

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 04 en el Repositorio GitHub.
- <https://github.com/cmestasz/fp2-23b/tree/main/fase01/lab04>

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 } 156 157 public static void intercambiar(Nave[] flota, int a, int b) { 158 + Nave t = flota[a]; 159 + flota[a] = flota[b]; 160 + flota[b] = t; 161 } 162 } </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 // Método que ordena por nombre de A a Z 150 public static void ordenarPorNombreBurbuja(Nave[] flota) { 151 - 152 - 153 - 154 } 155 156 // </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 } 158 // </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[i].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[i].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	}

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	}

Noveno Commit.

5. Código desarrollado

Nave.java

```
1 package fase01.lab04;
2
3 public class Nave {
4     private String nombre;
5     private int fila;
6     private String columna;
7     private boolean estado;
8     private int puntos;
9
10    // Metodos mutadores
11    public void setNombre(String n) {
12        nombre = n;
13    }
14
15    public void setFila(int f) {
16        fila = f;
17    }
18
19    public void setColumna(String c) {
20        columna = c;
21    }
22
23    public void setEstado(boolean e) {
24        estado = e;
25    }
26
27    public void setPuntos(int p) {
28        puntos = p;
29    }
30
31    // Metodos accesores
32    public String getNombre() {
33        return nombre;
34    }
35
36
37    public int getFila() {
38        return fila;
39    }
40
41
42    public String getColumna() {
43        return columna;
44    }
45
46
47    public boolean getEstado() {
48        return estado;
49    }
50
51
52    public int getPuntos() {
```

```
53     return puntos;
54
55 }
56
57 public String toString() {
58     return (nombre + ": (" + fila + ", " + columna + "). Estado: " + estado + ". Puntos:
59         " + puntos);
60 }
61 // Completar con otros metodos necesarios
62 }
```

- 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 package fase01.lab04;
6
7 import java.util.*;
8
9 public class DemoBatalla {
10     public static void main(String[] args) {
11         Nave[] misNaves = new Nave[10];
12         Scanner sc = new Scanner(System.in);
13         String nomb, col;
14         int fil, punt;
15         boolean est;
16         for (int i = 0; i < misNaves.length; i++) {
17             System.out.println("Nave " + (i + 1));
18             System.out.print("Nombre: ");
19             nomb = sc.nextLine();
20             System.out.print("Fila: ");
21             fil = sc.nextInt();
22             sc.nextLine();
23             System.out.print("Columna: ");
24             col = sc.nextLine();
25             System.out.print("Estado: ");
26             est = sc.nextBoolean();
27             sc.nextLine();
28             System.out.print("Puntos: ");
29             punt = sc.nextInt();
30             sc.nextLine();
31             misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
                                     // misNaves
32             misNaves[i].setNombre(nomb);
33             misNaves[i].setFila(fil);
34             misNaves[i].setColumna(col);
35             misNaves[i].setEstado(est);
36             misNaves[i].setPuntos(punt);
37             System.out.println();
38         }
39         System.out.println("Naves creadas:");
40         mostrarNaves(misNaves);
41         mostrarPorNombre(misNaves);
42         mostrarPorPuntos(misNaves);
43         System.out.println("Nave con mayor numero de puntos: " + mostrarMayorPuntos(misNaves)
44                             + "\n");
45
46         // leer un nombre
47         // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
48         // caso contrario
49         System.out.println("Ingrese el nombre a buscar:");
50         String nombre = sc.nextLine();
51         int pos = busquedaLinealNombre(misNaves, nombre);
52         if (pos == -1)
53             System.out.println("Nave no encontrada.\n");
54         else
```



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

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

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

```
221 public static void intercambiar(Nave[] flota, int a, int b) {  
222     Nave t = flota[a];  
223     flota[a] = flota[b];  
224     flota[b] = t;  
225 }  
226  
227 }
```

- 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

DemoBatalla.java

```
1 Nave 1
2 Nombre: naveuno
3 Fila: 5
4 Columna: G
5 Estado: true
6 Puntos: 6
7
8 Nave 2
9 Nombre: navedos
10 Fila: 2
11 Columna: A
12 Estado: false
13 Puntos: 7
14
15 Nave 3
16 Nombre: navetres
17 Fila: 8
18 Columna: J
19 Estado: true
20 Puntos: 11
21
22 Nave 4
23 Nombre: navecuatro
24 Fila: 8
25 Columna: D
26 Estado: true
27 Puntos: 9
28
29 Nave 5
30 Nombre: navecinco
31 Fila: 6
32 Columna: F
33 Estado: false
34 Puntos: 11
35
36 Nave 6
37 Nombre: naveseis
38 Fila: 2
39 Columna: B
40 Estado: false
41 Puntos: 7
42
43 Nave 7
44 Nombre: navesiete
45 Fila: 3
46 Columna: F
47 Estado: true
48 Puntos: 2
49
50 Nave 8
51 Nombre: naveocho
52 Fila: 4
```

```
53 Columna: A
54 Estado: true
55 Puntos: 10
56
57 Nave 9
58 Nombre: navenueve
59 Fila: 3
60 Columna: I
61 Estado: false
62 Puntos: 13
63
64 Nave 10
65 Nombre: navediez
66 Fila: 2
67 Columna: E
68 Estado: true
69 Puntos: 7
70
71 Naves creadas:
72 naveuno: (5, G). Estado: true. Puntos: 6
73 navedos: (2, A). Estado: false. Puntos: 7
74 navetres: (8, J). Estado: true. Puntos: 11
75 navecuatro: (8, D). Estado: true. Puntos: 9
76 navecinco: (6, F). Estado: false. Puntos: 11
77 naveseis: (2, B). Estado: false. Puntos: 7
78 navesiete: (3, F). Estado: true. Puntos: 2
79 naveocho: (4, A). Estado: true. Puntos: 10
80 navenueve: (3, I). Estado: false. Puntos: 13
81 navediez: (2, E). Estado: true. Puntos: 7
82
83 Ingrese el nombre a buscar:
84 navecinco
85 Naves con el nombre navecinco:
86 navecinco: (6, F). Estado: false. Puntos: 11
87
88 Ingrese la cantidad de puntos maximos:
89 6
90 Naves con 6 puntos como maximo:
91 naveuno: (5, G). Estado: true. Puntos: 6
92 navesiete: (3, F). Estado: true. Puntos: 2
93
94 Nave con mayor numero de puntos: navenueve: (3, I). Estado: false. Puntos: 13
95
96 Ingrese el nombre a buscar:
97 naveonce
98 Nave no encontrada.
99
100 Naves ordenadas por puntos:
101 navesiete: (3, F). Estado: true. Puntos: 2
102 naveuno: (5, G). Estado: true. Puntos: 6
103 navedos: (2, A). Estado: false. Puntos: 7
104 naveseis: (2, B). Estado: false. Puntos: 7
105 navediez: (2, E). Estado: true. Puntos: 7
106 navecuatro: (8, D). Estado: true. Puntos: 9
107 naveocho: (4, A). Estado: true. Puntos: 10
108 navetres: (8, J). Estado: true. Puntos: 11
```

```
109 navecinco: (6, F). Estado: false. Puntos: 11
110 navenueve: (3, I). Estado: false. Puntos: 13
111
112 Naves ordenadas por nombre:
113 navecinco: (6, F). Estado: false. Puntos: 11
114 navecuatro: (8, D). Estado: true. Puntos: 9
115 navediez: (2, E). Estado: true. Puntos: 7
116 navedos: (2, A). Estado: false. Puntos: 7
117 navenueve: (3, I). Estado: false. Puntos: 13
118 naveocho: (4, A). Estado: true. Puntos: 10
119 naveseis: (2, B). Estado: false. Puntos: 7
120 navesiete: (3, F). Estado: true. Puntos: 2
121 navetres: (8, J). Estado: true. Puntos: 11
122 naveuno: (5, G). Estado: true. Puntos: 6
123
124 Ingrese el nombre a buscar:
125 navedos
126 Nave encontrada: navedos: (2, A). Estado: false. Puntos: 7
127
128 Naves ordenadas por puntos:
129 navesiete: (3, F). Estado: true. Puntos: 2
130 naveuno: (5, G). Estado: true. Puntos: 6
131 navediez: (2, E). Estado: true. Puntos: 7
132 navedos: (2, A). Estado: false. Puntos: 7
133 naveseis: (2, B). Estado: false. Puntos: 7
134 navecuatro: (8, D). Estado: true. Puntos: 9
135 naveocho: (4, A). Estado: true. Puntos: 10
136 navecinco: (6, F). Estado: false. Puntos: 11
137 navetres: (8, J). Estado: true. Puntos: 11
138 navenueve: (3, I). Estado: false. Puntos: 13
139
140 Naves ordenadas por nombre:
141 navecinco: (6, F). Estado: false. Puntos: 11
142 navecuatro: (8, D). Estado: true. Puntos: 9
143 navediez: (2, E). Estado: true. Puntos: 7
144 navedos: (2, A). Estado: false. Puntos: 7
145 navenueve: (3, I). Estado: false. Puntos: 13
146 naveocho: (4, A). Estado: true. Puntos: 10
147 naveseis: (2, B). Estado: false. Puntos: 7
148 navesiete: (3, F). Estado: true. Puntos: 2
149 navetres: (8, J). Estado: true. Puntos: 11
150 naveuno: (5, G). Estado: true. Puntos: 6
151
152 Naves ordenadas por puntos (Invertido):
153 navenueve: (3, I). Estado: false. Puntos: 13
154 navecinco: (6, F). Estado: false. Puntos: 11
155 navetres: (8, J). Estado: true. Puntos: 11
156 naveocho: (4, A). Estado: true. Puntos: 10
157 navecuatro: (8, D). Estado: true. Puntos: 9
158 navediez: (2, E). Estado: true. Puntos: 7
159 navedos: (2, A). Estado: false. Puntos: 7
160 naveseis: (2, B). Estado: false. Puntos: 7
161 naveuno: (5, G). Estado: true. Puntos: 6
162 navesiete: (3, F). Estado: true. Puntos: 2
163
164 Naves ordenadas por nombre (Invertido):
```

```
165 naveuno: (5, G). Estado: true. Puntos: 6
166 navetres: (8, J). Estado: true. Puntos: 11
167 navesiete: (3, F). Estado: true. Puntos: 2
168 naveseis: (2, B). Estado: false. Puntos: 7
169 naveocho: (4, A). Estado: true. Puntos: 10
170 navenueve: (3, I). Estado: false. Puntos: 13
171 navedos: (2, A). Estado: false. Puntos: 7
172 navediez: (2, E). Estado: true. Puntos: 7
173 navecuatro: (8, D). Estado: true. Puntos: 9
174 navecinco: (6, F). Estado: false. Puntos: 11
```


7. Estructura de laboratorio 04

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

```
lab04/  
|--- Nave.java  
|--- DemoBatalla.java  
|--- ejec01.bash  
|--- ejec02.bash  
|--- ejec03.bash  
|--- Informe.tex  
|--- Informe.pdf  
|--- img  
|   |--- logo_abet.png  
|   |--- logo_episunsa.png  
|   |--- logo_unsa.jpg  
|   |--- commit01.jpg  
|   |--- commit02.jpg  
|   |--- commit03.jpg  
|   |--- commit04.jpg  
|   |--- commit05.jpg  
|   |--- commit06.jpg  
|   |--- commit07.jpg  
|   |--- commit08.jpg  
|   |--- commit09.jpg
```

8. Rúbricas

8.1. Entregable Informe

Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

8.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobatoria, siempre y cuando cumpla con todos los items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Niveles de desempeño

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.