
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
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					
ASIGNATURA:	<i>Fundamentos de la programación 02</i>				
TÍTULO DE LA PRÁCTICA:					
NÚMERO DE PRÁCTICA:	<i>04</i>	AÑO LECTIVO:	<i>2024-B</i>	NRO. SEMESTRE:	<i>//</i>
FECHA DE PRESENTACIÓN	<i>13/10/2024</i>	HORA DE PRESENTACIÓN	<i>00:15:30</i>		
INTEGRANTE (s) <i>Mauro Snayder Sullca Mamani</i>				NOTA (0-20)	
DOCENTE(s): <i>Ing. Lino Jose Pinto Oppe</i>					

RESULTADOS Y PRUEBAS	
I. EJERCICIOS RESUELTOS:	
<pre> 5 package laboratorio_04; 6 7 /** 8 * 9 * @author Usuario24B 10 */ 11 public class Nave { 12 // Creamos los atributos 13 private String nombre; 14 private int fila; 15 private String columna; 16 private boolean estado; 17 private int puntos; 18 19 // Creamos los Set y Get de cada atributo 20 public String getNombre() { 21 return nombre; 22 } 23 24 public void setNombre(String nombre) { 25 this.nombre = nombre; 26 } 27 28 public int getFila() { 29 return fila; 30 } 31 32 public void setFila(int fila) { 33 this.fila = fila; 34 } 35 36 public String getColumna() { 37 return columna; 38 } 39 40 public void setColumna(String columna) { 41 this.columna = columna; 42 } 43 44 public boolean isEstado() { 45 return estado; 46 } 47 48 public void setEstado(boolean estado) { 49 this.estado = estado; 50 } </pre>	

```

51     public int getPuntos() {
52         return puntos;
53     }
54
55     public void setPuntos(int puntos) {
56         this.puntos = puntos;
57     }
58
59     // Creamos el toString y un metodo que imprimirá el nombre y puntos
60     public String toString() {
61         return "Nave(" + "nombre=" + nombre + ", fila=" + fila + ", columna=" + columna + ", estado=" + estado + ", puntos=" + puntos + ')';
62     }
63
64     public String navePunto() {
65         return "Nave(" + "nombre=" + nombre + ", puntos=" + puntos + ')';
66     }
67
68 }

```

```

5 package laboratorio_04;
6
7 /**
8  *
9  * @author Usuario24B
10  */
11 import java.util.*;
12 public class DemoBatalla {
13
14     public static void main(String [] args){
15         Nave[] misNaves=new Nave[10]; //creamos un arreglo para 10 objetos tipo nave
16         Scanner scan=new Scanner(System.in);
17         String nomb,col;
18         int fil,punt;
19         boolean est;
20         for (int i=0;i<misNaves.length;i++){ // Ingresamos los datos para cada nave
21             System.out.println("Nave " + (i+1));
22             System.out.print("Nombre: ");
23             nomb = scan.next();
24             System.out.print("Fila: ");
25             fil = scan.nextInt();
26             System.out.print("Columna: ");
27             col = scan.nextInt();
28             System.out.print("Estado (true,false): ");
29             est = scan.nextBoolean();
30             System.out.print("Puntos: ");
31             punt = scan.nextInt();
32
33             misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
34
35             misNaves[i].setNombre(nomb);
36             misNaves[i].setFila(fil);
37             misNaves[i].setColumna(col);
38             misNaves[i].setEstado(est);
39             misNaves[i].setPuntos(punt);
40         }
41         System.out.println("\nNaves creadas:");
42         mostrarNaves(misNaves);
43         mostrarPorNombre(misNaves,scan);
44         mostrarPorPuntos(misNaves,scan);
45         System.out.println("\nNave con mayor numero de puntos: " + mostrarMayorPuntos(misNaves));
46
47         //leer un nombre
48         //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
49         System.out.println("Ingrese el nombre de la nave que desea buscar: ");
50         String buscar=scan.next();
51         int pos=busquedaLinealNombre(misNaves,buscar);
52         int pos=busquedaLinealNombre(misNaves,buscar);
53         if (pos!=-1)
54             System.out.println(misNaves[pos].toString());
55         else
56             System.out.println("No encontrado");
57
58         ordenarPorPuntosBurbuja(misNaves);
59         mostrarNaves(misNaves);
60         ordenarPorNombreBurbuja(misNaves);
61         mostrarNaves(misNaves);
62
63         //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
64         System.out.println("Ingrese el nombre de la nave que desea buscar: ");
65         buscar=scan.next();
66         pos=busquedaBinariaNombre(misNaves,buscar);
67         if (pos!=-1)
68             System.out.println(misNaves[pos].toString());
69         else
70             System.out.println("No encontrado");

```

```



71 ordenarPorPuntosSeleccion(misNaves);
72 mostrarNaves(misNaves);
73 ordenarPorNombreSeleccion(misNaves);
74 mostrarNaves(misNaves);
75 ordenarPorPuntosInsercion(misNaves);
76 mostrarNaves(misNaves);
77 ordenarPorNombreInsercion(misNaves);
78 mostrarNaves(misNaves);
79 }
80
81 //Método para mostrar todas las naves
82 public static void mostrarNaves(Nave [] flota){
83     for (int i=0;i<flota.length;i++){
84         System.out.println(flota[i].toString());
85     }
86 }
87
88 //Método que muestra todas las naves de un nombre que se pide por teclado
89 public static void mostrarPorNombre(Nave [] flota,Scanner scan){
90     System.out.println("Ingrese el nombre de su nave: ");
91     String nombre=scan.next();
92     for (int i=0;i<flota.length;i++){
93         if (flota[i].getNombre().equals(nombre)) // Si los nombres son iguales, se imprime los datos de la nave
94             System.out.println(flota[i].toString());
95     }
96 }
97
98 //Método que muestra todas las naves con un número de puntos inferior o igual
99 //al número de puntos que se pide por teclado
100 public static void mostrarPorPuntos(Nave [] flota,Scanner scan){
101     System.out.println("Ingrese un numero de puntos: ");
102     int numero=scan.nextInt();
103     for (int i=0;i<flota.length;i++){
104         if (flota[i].getPuntos()<=numero) //Si la nave tiene menos puntos que "numero", entonces se imprime
105             System.out.println(flota[i].navePunto());
106     }
107 }
108
109 //Método que devuelve la Nave con mayor número de Puntos
110 public static Nave mostrarMayorPuntos(Nave [] flota){
111     Nave mayor=flota[0]; //Definimos temporalmente a la nave mayor
112     for (int i=1;i<flota.length;i++){
113         if (flota[i].getPuntos()>mayor.getPuntos()) //Si existen una nave con mayor puntos
114             mayor=flota[i]; //entonces se actualiza la variable mayor
115     }
116     return mayor;
117 }
118
119 //Método para buscar la primera nave con un nombre que se pidió por teclado
120 public static int busquedaLinealNombre(Nave[] flota, String s){
121     for (int i=0;i<flota.length;i++){
122         if (flota[i].getNombre().equals(s)) //Busca el nombre dentro del arreglo
123             return i; //si lo encuentra, retorna la posicion
124     }
125     return -1; //si no lo encuentra retorna -1
126 }
127
128 //Método que ordena por número de puntos de menor a mayor
129 public static void ordenarPorPuntosBurbuja(Nave[] flota){
130     Nave cambio;
131     for (int i=0;i<flota.length-1;i++){ //numero de repeticiones
132         for (int j=0;j<flota.length-1-i;j++){ //compara dos posiciones que son adyacentes
133             if (flota[j].getPuntos()>flota[j+1].getPuntos()) //si la posicion actual es mayor a la posicion siguiente
134                 cambio=flota[j]; //entonces se cambian las posiciones
135                 flota[j]=flota[j+1];
136                 flota[j+1]=cambio;
137             }
138         }
139     }
140 }
141
142 //Método que ordena por nombre de A a Z
143 public static void ordenarPorNombreBurbuja(Nave[] flota){
144     Nave cambio;
145     for (int i=0;i<flota.length-1;i++){ //numero de repeticiones
146         for (int j=0;j<flota.length-1-i;j++){ //compara dos posiciones que son adyacentes
147             if (flota[j].getNombre().compareTo(flota[j+1].getNombre())>0) //si la posicion actual es mayor a la posicion siguiente
148                 cambio=flota[j]; //entonces se cambian las posiciones
149                 flota[j]=flota[j+1];
150                 flota[j+1]=cambio;
151             }
152         }
153     }
154 }
155 }
156
157

```

```

158 //Método para buscar la primera nave con un nombre que se pidió por teclado
159 public static int busquedaBinariaNombre(Nave[] flota, String s){
160     int alta,media,baja=0;
161     alta=flota.length-1; //definimos las posiciones para alta y baja
162     while (baja<=alta){
163         media=(alta+baja)/2; //definimos la posicion de media
164         if (flota[media].getNombre().equals(s))
165             return media; //si el nombre "s" se encuentra en la mitad del arreglo, retornamos "media"
166         else if (s.compareTo(flota[media].getNombre())<0)
167             alta=media-1; //si el nombre "s" se encuentra antes de "media", entonces "alta" cambia de valor
168         else
169             baja=media+1; //caso contrario, "baja" cambiaria de valor
170     }
171     return -1; //si no lo encuentra retorna -1
172 }
173
174 //Método que ordena por número de puntos de menor a mayor
175 public static void ordenarPorPuntosSeleccion(Nave[] flota){
176     Nave cambio;
177     int posMenor;
178     for (int i=0;i<flota.length-1;i++){ //numero de repeticiones
179         posMenor=posicionPuntosMenor(i,flota); //encuentra la posicion del menor puntaje
180         if (flota[i].getPuntos()>flota[posMenor].getPuntos()){ //comparamos la posicion "i" con el "posMenor"
181             cambio=flota[i]; //para ver si se intercambian o no
182             flota[i]=flota[posMenor];
183             flota[posMenor]=cambio;
184         }
185     }
186 }
187
188 //Método que ordena por nombre de A a Z
189 public static void ordenarPorNombreSeleccion(Nave[] flota){
190     Nave cambio;
191     int posMenor;
192     for (int i=0;i<flota.length-1;i++){ //numero de repeticiones
193         posMenor=posicionNombreMenor(i,flota); //encuentra la posicion del menor puntaje
194         if (flota[i].getNombre().compareTo(flota[posMenor].getNombre())>0){ //comparamos la posicion "i" con el "posMenor"
195             cambio=flota[i]; //para ver si se intercambian o no
196             flota[i]=flota[posMenor];
197             flota[posMenor]=cambio;
198         }
199     }
200 }
201
202 //Método que muestra las naves ordenadas por número de puntos de mayor a menor
203 public static void ordenarPorPuntosInsercion(Nave[] flota){
204     for (int i=1;i<flota.length;i++){ // Recorre el arreglo desde el segundo elemento hasta el final.
205         Nave naveComparacion=flota[i];
206         int puntoComparacion=flota[i].getPuntos(); //obtenemos los puntos de la nave actual
207         int j;
208         for (j=i-1;j>=0 && flota[j].getPuntos()>puntoComparacion;j--){
209             flota[j+1] = flota[j]; // Desplaza las naves que tienen más puntos a la derecha.
210             flota[j+1]=naveComparacion; //cuando de acaba el bucle, se inserta la nave actual a su nueva posicion
211         }
212     }
213 }
214
215 //Método que muestra las naves ordenadas por nombre de Z a A
216 public static void ordenarPorNombreInsercion(Nave[] flota){
217     for (int i=1;i<flota.length;i++){ // Recorre el arreglo desde el segundo elemento hasta el final.
218         Nave naveComparacion=flota[i]; //definimos la nave actual para comparar
219         String nombreComparacion=flota[i].getNombre(); //obtenemos el nombre de la nave actual
220         int j;
221         for (j=i-1;j>=0 && flota[j].getNombre().compareTo(nombreComparacion)>0;j--){
222             flota[j+1]=flota[j]; // Desplaza las naves que tienen un orden alfabetico mas cercano a "z" a la derecha.
223             flota[j+1]=naveComparacion; //cuando de acaba el bucle, se inserta la nave actual a su nueva posicion
224         }
225     }
226 }
227
228 //Metodo que halla la posicion de la nave con menor puntaje
229 public static int posicionPuntosMenor(int inicio,Nave[] flota){
230     int posMenor=inicio;
231     for (int i=inicio+1;i<flota.length;i++){
232         if (flota[posMenor].getPuntos()>flota[i].getPuntos())
233             posMenor=i;
234     }
235     return posMenor;
236 }
237
238 //Metodo que halla la posicion de la nave con un nombre cercano a "Z"
239 public static int posicionNombreMenor(int inicio,Nave[] flota){
240     int posMenor=inicio;
241     for (int i=inicio+1;i<flota.length;i++){
242         if (flota[posMenor].getNombre().compareTo(flota[i].getNombre())>0)
243             posMenor=i;
244     }
245     return posMenor;
246 }

```

	<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;">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: 5</p>

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

Para comprobar que la práctica estuviera correcta, se utilizaron diferentes valores de entrada que incluían nombres y puntajes variados para las naves. Estos datos permitieron evaluar la funcionalidad de los métodos de ordenación y búsqueda, asegurando que el código pudiera manejar diversas entradas adecuadamente y produjera resultados coherentes al interactuar con los objetos de tipo Nave.

¿Qué resultado esperabas obtener para cada valor de entrada?

Se esperaba que, al ordenar las naves por nombre, el resultado fuera alfabéticamente correcto, y que al ordenarlas por puntajes, se presentaran de menor a mayor. Además, se anticipó que la búsqueda de naves específicas devolviera los datos correctos, mientras que la búsqueda de un nombre inexistente debería resultar en un mensaje de "No encontrado". También se esperaba que los filtros funcionaran correctamente, mostrando las naves con puntajes inferiores o iguales al valor ingresado.

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

Al realizar las pruebas, se confirmaron los resultados esperados en su mayoría, ya que las naves se ordenaron correctamente por nombre y las búsquedas devolvieron la información adecuada. Sin embargo, se identificaron errores en el método de ordenación por puntajes, lo que afectó la salida final en esa categoría. En general, aunque algunas funcionalidades operaron correctamente, se necesitaron ajustes en ciertas partes del código para lograr el comportamiento deseado en todas las pruebas.

En esta ejecución estamos viendo el ingreso de datos (nombre, fila, columna, estado y puntos) de 10 naves

```

Output - Laboratorios (run)
Nombre: nave2
Fila: 3
Columna: 5
Estado (true,false): false
Puntos: 12
Nave 3
Nombre: nave3
Fila: 8
Columna: 4
Estado (true,false): true
Puntos: 9
Nave 4
Nombre: nave4
Fila: 5
Columna: 2
Estado (true,false): true
Puntos: 19
Nave 5
Nombre: nave5
Fila: 9
Columna: 5
Estado (true,false): true
Puntos: 17
Nave 6
Nombre: nave6
Fila: 8
Columna: 4
Estado (true,false): false
Puntos: 14

Nave 7
Nombre: nave7
Fila: 7
Columna: 4
Estado (true,false): true
Puntos: 20
Nave 8
Nombre: nave8
Fila: 3
Columna: 7
Estado (true,false): false
Puntos: 16
Nave 9
Nombre: nave9
Fila: 3
Columna: 5
Estado (true,false): true
Puntos: 11
Nave 10
Nombre: nave10
Fila: 4
Columna: 8
Estado (true,false): false
Puntos: 7
  
```

En esta imagen estamos viendo los datos de todas las naves que hemos ingresado, además la búsqueda de una nave en específico y las naves con un puntaje menor a 15.

```

Output - Laboratorios (run)
Naves creadas:
Nave[nombre=nave1, fila=3, columna=4, estado=true, puntos=15]
Nave[nombre=nave2, fila=3, columna=5, estado=false, puntos=12]
Nave[nombre=nave3, fila=8, columna=4, estado=true, puntos=9]
Nave[nombre=nave4, fila=5, columna=2, estado=true, puntos=19]
Nave[nombre=nave5, fila=9, columna=5, estado=true, puntos=17]
Nave[nombre=nave6, fila=8, columna=4, estado=false, puntos=14]
Nave[nombre=nave7, fila=7, columna=4, estado=true, puntos=20]
Nave[nombre=nave8, fila=3, columna=7, estado=false, puntos=16]
Nave[nombre=nave9, fila=3, columna=5, estado=true, puntos=11]
Nave[nombre=nave10, fila=4, columna=8, estado=false, puntos=7]
Ingrese el nombre de su nave:
nave6
Nave[nombre=nave6, fila=8, columna=4, estado=false, puntos=14]
Ingrese un numero de puntos:
15
Nave[nombre=nave1, puntos=15]
Nave[nombre=nave2, puntos=12]
Nave[nombre=nave3, puntos=9]
Nave[nombre=nave6, puntos=14]
Nave[nombre=nave9, puntos=11]
Nave[nombre=nave10, puntos=7]
  
```

En esta siguiente imagen estamos viendo la nave con mayor puntaje, la búsqueda lineal de una nave en específico y el ordenamiento Burbuja por puntos y nombres de las naves.

```

Nave con mayor numero de puntos: Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Ingrese el nombre de la nave que desea buscar:
nave2
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
  
```

Ordenamiento por puntos

Ordenamiento por nombre

En esta ultima imagen estamos viendo la búsqueda binaria de una nave en específico y el ordenamiento tanto por puntos como por nombres con los métodos de Selección e inserción.

```

Ingrese el nombre de la nave que desea buscar:
naves
No encontrado
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave1, fila=3, columna=4, estado=true, puntos=15)
Nave(nombre=nave10, fila=4, columna=8, estado=false, puntos=7)
Nave(nombre=nave2, fila=3, columna=5, estado=false, puntos=12)
Nave(nombre=nave3, fila=8, columna=4, estado=true, puntos=9)
Nave(nombre=nave4, fila=5, columna=2, estado=true, puntos=19)
Nave(nombre=nave5, fila=9, columna=5, estado=true, puntos=17)
Nave(nombre=nave6, fila=8, columna=4, estado=false, puntos=14)
Nave(nombre=nave7, fila=7, columna=4, estado=true, puntos=20)
Nave(nombre=nave8, fila=3, columna=7, estado=false, puntos=16)
Nave(nombre=nave9, fila=3, columna=5, estado=true, puntos=11)
BUILD SUCCESSFUL (total time: 8 minutes 40 seconds)
  
```



Ordenamiento por puntos

Ordenamiento por nombre

Ordenamiento por puntos

Ordenamiento por nombre

Output

	<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;">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: 8</p>

III. COMMITS:

Entramos a nuestra carpeta donde están nuestros archivos y añadimos los cambios.

```

MINGW64/c/Users/Mauro Snayder/Documents/NetBeansProjects/Laboratorios/src
Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Laboratorio_04/DemoBatalla.java
        modified:   Laboratorio_04/Nave.java

no changes added to commit (use "git add" and/or "git commit -a")

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git add .

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Laboratorio_04/DemoBatalla.java
        modified:   Laboratorio_04/Nave.java

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$

```

Hacemos un commit

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git commit -m "culminado"
[master 0afc6f6] culminado
 2 files changed, 34 insertions(+), 24 deletions(-)

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)

```

Subimos nuestro commit a nuestro repositorio remoto

```

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)
$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 909 bytes | 909.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/MauroSullcaMamani/FDLP2_LAB.git
   7e67647..0afc6f6  master -> master
branch 'master' set up to track 'origin/master'.

Mauro Snayder@Mauro MINGW64 ~/Documents/NetBeansProjects/Laboratorios/src (master)

```

Link de mi repositorio: https://github.com/MauroSullcaMamani/FDLP2_LAB.git



RUBRICA:

Contenido y demostración		Puntos	Checklis t	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	✓	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	✓	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	✓	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	✓	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	✓	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	✓	3	
TOTAL		20		18	

Tabla 2: Rúbrica para contenido del Informe y demostración

CONCLUSIONES

En conclusión, la utilización de arreglos de objetos, junto con técnicas de búsqueda y ordenamiento, permite manejar colecciones de datos de manera eficiente y organizada. Al trabajar con objetos, se pueden gestionar múltiples atributos de manera estructurada, lo que facilita la manipulación y el acceso a la información. Los algoritmos de búsqueda permiten encontrar elementos específicos rápidamente, mientras que los algoritmos de ordenamiento ayudan a organizar los datos según distintos criterios, mejorando la accesibilidad y optimizando el rendimiento de las aplicaciones.

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

METODOLOGÍA DE TRABAJO

Lo primero que hice, es leer cada los enunciados de cada ejercicio y también tomar en cuenta las restricciones que nos da para así poder buscar una solución al problema. Después observar el código y entender la funcionalidad de cada uno y completar las partes que están incompletas. Y por último comprobar nuestro código ingresando varias veces valores de prueba para ver que nuestro código está funcionando correctamente.

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.