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



## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
<b>ASIGNATURA:</b>	<i>Fundamentos de la programación 2</i>				
<b>TÍTULO DE LA PRÁCTICA:</b>	<i>Arreglos de objetos, Búsquedas y Ordenamientos</i>				
<b>NÚMERO DE PRÁCTICA:</b>	<i>Número de práctica</i>	<b>AÑO LECTIVO:</b>	<i>Año que corresponde</i>	<b>NRO. SEMESTRE:</b>	<i>Número de semestre</i>
<b>FECHA DE PRESENTACIÓN</b>	<i>15/10/2024</i>	<b>HORA DE PRESENTACIÓN</b>	<i>08:49:59</i>		
<b>INTEGRANTE (s)</b> <i>Layme Salas Rodrigo Fabricio</i>				<b>NOTA (0-20)</b>	
<b>DOCENTE(s):</b> <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<b>I. EJERCICIOS RESUELTOS:</b> <b>CÓDIGO:</b> <b>CLASE MAIN:</b>

```
1  /*Propósito: Recrear un DemoBatalla con ordenamientos en naves de diferentes formas */
2  import java.util.*;
3  public class DemoBatalla {
4      public static Scanner sc = new Scanner(System.in); //VARIABLE GLOBAL DE SCANNER
5      Run|Debug
6      public static void main(String[] args) {
7          Nave[] misNaves = new Nave[4];
8          String nomb, col;
9          int fil, punt;
10         boolean est;
11         for (int i = 0; i < misNaves.length; i++) { //GENERACIÓN DE NAVES
12             System.out.println("Nave " + (i + 1));
13             System.out.print(s:"Nombre: ");
14             nomb = sc.next();
15             System.out.print(s:"Fila: ");
16             fil = sc.nextInt();
17             System.out.print(s:"Columna: ");
18             col = sc.next();
19             System.out.print(s:"Estado: ");
20             est = sc.nextBoolean();
21             System.out.print(s:"Puntos: ");
22             punt = sc.nextInt();
23             misNaves[i] = new Nave();
24             misNaves[i].setNombre(nomb);
25             misNaves[i].setFila(fil);
26             misNaves[i].setColumna(col);
27             misNaves[i].setEstado(est);
28             misNaves[i].setPuntos(punt);
29         }
30         System.out.println(x:"\nNaves creadas:");
31         mostrarNaves(misNaves); // MUESTRA LA INFORMACIÓN DE LAS NAVES COMO SE INGRESARON
32         mostrarPorNombre(misNaves); // MUESTRA LA INFORMACIÓN DE LA O LAS NAVES CON EL NOMBRE INGRESADO (en función)
33         mostrarPorPuntos(misNaves); // MUESTRA LA INFORMACIÓN DE LA O LAS NAVES CON IGUAL O MENOR PUNTAJE INGRESADO (en función)
34         System.out.println("\nNave con mayor número de puntos: " + mostrarMayorPuntos(misNaves)); //MUESTRA LA NAVE CON MAYOR PUNTAJE
35         System.out.println(x:"Ingresa un nombre de una nave para realizar una búsqueda: ");
36         String nombre = sc.next();
37         int pos = busquedaLinealNombre(misNaves, nombre); // REGISTRA LA POSICIÓN DE LA NAVE QUE SE QUIERE BUSCAR
38         if (pos != -1) // IMPRIME SU INFORMACIÓN SI ES DIFERENTE DE -1
39             System.out.println("INFORMACIÓN: " + misNaves[pos].toString());
40         else
41             System.out.println(x:"¡ERROR! No se encontró una nave con ese nombre");
42         System.out.println(x:"ORDENAMIENTO POR PUNTOS: ");
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 3</p>

```

42     ordenarPorPuntosBurbuja(misNaves);           // ORDENAMIENTO POR PUNTOS CON MÉTODO BURBUJA
43     mostrarNaves(misNaves);
44     System.out.println(x:"ORDENAMIENTO POR NOMBRE: ");
45     ordenarPorNombreBurbuja(misNaves); // ORDENAMIENTO POR ORDEN ALFABÉTICO CON MÉTODO BURBUJA
46     mostrarNaves(misNaves);
47     System.out.print(s:"Ingrese un nombre de nave para realizar un búsqueda binaria: ");
48     String nombreBinaria = sc.next();
49     pos = busquedaBinariaNombre(misNaves, nombreBinaria); // ALMACENA LA POSICIÓN DE LA NAVE BUSCADA
50     if (pos != -1)                                     // IMPRIME SU INFORMACIÓN SI ES DIFERENTE DE -1
51     {
52         System.out.println(misNaves[pos].toString());
53     }
54     else
55     {
56         System.out.println(x:"ERROR, NOMBRE NO ENCONTRADO");
57     }
58     ordenarPorPuntosSeleccion(misNaves); // ORDENA POR PUNTOS CON EL MÉTODO DE SELECCIÓN
59     mostrarNaves(misNaves);               // ORDENA POR PUNTOS CON EL MÉTODO DE SELECCIÓN
60     ordenarPorNombreSeleccion(misNaves);
61     mostrarNaves(misNaves);               // ORDENA POR ORDEN ALFABÉTICO CON EL MÉTODO DE SELECCIÓN
62     ordenarPorPuntosInsercion(misNaves);
63     mostrarNaves(misNaves);               // ORDENA POR PUNTOS CON EL MÉTODO DE INSERCIÓN
64     ordenarPorNombreInsercion(misNaves);
65     mostrarNaves(misNaves);               // ORDENA POR ORDEN ALFABÉTICO CON EL MÉTODO DE INSERCIÓN
66 }
67
68 public static void mostrarNaves(Nave[] var0) {
69     for (int var1 = 0; var1 < var0.length; ++var1) {
70         System.out.println("NAVE " + (var1 + 1) + ":");
71         System.out.println(var0[var1].toString()); //SE MUESTRA LA INFORMACIÓN CON EL toString DE LA CLASE NAVE
72     }
73 }
74
75 public static void mostrarPorNombre(Nave[] var0) {
76     System.out.println(x:"Ingrese el nombre de las naves que desea mostrar:");
77     String var1 = sc.next();
78     for (int var2 = 0; var2 < var0.length; ++var2) {
79         if (var0[var2].getNombre().equals(var1)) { // CONDICIÓN PARA SOLO IMPRIMIR LAS NAVES CON IGUAL NOMBRE
80             System.out.println("Nave " + (var2 + 1));
81             System.out.println(var0[var2].toString());
82         }
83     }
84 }
85
86 }

```

```

81 public static void mostrarPorPuntos(Nave[] var0) {
82     System.out.println(
83         x:"Ingrese una cantidad de puntos, se mostrar\u00c3\u00a1 las naves que tengan menor o igual puntaje: ");
84     int var1 = sc.nextInt();
85     for (int var2 = 0; var2 < var0.length; ++var2)
86         if (var0[var2].getPuntos() == var1 || var0[var2].getPuntos() < var1) { // CONDICION PARA IMPRIMIR SOLO LAS NAVES CON IGUAL O MENOR PUNTAJE
87             System.out.println("NAVE " + (var2 + 1) + ":\n");
88             System.out.println(var0[var2].toString());
89         }
90     }
91
92 public static Nave mostrarMayorPuntos(Nave[] var0) {
93     Nave var1 = var0[0];
94     for (int var2 = 1; var2 < var0.length; ++var2) {
95         if (var0[var2].getPuntos() > var1.getPuntos()) {
96             var1 = var0[var2]; // VA ALMACENANDO LA NAVE CON MAYOR PUNTAJE
97         }
98     }
99     return var1;
100 }
101
102 public static int busquedaLinealNombre(Nave[] flota, String s) {
103     for (int i = 0; i < flota.length; i++) {
104         if (flota[i].getNombre().equals(s)) //CUANDO ENCUENTRA LA POSICIÓN DE LA NAVE CON EL NOMBRE INGRESADO RETORNA SU POSICIÓN
105             return i;
106     }
107     return -1;
108 }
109
110 public static void ordenarPorPuntosBurbuja(Nave[] flota) {
111     boolean intercambio = true;
112     while (intercambio) {
113         intercambio = false; // LO MANTIENE FALSO HASTA QUE SE HAYA UN INTERCAMBIO, SINO SE SALE DEL BUCLE
114         for (int i = 0; i < flota.length - 1; i++)
115             if (flota[i].getPuntos() > flota[i + 1].getPuntos()) {
116                 intercambio = true;
117                 Nave temp = new Nave(); //VARIABLE TEMPORAL PARA EL INTERCAMBIO
118                 temp = flota[i + 1];
119                 flota[i + 1] = flota[i];
120                 flota[i] = temp;

```

```
121     }
122   }
123 }
124
125 public static void ordenarPorNombreBurbuja(Nave[] flota) {
126     boolean intercambio = true;
127     while (intercambio) {
128         intercambio = false;
129         for (int i = 0; i < flota.length - 1; i++) {
130             if (flota[i].getNombre().compareTo(flota[i + 1].getNombre()) > 0) { // compareTo INDICA SI ES MAYOR O MENOR EN UN ORDEN ALFABÉTICO
131                 intercambio = true;
132                 Nave temp = new Nave();
133                 temp = flota[i + 1];
134                 flota[i + 1] = flota[i];
135                 flota[i] = temp;
136             }
137         }
138     }
139 }
140
141 public static int busquedaBinariaNombre(Nave[] flota, String s) {
142     int baja = 0, alta = flota.length - 1;
143     while (baja <= alta) {
144         int media = (baja + alta) / 2;
145         int comparacion = flota[media].getNombre().compareTo(s);
146         if (comparacion == 0) {
147             return media; // Se encontró el nombre
148         } else if (comparacion < 0) {
149             baja = media + 1; // Buscar en la parte derecha
150         } else {
151             alta = media - 1; // Buscar en la parte izquierda
152         }
153     }
154     return -1; // No se encontró el nombre
155 }
156
157 public static void ordenarPorPuntosSeleccion(Nave[] flota) {
158     for (int i = 0; i < flota.length - 1; i++) {
159         int menor = i;
```



```
160         for (int j = i + 1; j < flota.length; j++) {
161             if (flota[j].getPuntos() < flota[menor].getPuntos()) {
162                 menor = j; // Almacena la posición del menor
163             }
164         }
165         // Intercambio de elementos
166         Nave temp = flota[menor];
167         flota[menor] = flota[i];
168         flota[i] = temp;
169     }
170 }
171
172 public static void ordenarPorNombreSeleccion(Nave[] flota) {
173     for (int i = 0; i < flota.length - 1; i++) {
174         int menor = i;
175         for (int j = i + 1; j < flota.length; j++) {
176             if (flota[j].getNombre().compareTo(flota[menor].getNombre()) < 0) {
177                 menor = j; // Almacena la posición del menor alfabéticamente
178             }
179         }
180         // Intercambio de elementos
181         Nave temp = flota[menor];
182         flota[menor] = flota[i];
183         flota[i] = temp;
184     }
185 }
186
187 public static void ordenarPorPuntosInsercion(Nave[] flota) {
188     for (int i = 1; i < flota.length; i++) {
189         Nave naveActual = flota[i]; // GUARDA LA NAVE PARA QUE NO DESAPAREZCA AL DESPLAZAR
190         int j = i - 1;
191         while (j >= 0 && flota[j].getPuntos() > naveActual.getPuntos()) {
192             flota[j + 1] = flota[j]; // DESPLAZA TODAS LAS NAVES A LA DERECHA
193             j--;
194         }
195         flota[j + 1] = naveActual; // ASIGNA EL VALOR DE LA NAVE INTERCAMBIADA
196     }
197 }
198
199 public static void ordenarPorNombreInsercion(Nave[] flota) {
200     for (int i = 1; i < flota.length; i++) {
201         Nave naveActual = flota[i];
202         int j = i - 1;
203         while (j >= 0 && flota[j].getNombre().compareTo(naveActual.getNombre()) > 0) { // COMPARA ALFABÉTICAMENTE
204             flota[j + 1] = flota[j]; // DESPLAZA TODAS LAS NAVES A LA DERECHA
205             j--; // RETROCEDE EL INDEX HASTA QUE NO CUMPLA LA CONDICIÓN
206         }
207         flota[j + 1] = naveActual;
208     }
209 }
210 }
```

**CLASE NAVE:**

```
1 public class Nave {
2     private String nombre;
3     private int fila;
4     private String columna;
5     private boolean estado;
6     private int puntos;
7     public void setNombre(String n) {
8         nombre = n;
9     }
10    public void setFila(int f) {
11        fila = f;
12    }
13    public void setColumna(String c) {
14        columna = c;
15    }
16    public void setEstado(boolean e) {
17        estado = e;
18    }
19    public void setPuntos(int p) {
20        puntos = p;
21    }
22    public String getNombre() {
23        return nombre;
24    }
25    public int getFila() {
26        return fila;
27    }
28    public String getColumna() {
29        return columna;
30    }
31    public boolean getEstado() {
32        return estado;
33    }
34    public int getPuntos() {
35        return puntos;
36    }
37    public String toString() {
38        return "Nombre: " + nombre + "\nFila: " + fila + "\nColumna: " + columna + "\nEstado: " + estado + "\nPuntos: "
39            + puntos;
40    }
41 }
```

**PRUEBAS:**

**TRABAJÉ CON 4 NAVES PARA AGILIZAR LA TOMA DE CAPTURAS PARA ESTE INFORME**  
**INGRESO DE DATOS y MUESTRA DE NAVES CREADAS:**

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 8</p>

```

Nave 1
Nombre: A
Fila: 1
Columna: 2
Estado: TRUE
Puntos: 10
Nave 2
Nombre: D
Fila: 2
Columna: 3
Estado: FALSE
Puntos: 5
Nave 3
Nombre: C
Fila: 3
Columna: 4
Estado: TRUE
Puntos: 7
Nave 4
Nombre: B
Fila: 4
Columna: 5
Estado: TRUE
Puntos: 1

```

Naves creadas:

```

NAVE 1:
Nombre: A
Fila: 1
Columna: 2
Estado: true
Puntos: 10
NAVE 2:
Nombre: D
Fila: 2
Columna: 3
Estado: false
Puntos: 5
NAVE 3:
Nombre: C
Fila: 3
Columna: 4
Estado: true
Puntos: 7
NAVE 4:
Nombre: B
Fila: 4
Columna: 5
Estado: true
Puntos: 1
Ingrese el nombre

```



**MOSTRANDO INFORMACIÓN DE NAVES CON EL NOMBRE INGRESADO:**

```

Ingrese el nombre de las naves que desea mostrar:
B
Nave 4
Nombre: B
Fila: 4
Columna: 5
Estado: true
Puntos: 1

```



	<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: 9</p>

#### **MOSTRANDO NAVES CON MENOR O IGUAL PUNTAJE AL INGRESADO:**

```

Ingrese una cantidad de puntos, se mostrarán las naves que tengan menor o igual puntaje:
5
NAVE 2:

Nombre: D
Fila: 2
Columna: 3
Estado: false
Puntos: 5
NAVE 4:

Nombre: B
Fila: 4
Columna: 5
Estado: true
Puntos: 1

```

#### **MOSTRANDO LA NAVE CON MAYOR PUNTAJE:**

```

Nave con mayor número de puntos: Nombre: A
Fila: 1
Columna: 2
Estado: true
Puntos: 10



```

#### **BÚSQUEDA LINEAL POR NOMBRE:**

```

Ingresa un nombre de una nave para realizar una búsqueda:
C
INFORMACIÓN: Nombre: C
Fila: 3
Columna: 4
Estado: true
Puntos: 7

```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> 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>

#### ORDENAMIENTO POR PUNTOS:

##### ORDENAMIENTO POR PUNTOS:

NAVE 1:

Nombre: B

Fila: 4

Columna: 5

Estado: true

Puntos: 1

NAVE 2:

Nombre: D

Fila: 2

Columna: 3

Estado: false

Puntos: 5

NAVE 3:

Nombre: C

Fila: 3

Columna: 4

Estado: true

Puntos: 7

NAVE 4:



Nombre: A

Fila: 1

Columna: 2

Estado: true

Puntos: 10

	<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>



#### ORDENAMIENTO POR NOMBRE:

ORDENAMIENTO POR NOMBRE:

```
NAVE 1:
Nombre: A
Fila: 1
Columna: 2
Estado: true
Puntos: 10
NAVE 2:
Nombre: B
Fila: 4
Columna: 5
Estado: true
Puntos: 1
NAVE 3:
Nombre: C
Fila: 3
Columna: 4
Estado: true
Puntos: 7
NAVE 4:
Nombre: D
Fila: 2
Columna: 3
Estado: false
Puntos: 5
```

#### BÚSQUEDA BINARIA POR NOMBRE:



```
Ingrese un nombre de nave para realizar un búsqueda binaria: A
Nombre: A
Fila: 1
Columna: 2
Estado: true
Puntos: 10
```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> 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>

**ORDENAMIENTO POR PUNTOS Y POR NOMBRE POR MÉTODO DE SELECCIÓN:**

NAVE 1:	NAVE 1:
Nombre: B	Nombre: A
Fila: 4	Fila: 1
Columna: 5	Columna: 2
Estado: true	Estado: true
Puntos: 1	Puntos: 10
NAVE 2:	NAVE 2:
Nombre: D	Nombre: B
Fila: 2	Fila: 4
Columna: 3	Columna: 5
Estado: false	Estado: true
Puntos: 5	Puntos: 1
NAVE 3:	NAVE 3:
Nombre: C	Nombre: C
Fila: 3	Fila: 3
Columna: 4	Columna: 4
Estado: true	Estado: true
Puntos: 7	Puntos: 7
NAVE 4:	NAVE 4:
Nombre: A	Nombre: D
Fila: 1	Fila: 2
Columna: 2	Columna: 3
Estado: true	Estado: false
Puntos: 10	Puntos: 5

**ORDENAMIENTO POR PUNTOS Y POR NOMBRE POR MÉTODO DE INSERCIÓN:**

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

NAVE 1:	NAVE 1:
Nombre: B	Nombre: A
Fila: 4	Fila: 1
Columna: 5	Columna: 2
Estado: true	Estado: true
Puntos: 1	Puntos: 10
NAVE 2:	NAVE 2:
Nombre: D	Nombre: B
Fila: 2	Fila: 4
Columna: 3	Columna: 5
Estado: false	Estado: true
Puntos: 5	Puntos: 1
NAVE 3:	NAVE 3:
Nombre: C	Nombre: C
Fila: 3	Fila: 3
Columna: 4	Columna: 4
Estado: true	Estado: true
Puntos: 7	Puntos: 7
NAVE 4:	NAVE 4:
Nombre: A	Nombre: D
Fila: 1	Fila: 2
Columna: 2	Columna: 3
Estado: true	Estado: false
Puntos: 10	Puntos: 5

X

## II. PRUEBAS

*¿Con que valores comprobaste que tu práctica estuviera correcta?*

*Con valores int, String y booleanos, además datos que parecía que el programa aceptaría como caracteres especiales para probar como funcionaría cada método.*

*¿Qué resultado esperabas obtener para cada valor de entrada?*



*Esperaba que se almacenara dentro del objeto y atributo que quería; esperaba no tener errores pero tuve varios al momento de construir correctamente el ordenamiento por inserción, pero pronto lo resolví.*

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

*Obtuve, al final, el correcto, una secuencia limpia de los métodos usados en el main y sin ningún error.*

## III. CUESTIONARIO:

**PRUEBAS DE COMMIT HECHO EN GIT BASH:**

	<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: 14</p>

```
PS C:\Users\ASUS\PF2> git add LAYME_SALAS_LABORATORIO_04/DemoBatalla.java
```

```
PS C:\Users\ASUS\PF2> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   LAYME_SALAS_LABORATORIO_04/DemoBatalla.java


Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Copia de Rubrica de calificacion de laboratorio.docx

PS C:\Users\ASUS\PF2> git commit -m "codigo de lab 4 terminado"
Author identity unknown
```

El primer paso es mi ejecutar los comandos git add y git status, está vez lo hice desde el terminal.

```
PS C:\Users\ASUS\PF2> git push
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 2.57 KiB | 1.28 MiB/s, done.
Total 4 (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/F4brici0L4yme/PF2.git
   c8422e9..e172aa1  main -> main
PS C:\Users\ASUS\PF2> |
```

Como ya tenía el repositorio enlazado y no era la primera vez que hacía un git push, el envío fue inmediato



 DemoBatalla.java

codigo de lab 4 terminado

2 minutes ago

Así quedó en mi repositorio de GitHub.

## CONCLUSIONES

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 15

*Los métodos para ordenar los Arrays son bastante útiles, requiere unas líneas más al tratarse arreglos de objetos, pero siguen funcionando sin problema. Además, bastantes versátiles, pueden ser usados en todos los tipos de listas.*



### METODOLOGÍA DE TRABAJO

*Usé las mismas que fui usando durante estos laboratorios, comentar bloques de código para poder concentrarme en una parte y revisando problemas pasados y similares que ya resolví para tener una idea y construir un nuevo método.*

### REFERENCIAS Y BIBLIOGRAFÍA

*E. G. Castro Gutiérrez and M. W. Aedo López, Fundamentos de programación 2: tópicos de programación orientada a objetos, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021, pp. 170, ISBN 978-612-5035-20-2.*

***RUBRICA DE CALIFICACIÓN EN LABORATORIOS***  
***(SIGUIENTE PÁGINA)***

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p style="text-align: center;"><b>Código:</b> GUIA-PRLE-001</p>	<p style="text-align: right;"><b>Página:</b> 16</p>

Contenido y demostración		Puntos	<u>Checklis</u> <u>t</u>	Estudiant e	Profeso r
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. <u>Commits</u>	Hay capturas de pantalla de los <u>commits</u> más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	x	4	
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		0	
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	2	
TOTAL		20		16	