

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la Programación II				
TÍTULO DE LA PRÁCTICA:	Arreglos de Objetos				
NÚMERO DE PRÁCTICA:	03	AÑO LECTIVO:	2024	NRO. SEMESTRE:	2
FECHA DE PRESENTACIÓN	02/10/2024	HORA DE PRESENTACIÓN	hh/mm/ss		
INTEGRANTE (s) Santiago Enrique Palma Apaza				NOTA (0-20)	Nota colocada por el docente
DOCENTE(s): Pinto Oppe Lino José					

RESULTADOS Y PRUEBAS
<p>I. EJERCICIOS RESUELTOS:</p> <p><i>El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.</i></p> <p><i>El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado</i></p> <p>Cree un Proyecto llamado Laboratorio3 Usted deberá agregar las clases Nave.java y DemoBatalla.java. 1. Analice, complete y pruebe el Código de la clase DemoBatalla</p> <p>Clase Nave</p> <p>La clase Nave representa una unidad en un juego. Contiene atributos que describen las características de la nave, como su nombre, posición (fila y columna), estado y puntos. Los métodos de la clase permiten modificar y acceder a estos atributos.</p> <p>Métodos de la Clase Nave</p> <ol style="list-style-type: none"> setNombre(String n) <ul style="list-style-type: none"> Establece el nombre de la nave. setFila(int f) <ul style="list-style-type: none"> Establece la fila en la que se encuentra la nave. setColumna(String c)

- Establece la columna en la que se encuentra la nave.
- 4. **setEstado(boolean e)**
 - Establece el estado de la nave (activa o inactiva).
- 5. **setPuntos(int p)**
 - Establece el número de puntos que tiene la nave.
- 6. **getNombre()**
 - Devuelve el nombre de la nave.
- 7. **getFila()**
 - Devuelve la fila en la que se encuentra la nave.
- 8. **getColumna()**
 - Devuelve la columna en la que se encuentra la nave.
- 9. **getEstado()**
 - Devuelve el estado de la nave.
- 10. **getPuntos()**
 - Devuelve el número de puntos que tiene la nave.

```
11 public class Nave {
12     private String nombre;
13     private int fila;
14     private String columna;
15     private boolean estado;
16     private int puntos;
17
18     // Métodos mutadores
19     public void setNombre(String n) {
20         nombre = n;
21     }
22
23     public void setFila(int f) {
24         fila = f;
25     }
26
27     public void setColumna(String c) {
28         columna = c;
29     }
30
31     public void setEstado(boolean e) {
32         estado = e;
33     }
34
35     public void setPuntos(int p) {
36         puntos = p;
37     }
38
39     // Métodos accesorios
40     public String getNombre() {
41         return nombre;
42     }
43
44     public int getFila() {
45         return fila;
46     }
47
48     public String getColumna() {
49         return columna;
50     }
51
52     public boolean getEstado() {
53         return estado;
54     }
55
56     public int getPuntos() {
57         return puntos;
```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

Métodos de la Clase DemoBatalla

1. mostrarNaves(Nave[] flota)

- Muestra información sobre todas las naves en el arreglo flota.

2. mostrarPorNombre(Nave[] flota)

- Permite al usuario buscar y mostrar naves por nombre.

3. mostrarPorPuntos(Nave[] flota)

- Muestra todas las naves con puntos menores o iguales a un valor ingresado por el usuario.

4. mostrarMayorPuntos(Nave[] flota)

- Devuelve la nave con el mayor número de puntos en el arreglo flota.

5. desordenarNaves(Nave[] flota)

- Devuelve un nuevo arreglo de naves desordenadas.

```

16  import java.util.*;
17  public class DemoBatalla {
18      public static void main(String[] args) {
19          Nave[] misNaves = new Nave[10];
20          Scanner sc = new Scanner(System.in);
21          String nomb, col;
22          int fil, punt;
23          boolean est;
24
25          for (int i = 0; i < misNaves.length; i++) {
26              System.out.println("Nave " + (i + 1));
27              System.out.print("Nombre: ");
28              nomb = sc.next();
29              System.out.print("Fila: ");
30              fil = sc.nextInt();
31              System.out.print("Columna: ");
32              col = sc.next();
33              System.out.print("Estado (true/false): ");
34              est = sc.nextBoolean();
35              System.out.print("Puntos: ");
36              punt = sc.nextInt();
37
38              misNaves[i] = new Nave();
39
40              misNaves[i].setNombre(nomb);
41              misNaves[i].setFila(fil);
42              misNaves[i].setColumna(col);
43              misNaves[i].setEstado(est);
44              misNaves[i].setPuntos(punt);
45          }
46
47          System.out.println("\nNaves creadas:");
48          mostrarNaves(misNaves);
49          mostrarPorNombre(misNaves);
50          mostrarPorPuntos(misNaves);
51          System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves).getNombre());
52      }
53  }
54
55  public static void mostrarNaves(Nave[] Naves) {

```

```

55 public static void mostrarNaves(Nave[] flota) {
56     for (Nave nave : flota) {
57         if (nave != null) {
58             System.out.println("Nombre: " + nave.getNombre() + ", Fila: " + nave.getFila() +
59                 ", Columna: " + nave.getColumna() + ", Estado: " + nave.getEstado() +
60                 ", Puntos: " + nave.getPuntos());
61         }
62     }
63 }
64
65 public static void mostrarPorNombre(Nave[] flota) {
66     Scanner sc = new Scanner(System.in);
67     System.out.print("\nIngrese el nombre de la nave a buscar: ");
68     String nombreBuscado = sc.next();
69
70     boolean encontrado = false;
71     for (Nave nave : flota) {
72         if (nave != null && nave.getNombre().equalsIgnoreCase(nombreBuscado)) {
73             System.out.println("Nave encontrada: " + nave.getNombre() + ", Fila: " + nave.getFila() +
74                 ", Columna: " + nave.getColumna() + ", Estado: " + nave.getEstado() +
75                 ", Puntos: " + nave.getPuntos());
76             encontrado = true;
77         }
78     }
79     if (!encontrado) {
80         System.out.println("No se encontró ninguna nave con ese nombre.");
81     }
82 }
83
84 public static void mostrarPorPuntos(Nave[] flota) {
85     Scanner sc = new Scanner(System.in);
86     System.out.print("\nIngrese el número de puntos: ");
87     int puntosBuscados = sc.nextInt();
88
89     System.out.println("Naves con puntos menores o iguales a " + puntosBuscados + ":");
90     boolean encontrado = false;
91     for (Nave nave : flota) {
92         if (nave != null && nave.getPuntos() <= puntosBuscados) {
93             System.out.println("Nombre: " + nave.getNombre() + ", Fila: " + nave.getFila() +
94                 ", Columna: " + nave.getColumna() + ", Estado: " + nave.getEstado() +
95                 ", Puntos: " + nave.getPuntos());
96             encontrado = true;
97         }
98     }
99     if (!encontrado) {
100         System.out.println("No se encontraron naves con esos puntos.");
101     }
102 }

```



```
103
104 [-] public static Nave mostrarMayorPuntos(Nave[] flota) {
105     Nave naveMayor = flota[0];
106
107 [-]     for (int i = 1; i < flota.length; i++) {
108         Nave nave = flota[i];
109 [-]         if (nave.getPuntos() > naveMayor.getPuntos()) {
110             naveMayor = nave;
111         }
112     }
113
114     return naveMayor;
115 }
116 [-] public static Nave[] desordenarNaves(Nave[] flota) {
117     Nave[] desordenadas = new Nave[flota.length];
118     System.arraycopy(flota, 0, desordenadas, 0, flota.length);
119     Random rand = new Random();
120 [-]     for (int i = desordenadas.length - 1; i > 0; i--) {
121         int j = rand.nextInt(i + 1);
122         Nave temp = desordenadas[i];
123         desordenadas[i] = desordenadas[j];
124         desordenadas[j] = temp;
125     }
126
127     return desordenadas;
128 }
129 }
```

Actividad 4: escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida. Ingresar sus datos y después mostrarlos.

```
DemoBatalla.java  Nave.java  Soldados.java  Ejercitos.java
1 package Lab_5;
2
3 import java.util.Scanner;
4
5 class Soldado {
6     private String nombre;
7     private int nivelVida;
8
9     public Soldado(String nombre, int nivelVida) {
10         this.nombre = nombre;
11         this.nivelVida = nivelVida;
12     }
13
14     public String getNombre() {
15         return nombre;
16     }
17
18     public int getNivelVida() {
19         return nivelVida;
20     }
21 }
22
23 public class Soldados {
24     public static void main(String[] args) {
25         final int NUM_SOLDADOS = 5;
26         Soldado[] soldados = new Soldado[NUM_SOLDADOS];
27         Scanner sc = new Scanner(System.in);
28
29         for (int i = 0; i < NUM_SOLDADOS; i++) {
30             System.out.println("Ingrese el nombre del Soldado " + (i + 1) + ": ");
31             String nombre = sc.nextLine();
32             System.out.print("Ingrese el nivel de vida del Soldado " + (i + 1) + ": ");
33             int nivelVida = sc.nextInt();
34             sc.nextLine();
35             soldados[i] = new Soldado(nombre, nivelVida);
36         }
37
38         System.out.println("\nDatos de los soldados:");
39         for (Soldado soldado : soldados) {
40             System.out.println("Soldado: " + soldado.getNombre() + ", Nivel de Vida: " + soldado.getNivelVida());
41         }
42     }
43 }
44
```

Actividad 5: escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.


```
DemoBatalla.java  Nave.java  Soldados.java  Ejercitos.java X
1 package Lab_5;
2
3 import java.util.Random;
4
5 class Ejercito {
6     private String[] soldados;
7
8     public Ejercito(int numSoldados) {
9         soldados = new String[numSoldados];
10        for (int i = 0; i < numSoldados; i++) {
11            soldados[i] = "Soldado" + i;
12        }
13    }
14
15    public int getTamaño() {
16        return soldados.length;
17    }
18
19    public void mostrarSoldados() {
20        for (String soldado : soldados) {
21            System.out.println(soldado);
22        }
23    }
24 }
25
26 public class Ejercitos {
27     public static void main(String[] args) {
28         Random rand = new Random();
29
30         Ejercito ejercito1 = new Ejercito(rand.nextInt(5) + 1);
31         Ejercito ejercito2 = new Ejercito(rand.nextInt(5) + 1);
32
33         System.out.println("Ejército 1:");
34         ejercito1.mostrarSoldados();
35
36         System.out.println("\nEjército 2:");
37         ejercito2.mostrarSoldados();
38
39         determinarGanador(ejercito1, ejercito2);
40     }
41
42     public static void determinarGanador(Ejercito ejercito1, Ejercito ejercito2) {
43         System.out.println("\nComparando ejércitos...");
44         int tamaño1 = ejercito1.getTamaño();
45         int tamaño2 = ejercito2.getTamaño();
46
47         if (tamaño1 > tamaño2) {
48             System.out.println("El Ejército 1 es el ganador con " + tamaño1 + " soldados.");
49         } else if (tamaño2 > tamaño1) {
50             System.out.println("El Ejército 2 es el ganador con " + tamaño2 + " soldados.");
51         } else {
52             System.out.println("Empate: ambos ejércitos tienen " + tamaño1 + " soldados.");
53         }
54     }
55 }
```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 10</p>

COMMITTS

```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas
$ git init
Initialized empty Git repository in C:/Users/santi/OneDrive/Escritorio/sas/.git/

santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git add Nave.java



santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git commit -m "Se añadió la clase Nave con atributos y métodos básicos: nombre
, fila, columna, estado y puntos. Ahora implementa la creación de naves y permit
e al usuario configurar nombre, posición, estado y puntos."
[master (root-commit) 765c2e6] Se añadió la clase Nave con atributos y métodos b
ásicos: nombre, fila, columna, estado y puntos. Ahora implementa la creación de
naves y permite al usuario configurar nombre, posición, estado y puntos.
1 file changed, 49 insertions(+)
create mode 100644 Nave.java


santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git remote add origin https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_
ENRIQUE_LABORATORIO_03.git
```


```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git branch
* master

santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 580 bytes | 580.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_ENRIQUE_LAB
ORATORIO_03/pull/new/master
remote:
To https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_0
3.git
 * [new branch]      master -> master

santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$
```


	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 11</p>



PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03
Public
Pin
Unwatch


master had recent pushes 27 seconds ago
 Compare & pull request

main
2 Branches
0 Tags

Add file
Code


santiagopalma12 Create PALMA_APAZA_SANTIAGO_LAB03_FP2_01
 fdc8df7 · 14 minutes ago
2 Commits


PALMA_APAZA_SANTIAGO_ENRIQUE_LABORA... Create PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO...
 2 days ago

```


santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git commit -m "Se mplementa la creación de naves y permite al usuario configurar nombre, posición, estado y puntos. Ahora añade el metodo para mostrar las naves creadas con sus atributos."
[master 429cde3] Se mplementa la creación de naves y permite al usuario configurar nombre, posición, estado y puntos. Ahora añade el metodo para mostrar las naves creadas con sus atributos.
1 file changed, 39 insertions(+)
create mode 100644 DemoBatalla.java


```

```

santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 790 bytes | 790.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03.git
   765c2e6..429cde3  master -> master


```


PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03
Public
Pin
Unwatch


master had recent pushes 1 minute ago
 Compare & pull request

main
2 Branches
0 Tags

Add file
Code



santiagopalma12 Create PALMA_APAZA_SANTIAGO_LAB03_FP2_01
 fdc8df7 · 21 minutes ago
2 Commits


	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 12</p>

```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 714 bytes | 714.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03.git
   429cde3..c8c389e  master -> master
```

```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git add DemoBatalla.java


santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git commit -m "Añade la funcionalidad para buscar y mostrar una nave por su nombre e Implementa la funcionalidad para mostrar naves con puntos menores o iguales a un valor ingresado. y por ultimo añadir también el metodo para desordenar naves"
[master c8c389e] Añade la funcionalidad para buscar y mostrar una nave por su nombre e Implementa la funcionalidad para mostrar naves con puntos menores o iguales a un valor ingresado. y por ultimo añadir también el metodo para desordenar naves
1 file changed, 14 insertions(+), 7 deletions(-)
```


PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03
Public
Pin
Unwatch




master had recent pushes 1 minute ago
 Compare & pull request

main
2 Branches
0 Tags

Add file
Code

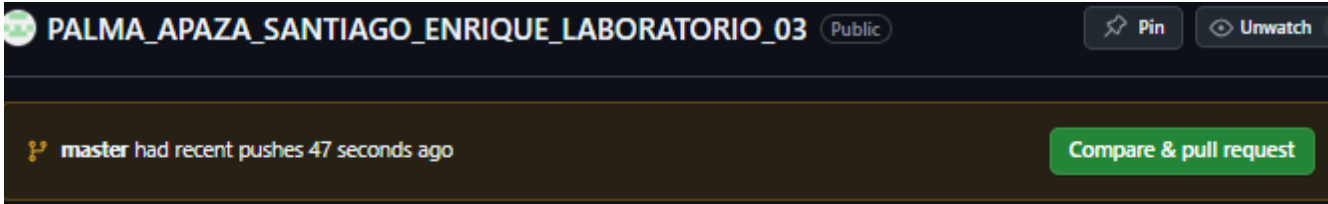

santiagopalma12 Create PALMA_APAZA_SANTIAGO_LAB03_FP2_01
 fdc8df7 · 29 minutes ago
2 Commits

```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git commit -m "Se concluyó con el trabajo, esta es la fase final. Probar valores y señalar errores."
[master 99234c4] Se concluyó con el trabajo, esta es la fase final. Probar valores y señalar errores.
1 file changed, 74 insertions(+), 4 deletions(-)
```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 13</p>

```
santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.27 KiB | 1.27 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/santiagopalma12/PALMA_APAZA_SANTIAGO_ENRIQUE_LABORATORIO_03.git
c8c389e..99234c4 master -> master

santi@5lofiu MINGW64 ~/Onedrive/Escritorio/sas (master)
$
```



II. PRUEBAS

¿Con qué valores comprobaste que tu práctica estuvo correcta?

Ingresa valores tipo String para los nombres de las naves y también para las columnas. usé enteros para las filas y para los puntos.

Output - Laboratorio 05 (run)



```
run:
Nave 1
Nombre: pescador1
Fila: 1
Columna: 1
Estado (true/false): true
Puntos: 100
Nave 2
Nombre: destructor
Fila: 2
Columna: 2
Estado (true/false): true
Puntos: 20
Nave 3
Nombre: huascar
Fila: 4
Columna: 4
Estado (true/false): true
Puntos: 30
Nave 4
Nombre: ian
Fila: 3
Columna: 3
Estado (true/false): true
Puntos: 300
```

```
Nave 6
Nombre: felicidades
Fila: 6
Columna: 6
Estado (true/false): true
Puntos: 10
Nave 7
Nombre: pinta
Fila: 7
Columna: 7
Estado (true/false): true
Puntos: 30
Nave 8
Nombre: niña
Fila: 1
Columna: 2
Estado (true/false): true
Puntos: 30
Nave 9
Nombre: santamaria
Fila: 3
Columna: 1
Estado (true/false): true
Puntos: 10
Nave 10
Nombre: atahualpa
Fila: 4
Columna: 5
Estado (true/false): false
Puntos: 60
```

¿Qué resultado esperas obtener para cada valor de entrada?

Espero que las naves se guarden con los nombres respectivos, en las filas y columnas respectivas, con el valor booleano ingresado y por último con la cantidad de puntos ingresada. Luego que me enseñe todas las naves ingresadas, que me pregunte por buscar esta nave y enseñarme todos sus datos y por último pedirme un número para enseñar de manera desordenada los valores con esa o menor puntuación. Y finalmente mostrar la nave con más puntos.

¿Qué valor o comportamiento obtuviste para cada valor de entrada?

Lo esperado en la consigna anterior.

Naves creadas:

Nombre: pescador1, Fila: 1, Columna: 1, Estado: true, Puntos: 100
Nombre: destructor, Fila: 2, Columna: 2, Estado: true, Puntos: 20
Nombre: huascar, Fila: 4, Columna: 4, Estado: true, Puntos: 30
Nombre: ian, Fila: 3, Columna: 3, Estado: true, Puntos: 300
Nombre: viejo, Fila: 5, Columna: 5, Estado: true, Puntos: 300
Nombre: felicidades, Fila: 6, Columna: 6, Estado: true, Puntos: 10
Nombre: pinta, Fila: 7, Columna: 7, Estado: true, Puntos: 30
Nombre: niña, Fila: 1, Columna: 2, Estado: true, Puntos: 30
Nombre: santamaria, Fila: 3, Columna: 1, Estado: true, Puntos: 10
Nombre: atahualpa, Fila: 4, Columna: 5, Estado: false, Puntos: 60

Ingrese el nombre de la nave a buscar: atahualpa

Nave encontrada: atahualpa, Fila: 4, Columna: 5, Estado: false, Puntos: 60

Ingrese el número de puntos: 30

Naves con puntos menores o iguales a 30:

Nombre: destructor, Fila: 2, Columna: 2, Estado: true, Puntos: 20
Nombre: huascar, Fila: 4, Columna: 4, Estado: true, Puntos: 30
Nombre: felicidades, Fila: 6, Columna: 6, Estado: true, Puntos: 10
Nombre: pinta, Fila: 7, Columna: 7, Estado: true, Puntos: 30
Nombre: niña, Fila: 1, Columna: 2, Estado: true, Puntos: 30
Nombre: santamaria, Fila: 3, Columna: 1, Estado: true, Puntos: 10

Nave con mayor número de puntos: ian

BUILD SUCCESSFUL (total time: 2 minutes 48 seconds)

Ejercicio 4

```
Console X
<terminated> Soldados [Java Application] C:\Users\Usuario24\
Ingrese el nombre del Soldado 1:
pepe
Ingrese el nivel de vida del Soldado 1: 10
Ingrese el nombre del Soldado 2:
juan
Ingrese el nivel de vida del Soldado 2: 30
Ingrese el nombre del Soldado 3:
pedro
Ingrese el nivel de vida del Soldado 3: 5
Ingrese el nombre del Soldado 4:
manuel
Ingrese el nivel de vida del Soldado 4: 4
Ingrese el nombre del Soldado 5:
segio
Ingrese el nivel de vida del Soldado 5: 4
|
Datos de los soldados:
Soldado: pepe, Nivel de Vida: 10
Soldado: juan, Nivel de Vida: 30
Soldado: pedro, Nivel de Vida: 5
Soldado: manuel, Nivel de Vida: 4
Soldado: segio, Nivel de Vida: 4
```

Ejercicio 5

```
Console X
<terminated> Ejercitos [Java Application] C:\Users\Usuario24B\p2
Ejército 1:
Soldado0
Soldado1
Soldado2
Soldado3

Ejército 2:
Soldado0
Soldado1
Soldado2
Soldado3
Soldado4

Comparando ejércitos...
El Ejército 2 es el ganador con 5 soldados.
```

RUBRICA

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	1	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	x	1	
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		16	

CONCLUSIONES

Durante la sesión de laboratorio se implementaron exitosamente las funcionalidades solicitadas para la gestión de buques utilizando programación orientada a objetos en Java. El código desarrollado permitió la creación, configuración y manipulación de objetos Nave, con capacidad de buscar por nombre y filtrar por puntos, mostrando un claro entendimiento de los conceptos de arrays y clases. La práctica resultó efectiva para reforzar

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 18</p>

la gestión de estructuras de control, métodos y acceso a atributos privados, elementos claves en el desarrollo de software modular. Los resultados obtenidos fueron satisfactorios y la sesión ayudó a consolidar los fundamentos en la gestión de datos y objetos.

METODOLOGÍA DE TRABAJO

Análisis del problema: Se inició con una comprensión clara de los requerimientos de la práctica, identificando los atributos y métodos necesarios para la clase **Nave**.

Diseño del programa: Se definió la estructura del código, diseñando la clase **Nave** con sus respectivos métodos para gestionar los atributos de cada nave.

Codificación por etapas: Se implementó el código en fases. Primero, la creación de naves y el almacenamiento en un array; luego, las funcionalidades de búsqueda y filtrado por atributos (nombre y puntos).

Pruebas y depuración: Se realizaron pruebas unitarias para verificar que cada funcionalidad (creación, búsqueda, y filtrado) funcionara correctamente. Se corrigieron errores encontrados durante las pruebas.

Revisión y optimización: Una vez funcional, se revisó el código para mejorar su legibilidad y eficiencia, asegurando que todas las excepciones posibles estuvieran controladas y se usara un código limpio.

REFERENCIAS Y BIBLIOGRAFÍA

Aedo López, M. W. (2019). *Fundamentos de programación I: Java Básico (1. ed.)*. Universidad Nacional de San Agustín. ISBN 978-612-4337-55-0.