

Informe de Laboratorio 04

Tema: Arreglos de Objetos, Búsqueda y Ordenamientos

Nota

Estudiante	Escuela	Asignatura
Christian Mestas Zegarra cmestasz@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Fundamentos de la Programación 2 Semestre: II Código: 1701213

Laboratorio	Tema	Duración
04	Arreglos de Objetos, Búsqueda y Ordenamientos	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 20 Setiembre 2023	Al 25 Setiembre 2023

1. Tarea

- **Actividad 1:** Completar el código de la clase DemoBatalla. Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. Creadas en Laboratorio 3

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.
- Búsqueda lineal y binaria.
- Ordenamientos burbuja, por selección y por inserción.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/cmestasz/fp2-23b.git>
- URL para el laboratorio 01 en el Repositorio GitHub.
- <https://github.com/cmestasz/fp2-23b/tree/main/fase01/lab01>

4. Actividades con el repositorio GitHub

Creando plantillas

```
$ mkdir lab04  
$ cd lab04  
$ code Nave.java  
$ code DemoBatalla.java
```

Primer Commit / Plantillas

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

Showing 2 changed files with 165 additions and 0 deletions.

Filter changed files

fase01/lab04
DemoBatalla.java
Nave.java

```

5 + import java.util.*;
6 +
7 + public class DemoBatalla {
8 +     public static void main(String[] args) {
9 +         Nave[] misNaves = new Nave[10];
10 +         Scanner sc = new Scanner(System.in);
11 +         String nomb, col;
12 +         int fil, punt;
13 +         boolean est;
14 +         for (int i = 0; i < misNaves.length; i++) {
15 +             System.out.println("Nave " + (i + 1));
16 +             System.out.print("Nombre: ");
17 +             nomb = sc.next();
18 +             System.out.print("Fila: ");
19 +             fil = sc.nextInt();
20 +             System.out.print("Columna: ");
21 +             col = sc.next();
22 +             System.out.print("Estado: ");
23 +             est = sc.nextBoolean();
24 +             System.out.print("Puntos: ");
25 +             punt = sc.nextInt();
26 +             misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a misNaves
27 +             misNaves[i].setNombre(nomb);
28 +             misNaves[i].setFila(fil);
29 +             misNaves[i].setColumna(col);
30 +             misNaves[i].setEstado(est);
31 +             misNaves[i].setPuntos(punt);
32 +             System.out.println();
33 +         }
34 +         System.out.println("Naves creadas:");
35 +         mostrarNaves(misNaves);
36 +         mostrarPorNombre(misNaves);
37 +         mostrarPorPuntos(misNaves);
38 +         System.out.println("Nave con mayor número de puntos: " + mostrarMayorPuntos(misNaves));
39 +         System.out.println();
40 +         System.out.println("Naves desordenadas: ");
41 +         mostrarNaves(desordenar(misNaves));
42 +     }
43 +
44 +     // Método para mostrar todas las naves
45 +     public static void mostrarNaves(Nave[] flota) {
46 +         for (Nave nave : flota) {
47 +             System.out.println(nave.toString());
48 +         }
49 +     }

```

Primer Commit.

Actualizando DemoBatalla.java

```
$ code DemoBatalla.java
```

Segundo - Noveno Commit / DemoBatalla.java

```

$ git add .
$ git commit -m "Metodo busquedaLinealNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo busquedaBinariaNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosBurbuja()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorNombreBurbuja()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosSeleccion()"
$ git push
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorNombreSeleccion()"
$ git push
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosInsercion()"
$ git push
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorNombreInsercion()"
$ git push

```

Showing 1 changed file with 70 additions and 17 deletions.

<pre> 89 - // Crear un método que devuelva un nuevo arreglo de objetos con todos los 90 - // objetos previamente ingresados 91 - // pero aleatoriamente desordenados 92 - public static Nave[] desordenar(Nave[] flota) { 93 - Nave[] nuevaFlota = new Nave[flota.length]; 94 - Random r = new Random(); 95 - System.arraycopy(flota, 0, nuevaFlota, 0, flota.length); 96 - for (int idx = 0; idx < nuevaFlota.length; idx++) { 97 - int nIdx = r.nextInt(nuevaFlota.length); 98 - Nave t = nuevaFlota[idx]; 99 - nuevaFlota[idx] = nuevaFlota[nIdx]; 100 - nuevaFlota[nIdx] = t; 101 } 102 - return nuevaFlota; </pre>	<pre> 113 + // Método para buscar la primera nave con un nombre que se pidió por teclado 114 + public static int busquedaLinealNombre(Nave[] flota, String s) { 115 + for (int i = 0; i < flota.length; i++) { 116 + if (flota[i].getNombre().equals(s)) { 117 + return i; 118 + } 119 } 120 + return -1; 121 + } </pre>
--	---

Segundo Commit.

Showing 1 changed file with 25 additions and 5 deletions.

<pre> 131 // Método para buscar la primera nave con un nombre que se pidió por teclado 132 public static int busquedaLinealNombre(Nave[] flota, String s) { 133 - 134 - 135 - 136 } </pre>	<pre> 142 // Método para buscar la primera nave con un nombre que se pidió por teclado 143 public static int busquedaLinealNombre(Nave[] flota, String s) { 144 + int baja = 0, alta = flota.length - 1, media; 145 + while (baja <= alta) { 146 + media = (baja + alta) / 2; 147 + String nombre = flota[media].getNombre(); 148 + if (nombre.equals(s)) { 149 + return media; 150 + } 151 + if (nombre.compareTo(s) < 0) { 152 + baja = media + 1; 153 + } 154 + else { 155 + alta = media - 1; 156 + } 157 + } 158 + return -1; 159 } </pre>
--	---

Tercer Commit.

Showing 1 changed file with 24 additions and 6 deletions.

<pre> 131 @@ -131,7 +138,12 @@ public static int busquedaLinealNombre(Nave[] flota, String s) { 132 133 // Método que ordena por número de puntos de menor a mayor 134 public static void ordenarPorPuntosBurbuja(Nave[] flota) { 135 - 136 - 137 // Método que ordena por nombre de A a Z 138 @@ -174,4 +186,10 @@ public static void ordenarPorPuntosInsercion(Nave[] flota) { 174 public static void ordenarPorPuntosInsercion(Nave[] flota) { 175 - 176 - 177 } </pre>	<pre> 138 // Método que ordena por número de puntos de menor a mayor 139 public static void ordenarPorPuntosBurbuja(Nave[] flota) { 140 + for (int i = 0; i < flota.length - 1; i++) { 141 + for (int j = 0; j < flota.length - i - 1; j++) { 142 + if (flota[j].getPuntos() > flota[j + 1].getPuntos()) { 143 + intercambiar(flota, j, j + 1); 144 + } 145 + } 146 + } 147 } 148 149 // Método que ordena por nombre de A a Z 150 151 public static void ordenarPorNombreInsercion(Nave[] flota) { 152 - 153 - 154 - 155 } </pre>
--	---

Cuarto Commit.

Showing 1 changed file with 7 additions and 3 deletions.

<pre> 147 // 148 @@ -148,7 +147,12 @@ public static void ordenarPorPuntosBurbuja(Nave[] flota) { 149 150 // Método que ordena por nombre de A a Z 151 public static void ordenarPorNombreBurbuja(Nave[] flota) { 152 - 153 - 154 - 155 } </pre>	<pre> 147 // 148 // Método que ordena por nombre de A a Z 149 public static void ordenarPorNombreBurbuja(Nave[] flota) { 150 + for (int i = 0; i < flota.length - 1; i++) { 151 + for (int j = 0; j < flota.length - i - 1; j++) { 152 + if (flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0) { 153 + intercambiar(flota, j, j + 1); 154 + } 155 + } 156 + } 157 } </pre>
---	---

Quinto Commit.

Showing 1 changed file with 11 additions and 4 deletions.

File	Line	Code
fase01/1a04/DemoBatalla.java	175	// Método que ordena por número de puntos de menor a mayor
	176	public static void ordenarPorPuntosSeleccion(Nave[] flota) {
	177	for (int i = 0; i < flota.length - 1; i++) {
	178	int idx = i;
	179	for (int j = i + 1; j < flota.length; j++) {
	180	if (flota[j].getPuntos() < flota[idx].getPuntos())
	181	idx = j;
	182	intercambiar(flota, i, idx);
	183	}
	184	}
	185	}

Sexto Commit.

Showing 1 changed file with 8 additions and 1 deletion.

File	Line	Code
fase01/1a04/DemoBatalla.java	185	// Método que ordena por nombre de A a Z
	186	public static void ordenarPorNombreSeleccion(Nave[] flota) {
	187	for (int i = 0; i < flota.length - 1; i++) {
	188	int idx = i;
	189	for (int j = i + 1; j < flota.length; j++) {
	190	if (flota[j].getNombre().compareTo(flota[idx].getNombre()) < 0)
	191	idx = j;
	192	intercambiar(flota, i, idx);
	193	}
	194	}
	195	}

Séptimo Commit.

Showing 1 changed file with 6 additions and 1 deletion.

File	Line	Code
fase01/1a04/DemoBatalla.java	197	// Método que muestra las naves ordenadas por número de puntos de mayor a menor
	198	public static void ordenarPorPuntosInsercion(Nave[] flota) {
	199	for (int i = 1; i < flota.length; i++) {
	200	int j = i;
	201	while (j > 0 && flota[j - 1].getPuntos() > flota[j].getPuntos()) {
	202	intercambiar(flota, j - 1, j);
	203	j--;
	204	}
	205	}
	206	}
	207	}

Octavo Commit.

Showing 1 changed file with 11 additions and 4 deletions.

File	Line	Code
fase01/1a04/DemoBatalla.java	206	// Método que muestra las naves ordenadas por nombre de Z a A
	207	public static void ordenarPorNombreInsercion(Nave[] flota) {
	208	for (int i = 1; i < flota.length; i++) {
	209	int j = i;
	210	while (j > 0 && flota[j - 1].getNombre().compareTo(flota[j].getNombre()) < 0) {
	211	intercambiar(flota, j - 1, j);
	212	j--;
	213	}
	214	}
	215	}
	216	}

Noveno Commit.

5. Código desarrollado

Nave.java

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

```
53     }  
54  
55     public String toString() {  
56         return (nombre + ": (" + fila + ", " + columna + "). Estado: " + estado + ". Puntos:  
57             " + puntos);  
58     }  
59     // Completar con otros metodos necesarios  
}
```

- 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

```
1 // Laboratorio Nro 4 - Actividad 1
2 // Autor: Christian Mestas
3 // Colaboro: Marco Aedo, clases DemoBatalla y Nave
4
5 import java.util.*;
6
7 public class DemoBatalla {
8     public static void main(String[] args) {
9         Nave[] misNaves = new Nave[10];
10        Scanner sc = new Scanner(System.in);
11        String nomb, col;
12        int fil, punt;
13        boolean est;
14        for (int i = 0; i < misNaves.length; i++) {
15            System.out.println("Nave " + (i + 1));
16            System.out.print("Nombre: ");
17            nomb = sc.nextLine();
18            System.out.print("Fila: ");
19            fil = sc.nextInt();
20            sc.nextLine();
21            System.out.print("Columna: ");
22            col = sc.nextLine();
23            System.out.print("Estado: ");
24            est = sc.nextBoolean();
25            sc.nextLine();
26            System.out.print("Puntos: ");
27            punt = sc.nextInt();
28            sc.nextLine();
29            misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
30                                   misNaves
31            misNaves[i].setNombre(nomb);
32            misNaves[i].setFila(fil);
33            misNaves[i].setColumna(col);
34            misNaves[i].setEstado(est);
35            misNaves[i].setPuntos(punt);
36            System.out.println();
37        }
38        System.out.println("Naves creadas:");
39        mostrarNaves(misNaves);
40        mostrarPorNombre(misNaves);
41        mostrarPorPuntos(misNaves);
42        System.out.println("Nave con mayor numero de puntos: " + mostrarMayorPuntos(misNaves)
43                           + "\n");
44
45        // leer un nombre
46        // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
47        // caso contrario
48        System.out.println("Ingrese el nombre a buscar:");
49        String nombre = sc.nextLine();
50        int pos = busquedaLinealNombre(misNaves, nombre);
51        if (pos == -1)
52            System.out.println("Nave no encontrada.\n");
53        else
54            System.out.println("Nave encontrada: " + misNaves[pos] + "\n");
55        System.out.println("Naves ordenadas por puntos:");
```



```
54     ordenarPorPuntosBurbuja(misNaves);
55     mostrarNaves(misNaves);
56     System.out.println("Naves ordenadas por nombre:");
57     ordenarPorNombreBurbuja(misNaves);
58     mostrarNaves(misNaves);
59
60     // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
61     // caso contrario
62     System.out.println("Ingrese el nombre a buscar:");
63     nombre = sc.nextLine();
64     pos = busquedaBinariaNombre(misNaves, nombre);
65     if (pos == -1)
66         System.out.println("Nave no encontrada.\n");
67     else
68         System.out.println("Nave encontrada: " + misNaves[pos] + "\n");
69     System.out.println("Naves ordenadas por puntos:");
70     ordenarPorPuntosSeleccion(misNaves);
71     mostrarNaves(misNaves);
72     System.out.println("Naves ordenadas por nombre:");
73     ordenarPorNombreSeleccion(misNaves);
74     mostrarNaves(misNaves);
75     System.out.println("Naves ordenadas por puntos (Invertido):");
76     ordenarPorPuntosInsercion(misNaves);
77     mostrarNaves(misNaves);
78     System.out.println("Naves ordenadas por nombre (Invertido):");
79     ordenarPorNombreInsercion(misNaves);
80     mostrarNaves(misNaves);
81 }
82
83 // Metodo para mostrar todas las naves
84 public static void mostrarNaves(Nave[] flota) {
85     for (Nave nave : flota)
86         System.out.println(nave.toString());
87     System.out.println();
88 }
89
90 // Metodo para mostrar todas las naves de un nombre que se pide por teclado
91 public static void mostrarPorNombre(Nave[] flota) {
92     Scanner sc = new Scanner(System.in);
93     System.out.println("Ingrese el nombre a buscar:");
94     String nombre = sc.nextLine();
95     System.out.println("Naves con el nombre " + nombre + ":");
96     for (Nave nave : flota) {
97         if (nave.getNombre().equals(nombre))
98             System.out.println(nave.toString());
99     }
100     System.out.println();
101 }
102
103 // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
104 // al numero de puntos que se pide por teclado
105 public static void mostrarPorPuntos(Nave[] flota) {
106     Scanner sc = new Scanner(System.in);
107     System.out.println("Ingrese la cantidad de puntos maximos:");
108     int puntos = sc.nextInt();
109     sc.nextLine();
```

```
110     System.out.println("Naves con " + puntos + " puntos como maximo:");
111     for (Nave nave : flota) {
112         if (nave.getPuntos() <= puntos)
113             System.out.println(nave.toString());
114     }
115     System.out.println();
116 }
117
118 // Metodo que devuelve la Nave con mayor numero de Puntos
119 public static Nave mostrarMayorPuntos(Nave[] flota) {
120     int maxIdx = 0;
121     for (int i = 0; i < flota.length; i++) {
122         if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
123             maxIdx = i;
124     }
125     return flota[maxIdx];
126 }
127
128 // Metodo para buscar la primera nave con un nombre que se pidio por teclado
129 public static int busquedaLinealNombre(Nave[] flota, String s) {
130     for (int i = 0; i < flota.length; i++) {
131         if (flota[i].getNombre().equals(s)) {
132             return i;
133         }
134     }
135     return -1;
136 }
137
138 // Metodo que ordena por numero de puntos de menor a mayor
139 public static void ordenarPorPuntosBurbuja(Nave[] flota) {
140     for (int i = 0; i < flota.length - 1; i++) {
141         for (int j = 0; j < flota.length - i - 1; j++) {
142             if (flota[j].getPuntos() > flota[j + 1].getPuntos())
143                 intercambiar(flota, j, j + 1);
144         }
145     }
146 }
147
148 // Metodo que ordena por nombre de A a Z
149 public static void ordenarPorNombreBurbuja(Nave[] flota) {
150     for (int i = 0; i < flota.length - 1; i++) {
151         for (int j = 0; j < flota.length - i - 1; j++) {
152             if (flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0)
153                 intercambiar(flota, j, j + 1);
154         }
155     }
156 }
157
158 // Metodo para buscar la primera nave con un nombre que se pidio por teclado
159 public static int busquedaBinariaNombre(Nave[] flota, String s) {
160     int baja = 0, alta = flota.length - 1, media;
161     while (baja <= alta) {
162         media = (baja + alta) / 2;
163         String nombre = flota[media].getNombre();
164         if (nombre.equals(s))
165             return media;
```

```
166         if (nombre.compareTo(s) < 0)
167             baja = media + 1;
168         else
169             alta = media - 1;
170     }
171     return -1;
172 }
173
174 // Metodo que ordena por numero de puntos de menor a mayor
175 public static void ordenarPorPuntosSeleccion(Nave[] flota) {
176     for (int i = 0; i < flota.length - 1; i++) {
177         int idx = i;
178         for (int j = i + 1; j < flota.length; j++) {
179             if (flota[j].getPuntos() < flota[idx].getPuntos())
180                 idx = j;
181         }
182         intercambiar(flota, i, idx);
183     }
184 }
185
186 // Metodo que ordena por nombre de A a Z
187 public static void ordenarPorNombreSeleccion(Nave[] flota) {
188     for (int i = 0; i < flota.length - 1; i++) {
189         int idx = i;
190         for (int j = i + 1; j < flota.length; j++) {
191             if (flota[j].getNombre().compareTo(flota[idx].getNombre()) < 0)
192                 idx = j;
193         }
194         intercambiar(flota, i, idx);
195     }
196 }
197
198 // Metodo que muestra las naves ordenadas por numero de puntos de mayor a menor
199 public static void ordenarPorPuntosInsercion(Nave[] flota) {
200     for (int i = 1; i < flota.length; i++) {
201         int j = i;
202         while (j - 1 >= 0 && flota[j - 1].getPuntos() < flota[j].getPuntos()) {
203             intercambiar(flota, j - 1, j);
204             j--;
205         }
206     }
207 }
208
209 // Metodo que muestra las naves ordenadas por nombre de Z a A
210 public static void ordenarPorNombreInsercion(Nave[] flota) {
211     for (int i = 1; i < flota.length; i++) {
212         int j = i;
213         while (j - 1 >= 0 && flota[j - 1].getNombre().compareTo(flota[j].getNombre()) < 0)
214             {
215                 intercambiar(flota, j - 1, j);
216                 j--;
217             }
218     }
219
220     public static void intercambiar(Nave[] flota, int a, int b) {
```

```
221     Nave t = flota[a];
222     flota[a] = flota[b];
223     flota[b] = t;
224 }
225 }
```

- Métodos reutilizados del laboratorio 03: mostrarNaves(), mostrarPorNombre(), mostrarPorPuntos(), mostrarMayorPuntos().
- Método busquedaLinealNombre() busca una nave linealmente por nombre y retorna el índice si es encontrada o -1 si no.
- Método ordenarPorPuntosBurbuja() ordena las naves por puntos de menor a mayor, usando ordenamiento burbuja.
- Método ordenarPorNombreBurbuja() ordena las naves por nombre alfabéticamente, usando ordenamiento burbuja.
- Método busquedaBinariaNombre() busca una nave por búsqueda binaria por nombre y retorna el índice si es encontrada o -1 si no.
- Método ordenarPorPuntosSeleccion() ordena las naves por puntos de menor a mayor, usando ordenamiento selección.
- Método ordenarPorNombreSeleccion() ordena las naves por nombre alfabéticamente, usando ordenamiento selección.
- Método ordenarPorPuntosInsercion() ordena las naves por puntos de mayor a menor, usando ordenamiento inserción.
- Método ordenarPorNombreInsercion() ordena las naves por nombre alfabéticamente reverso, usando ordenamiento inserción.

6. Ejecución del código

```

Windows PowerShell
PS F:\Documents\UNSA\IS2\FP2\lab\fp2-23b\fase01\lab04> java DemoBatalla
Nave 1
Nombre: naveuno
Pila: 3
Columna: A
Estado: true
Puntos: 14

Nave 2
Nombre: navedos
Pila: 1
Columna: D
Estado: false
Puntos: 12

Nave 3
Nombre: navetres
Pila: 9
Columna: C
Estado: true
Puntos: 5

Nave 4
Nombre: navecuatro
Pila: 1
Columna: J
Estado: true
Puntos: 10

Nave 5
Nombre: navecinco
Pila: 3
Columna: I
Estado: false
Puntos: 7

Nave 6
Nombre: naveseis
Pila: 6
Columna: G
Estado: true
Puntos: 9

Nave 7
Nombre: naveseite
Pila: 3
Columna: F
Estado: true
Puntos: 15

Nave 8
Nombre: naveocho
Pila: 1
Columna: A
Estado: false
Puntos: 3

Nave 9
Nombre: navenueve
Pila: 6
Columna: E
Estado: false
Puntos: 11

Nave 10
Nombre: navediez
Pila: 4
Columna: F
Estado: false
Puntos: 5

```

DemoBatalla.java

```

Windows PowerShell
Ingrese el nombre a buscar:
naveseite
Nave encontrada: naveseite: (3, F), Estado: true, Puntos: 15

Naves ordenadas por puntos:
naveocho: (1, A), Estado: false, Puntos: 3
navetres: (9, C), Estado: true, Puntos: 5
navediez: (4, F), Estado: false, Puntos: 5
navecinco: (3, I), Estado: false, Puntos: 7
naveseis: (6, G), Estado: true, Puntos: 9
navecuatro: (1, J), Estado: true, Puntos: 10
navenueve: (6, E), Estado: false, Puntos: 11
navedos: (1, D), Estado: false, Puntos: 12
naveuno: (8, A), Estado: true, Puntos: 14
naveseite: (3, F), Estado: true, Puntos: 15

Naves ordenadas por nombre:
navecinco: (3, I), Estado: false, Puntos: 7
navecuatro: (1, J), Estado: true, Puntos: 10
navediez: (4, F), Estado: false, Puntos: 5
navedos: (1, D), Estado: false, Puntos: 12
navenueve: (6, E), Estado: false, Puntos: 11
naveocho: (1, A), Estado: false, Puntos: 3
naveseis: (6, G), Estado: true, Puntos: 9
naveseite: (3, F), Estado: true, Puntos: 15
navetres: (9, C), Estado: true, Puntos: 5
naveuno: (8, A), Estado: true, Puntos: 14

```

DemoBatalla.java

```
Windows PowerShell
Ingresar el nombre a buscar:
navecorce
Nave no encontrada.

Naves ordenadas por puntos:
navecho: (1, A), Estado: false, Puntos: 3
navele: (4, F), Estado: false, Puntos: 5
navetres: (9, C), Estado: true, Puntos: 5
navecinco: (5, D), Estado: false, Puntos: 7
naveis: (5, G), Estado: true, Puntos: 9
navecuatro: (1, J), Estado: true, Puntos: 10
naveve: (6, E), Estado: false, Puntos: 11
naved: (3, D), Estado: false, Puntos: 12
naveuno: (8, A), Estado: true, Puntos: 14
navestete: (3, F), Estado: true, Puntos: 15

Naves ordenadas por nombre:
navecinco: (1, J), Estado: false, Puntos: 7
navecuatro: (1, J), Estado: true, Puntos: 10
navele: (4, F), Estado: false, Puntos: 5
naved: (3, D), Estado: false, Puntos: 12
naveve: (6, E), Estado: false, Puntos: 11
navecho: (1, A), Estado: false, Puntos: 3
naveis: (5, G), Estado: true, Puntos: 9
navestete: (3, F), Estado: true, Puntos: 15
navetres: (9, C), Estado: true, Puntos: 5
naveuno: (8, A), Estado: true, Puntos: 14

Naves ordenadas por puntos (Invertido):
navestete: (3, F), Estado: true, Puntos: 15
naveuno: (8, A), Estado: true, Puntos: 14
naved: (3, D), Estado: false, Puntos: 12
naveve: (6, E), Estado: false, Puntos: 11
navecuatro: (1, J), Estado: true, Puntos: 10
naveis: (5, G), Estado: true, Puntos: 9
navecinco: (5, D), Estado: false, Puntos: 7
navele: (4, F), Estado: false, Puntos: 5
navetres: (9, C), Estado: true, Puntos: 5
navecho: (1, A), Estado: false, Puntos: 3

Naves ordenadas por nombre (Invertido):
naveuno: (8, A), Estado: true, Puntos: 14
navetres: (9, C), Estado: true, Puntos: 5
navestete: (3, F), Estado: true, Puntos: 15
naveis: (5, G), Estado: true, Puntos: 9
navecho: (1, A), Estado: false, Puntos: 3
naveve: (6, E), Estado: false, Puntos: 11
naved: (3, D), Estado: false, Puntos: 12
navele: (4, F), Estado: false, Puntos: 5
navecuatro: (1, J), Estado: true, Puntos: 10
navecinco: (3, J), Estado: false, Puntos: 7
```

DemoBatalla.java

7. Estructura de laboratorio 04

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

```
lab04/  
|--- Nave.java  
|--- DemoBatalla.java  
|--- 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  
|   |--- ejec01.jpg  
|   |--- ejec02.jpg  
|   |--- ejec03.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 fácil 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 cumple 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 ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Niveles de desempeño

Puntos	Nivel			
	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.