla.java

```
// Laboratorio Nro 3 - Actividad 1
   // Autor: Christian Mestas
   // Colaboro: Marco Aedo, clases DemoBatalla y Nave
   package fase01.lab03;
   import java.util.*;
   public class DemoBatalla {
10
       public static void main(String[] args) {
           Nave[] misNaves = new Nave[10];
11
           Scanner sc = new Scanner(System.in);
           String nomb, col;
13
           int fil, punt;
           boolean est;
           for (int i = 0; i < misNaves.length; i++) {</pre>
              System.out.println("Nave " + (i + 1));
17
              System.out.print("Nombre: ");
18
              nomb = sc.nextLine();
19
              System.out.print("Fila: ");
20
              fil = sc.nextInt();
21
              sc.nextLine();
              System.out.print("Columna: ");
               col = sc.nextLine();
24
              System.out.print("Estado: ");
              est = sc.nextBoolean();
              sc.nextLine();
              System.out.print("Puntos: ");
              punt = sc.nextInt();
               sc.nextLine();
30
              misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
31
                   misNaves
              misNaves[i].setNombre(nomb);
              misNaves[i].setFila(fil);
33
              misNaves[i].setColumna(col);
              misNaves[i].setEstado(est);
              misNaves[i].setPuntos(punt);
36
              System.out.println();
           System.out.println("Naves creadas:");
           mostrarNaves(misNaves);
           mostrarPorNombre(misNaves);
           mostrarPorPuntos(misNaves);
42
           System.out.println("Nave con mayor numero de puntos: " +
43
               mostrarMayorPuntos(misNaves));
           System.out.println();
44
           System.out.println("Naves desordenadas: ");
           mostrarNaves(desordenar(misNaves));
       }
48
       // Metodo para mostrar todas las naves
49
       public static void mostrarNaves(Nave[] flota) {
50
           for (Nave nave : flota)
              System.out.println(nave.toString());
           System.out.println();
```





```
54
        // Metodo para mostrar todas las naves de un nombre que se pide por teclado
56
        public static void mostrarPorNombre(Nave[] flota) {
            Scanner sc = new Scanner(System.in);
58
            System.out.println("Ingrese el nombre a buscar:");
59
            String nombre = sc.nextLine();
            System.out.println("Naves con el nombre " + nombre + ":");
61
            for (Nave nave : flota) {
62
               if (nave.getNombre().equals(nombre))
63
                   System.out.println(nave.toString());
64
            System.out.println();
        }
        // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
69
        // al numero de puntos que se pide por teclado
70
        public static void mostrarPorPuntos(Nave[] flota) {
71
            Scanner sc = new Scanner(System.in);
            System.out.println("Ingrese la cantidad de puntos maximos:");
73
            int puntos = sc.nextInt();
74
            sc.nextLine();
            System.out.println("Naves con " + puntos + " puntos como maximo:");
            for (Nave nave : flota) {
               if (nave.getPuntos() <= puntos)</pre>
                   System.out.println(nave.toString());
            System.out.println();
81
82
83
        // Metodo que devuelve la Nave con mayor numero de Puntos
84
        public static Nave mostrarMayorPuntos(Nave[] flota) {
85
            int maxIdx = 0;
            for (int i = 0; i < flota.length; i++) {</pre>
               if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
                   maxIdx = i;
89
90
            return flota[maxIdx];
91
        }
        // Crear un metodo que devuelva un nuevo arreglo de objetos con todos los
        // objetos previamente ingresados
95
        // pero aleatoriamente desordenados
96
        public static Nave[] desordenar(Nave[] flota) {
97
            Nave[] nuevaFlota = new Nave[flota.length];
98
            Random r = new Random();
99
            System.arraycopy(flota, 0, nuevaFlota, 0, flota.length);
100
            for (int idx = 0; idx < nuevaFlota.length; idx++) {</pre>
               int nIdx = r.nextInt(nuevaFlota.length);
               Nave t = nuevaFlota[idx];
               nuevaFlota[idx] = nuevaFlota[nIdx];
104
               nuevaFlota[nIdx] = t;
            }
            return nuevaFlota;
108
```





- Se instancian 10 objetos de la clase nave en el arreglo misNaves, todos los datos son llenados por el usuario.
- mostrarNaves() imprime todas las naves usando su método toString().
- mostrarPorNombre() imprime todas las naves con el nombre que el usuario ingrese.
- mostrarPorPuntos() imprime todas las naves con menor o igual numero de puntos a lo que el usuario ingrese.
- mostrarMayorPuntos() retorna la nave que tiene la mayor cantidad de puntos, que luego es impresa.
- desordenar() retorna un nuevo arreglo de naves con las naves desordenadas al azar, que luego es impreso con el método mostrarNaves().



Soldado.java

```
package fase01.lab03;
   public class Soldado {
       private String nombre;
       private int vida;
       public Soldado(String nombre, int vida) {
           setNombre(nombre);
           setVida(vida);
10
11
       public void setNombre(String nombre) {
           this.nombre = nombre;
13
       public void setVida(int vida) {
16
           this.vida = vida;
17
18
19
       public String getNombre() {
20
           return nombre;
21
24
       public int getVida() {
           return vida;
25
26
       public String toString() {
           return ("Nombre: " + nombre + ". Vida: " + vida);
29
30
31
```

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



Actividad2.java

```
// Laboratorio Nro 3 - Actividad 2
   // Autor: Christian Mestas
   package fase01.lab03;
   import java.util.Scanner;
   public class Actividad2 {
       public static void main(String[] args) {
10
           Scanner sc = new Scanner(System.in);
           Soldado[] soldados = new Soldado[5];
11
           for (int i = 0; i < soldados.length; i++) {</pre>
              String nombre = sc.nextLine();
              int vida = sc.nextInt();
              sc.nextLine();
               soldados[i] = new Soldado(nombre, vida);
16
17
           for (int i = 0; i < soldados.length; i++) {</pre>
18
               System.out.println("Soldado " + (i + 1) + ":");
19
              System.out.println(soldados[i]);
20
21
       }
```

- Los 5 nombres y sus respectivas vidas son leídas, para luego instanciar objetos de la clase soldado con sus respectivos atributos.
- Luego los atributos son impresos en el orden en el que se ingresaron.



Actividad3.java

```
// Laboratorio Nro 3 - Actividad 3
   // Autor: Christian Mestas
   package fase01.lab03;
   import java.util.Random;
6
   public class Actividad3 {
       public static void main(String[] args) {
10
           Soldado[] ejercito1 = inicializarEjercito();
           Soldado[] ejercito2 = inicializarEjercito();
11
           mostrarEjercito(ejercito1, 1);
           mostrarEjercito(ejercito2, 2);
           mostrarGanador(ejercito1, ejercito2);
14
       }
       public static Soldado[] inicializarEjercito() {
17
           Random random = new Random();
18
           Soldado[] ejercito = new Soldado[random.nextInt(5) + 1];
19
           for (int i = 0; i < ejercito.length; i++)</pre>
20
               ejercito[i] = new Soldado("Soldado" + i, 0);
21
           return ejercito;
23
       }
24
       public static void mostrarEjercito(Soldado[] ejercito, int numero) {
25
           System.out.println("Ejercito " + numero + ":");
           for (int i = 0; i < ejercito.length; i++)</pre>
               System.out.println("Soldado " + (i + 1) + ": " + ejercito[i].getNombre());
       }
30
       public static void mostrarGanador(Soldado[] ejercito1, Soldado[] ejercito2) {
31
           if (ejercito1.length > ejercito2.length)
32
               System.out.println("Gana el ejercito 1!");
33
           else if (ejercito1.length < ejercito2.length)</pre>
34
              System.out.println("Gana el ejercito 2!");
36
           else
37
              System.out.println("Hubo un empate!");
       }
38
   }
39
```

- Se crean 2 ejércitos con el método inicializarEjercito(), que crea un arreglo de soldados de tamaño al azar entre 1 y 5.
- mostrarEjercito() imprime una lista de los soldados de cada ejército.
- mostrarGanador() compara los tamaños de los ejércitos y declara al ganador como el ejército mas grande.





6. Ejecución del código

DemoBatalla.java

```
Nave 1
   Nombre: naveuno
   Fila: 5
   Columna: G
   Estado: true
   Puntos: 6
   Nave 2
   Nombre: navedos
   Fila: 2
   Columna: A
   Estado: false
   Puntos: 7
13
14
   Nave 3
15
   Nombre: navetres
   Fila: 8
   Columna: J
   Estado: true
   Puntos: 11
   Nave 4
   Nombre: navecuatro
   Fila: 8
   Columna: D
   Estado: true
   Puntos: 9
27
28
   Nave 5
   Nombre: navecinco
   Fila: 6
   Columna: F
   Estado: false
   Puntos: 11
   Nave 6
   Nombre: naveseis
   Fila: 2
   Columna: B
   Estado: false
40
   Puntos: 7
41
42
   Nave 7
   Nombre: navesiete
   Fila: 3
   Columna: F
   Estado: true
   Puntos: 2
   Nave 8
   Nombre: naveocho
   Fila: 4
```



```
Columna: A
    Estado: true
    Puntos: 10
   Nave 9
57
   Nombre: navenueve
   Fila: 3
   Columna: I
   Estado: false
   Puntos: 13
62
   Nave 10
   Nombre: navediez
   Fila: 2
    Columna: E
    Estado: true
   Puntos: 7
69
70
   Naves creadas:
71
   naveuno: (5, G). Estado: true. Puntos: 6
   navedos: (2, A). Estado: false. Puntos: 7
   navetres: (8, J). Estado: true. Puntos: 11
   navecuatro: (8, D). Estado: true. Puntos: 9
   navecinco: (6, F). Estado: false. Puntos: 11
   naveseis: (2, B). Estado: false. Puntos: 7
   navesiete: (3, F). Estado: true. Puntos: 2
    naveocho: (4, A). Estado: true. Puntos: 10
    navenueve: (3, I). Estado: false. Puntos: 13
    navediez: (2, E). Estado: true. Puntos: 7
81
82
   Ingrese el nombre a buscar:
83
   navecinco
84
   Naves con el nombre navecinco:
    navecinco: (6, F). Estado: false. Puntos: 11
    Ingrese la cantidad de puntos maximos: 8
88
   Naves con 8 puntos como maximo:
   naveuno: (5, G). Estado: true. Puntos: 6
   navedos: (2, A). Estado: false. Puntos: 7
    naveseis: (2, B). Estado: false. Puntos: 7
    navesiete: (3, F). Estado: true. Puntos: 2
    navediez: (2, E). Estado: true. Puntos: 7
94
95
   Nave con mayor numero de puntos: navenueve: (3, I). Estado: false. Puntos: 13
96
97
   Naves desordenadas:
   navecinco: (6, F). Estado: false. Puntos: 11
   naveseis: (2, B). Estado: false. Puntos: 7
100
   naveocho: (4, A). Estado: true. Puntos: 10
   navesiete: (3, F). Estado: true. Puntos: 2
   navediez: (2, E). Estado: true. Puntos: 7
   navetres: (8, J). Estado: true. Puntos: 11
   naveuno: (5, G). Estado: true. Puntos: 6
   navenueve: (3, I). Estado: false. Puntos: 13
    navedos: (2, A). Estado: false. Puntos: 7
   navecuatro: (8, D). Estado: true. Puntos: 9
```





Actividad2.java

```
Luis Pancho
   15
   Marco Marco
   Jorge Mandela
   16
   Jose Rufs
   Esqueche Almino
11
   Soldado 1:
   Nombre: Luis Pancho. Vida: 15
   Soldado 2:
   Nombre: Marco Marco. Vida: 20
   Soldado 3:
   Nombre: Jorge Mandela. Vida: 16
   Soldado 4:
   Nombre: Jose Rufs. Vida: 18
   Soldado 5:
19
   Nombre: Esqueche Almino. Vida: 21
```

Actividad3.java

```
Ejercito 1:

Soldado 1: Soldado0

Soldado 2: Soldado1

Soldado 3: Soldado2

Soldado 4: Soldado3

Soldado 5: Soldado4

Ejercito 2:

Soldado 1: Soldado0

Gana el ejercito 1!
```



7. Estructura de laboratorio 03

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

```
lab03/
|--- Nave.java
|--- DemoBatalla.java
|--- Soldado.java
|--- Actividad2.java
|--- Actividad3.java
|--- ejec01.bash
|--- ejec02.bash
|--- ejec03.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
```



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 facil 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 cumplio 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 items.
- El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

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



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	1.5	
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	3	
	Total	20		16.5	

9. Referencias

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