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`

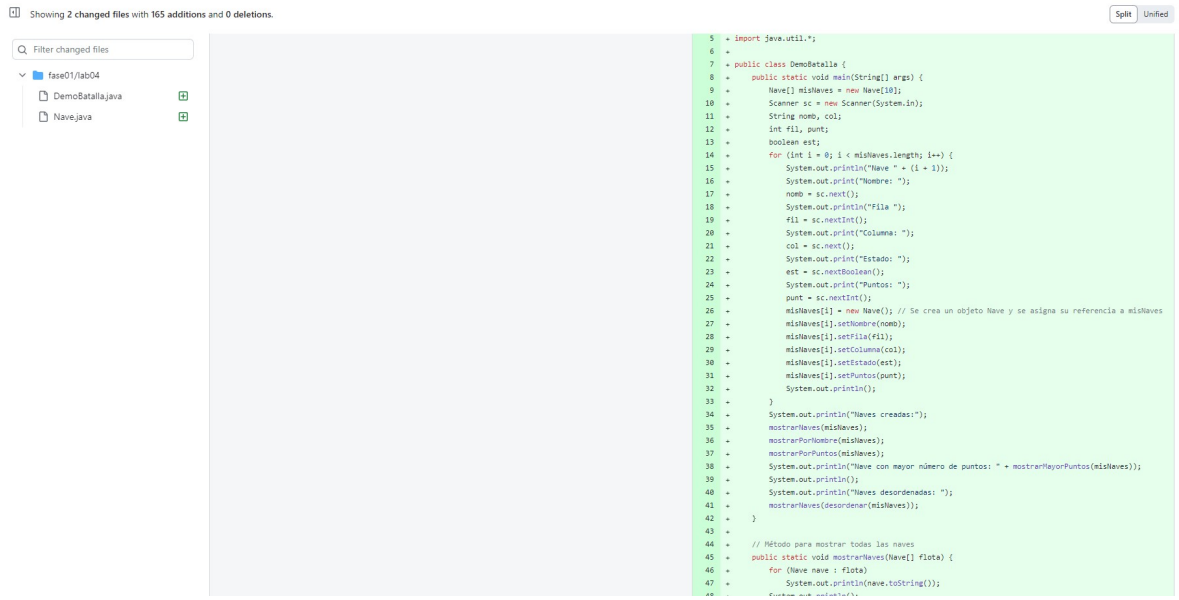
4. Actividades con el repositorio GitHub

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
```



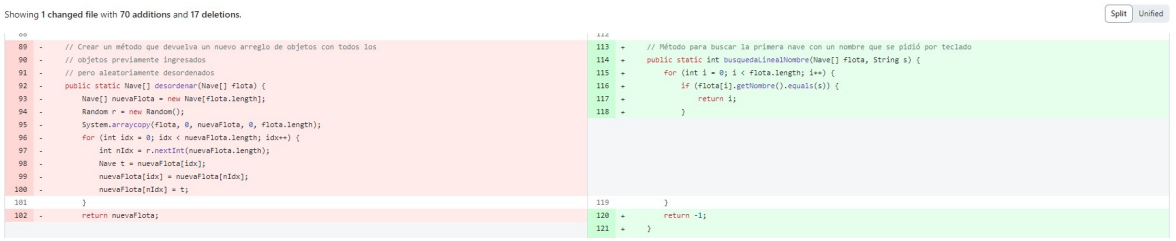
Primer Commit.

Actualizando DemoBatalla.java

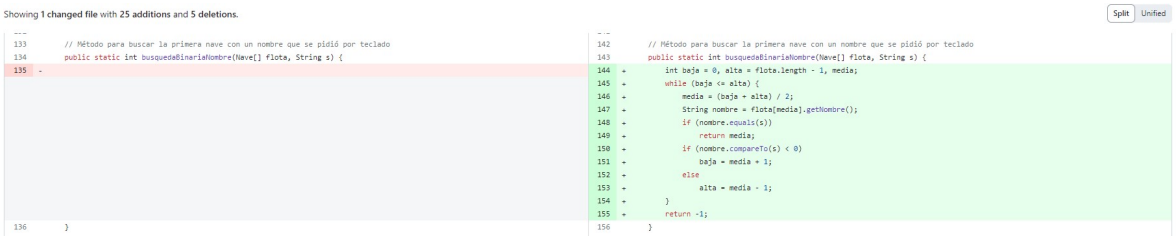
```
$ code DemoBatalla.java
```

Segundo - Noveno Commit / DemoBatalla.java

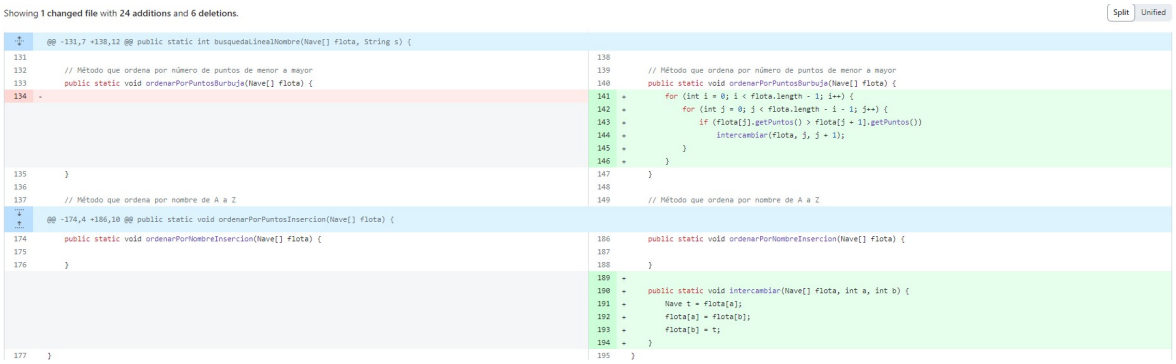
```
$ git add .
$ git commit -m "Metodo busquedaLinealNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo busquedaBinariaNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosBurbuja()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ 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
```



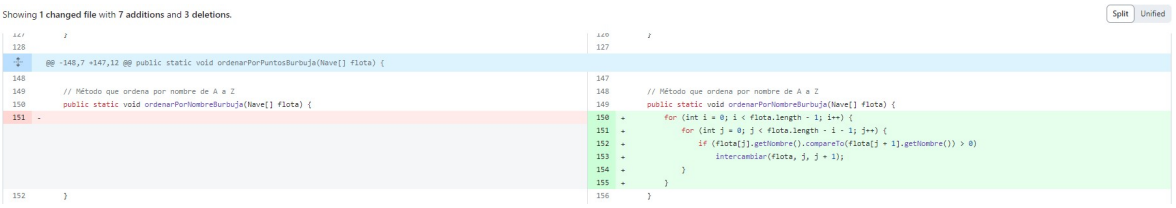
Segundo Commit.



Tercer Commit.

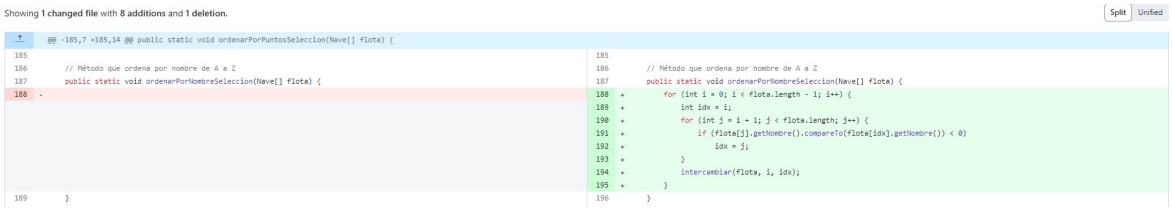


Cuarto Commit.

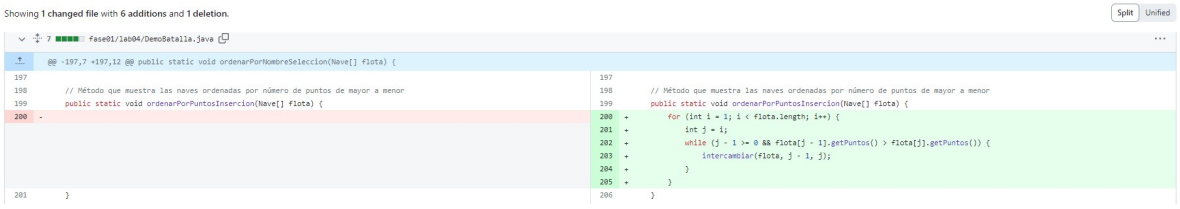


Quinto Commit.

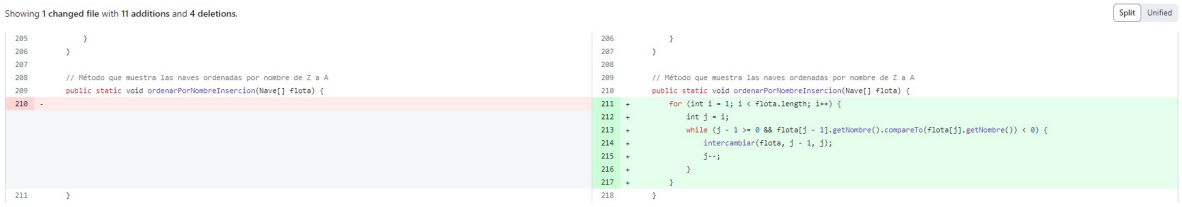
Sexto Commit.



Séptimo Commit.



Octavo Commit.



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:
                " + puntos);
59      }
60      // Completar con otros metodos necesarios
61  }
```

- 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

```java
// Laboratorio Nro 4 - Actividad 1
// Autor: Christian Mestas
// Colaboro: Marco Aedo, clases DemoBatalla y Nave

package fase01.lab04;

import java.util.*;

public class DemoBatalla {
    public static void main(String[] args) {
        Nave[] misNaves = new Nave[10];
        Scanner sc = new Scanner(System.in);
        String nomb, col;
        int fil, punt;
        boolean est;
        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i + 1));
            System.out.print("Nombre: ");
            nomb = sc.nextLine();
            System.out.print("Fila: ");
            fil = sc.nextInt();
            sc.nextLine();
            System.out.print("Columna: ");
            col = sc.nextLine();
            System.out.print("Estado: ");
            est = sc.nextBoolean();
            sc.nextLine();
            System.out.print("Puntos: ");
            punt = sc.nextInt();
            sc.nextLine();
            misNaves[i] = new Nave(); // Se crea un objeto Nave y se asigna su referencia a
                misNaves
            misNaves[i].setNombre(nomb);
            misNaves[i].setFila(fil);
            misNaves[i].setColumna(col);
            misNaves[i].setEstado(est);
            misNaves[i].setPuntos(punt);
            System.out.println();
        }
        System.out.println("Naves creadas:");
        mostrarNaves(misNaves);
        mostrarPorNombre(misNaves);
        mostrarPorPuntos(misNaves);
        System.out.println("Nave con mayor numero de puntos: " + mostrarMayorPuntos(misNaves)
            + "\n");

        // leer un nombre
        // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
        // caso contrario
        System.out.println("Ingrese el nombre a buscar:");
        String nombre = sc.nextLine();
        int pos = busquedaLinealNombre(misNaves, nombre);
        if (pos == -1)
            System.out.println("Nave no encontrada.\n");
        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                 intercambiar(flota, j - 1, j);
217                 j--;
218             }
219         }
220     }
```

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

```
Nave 1
Nombre: naveuno
Fila: 5
Columna: G
Estado: true
Puntos: 6

Nave 2
Nombre: navedos
Fila: 2
Columna: A
Estado: false
Puntos: 7

Nave 3
Nombre: navetres
Fila: 8
Columna: J
Estado: true
Puntos: 11

Nave 4
Nombre: navecuatro
Fila: 8
Columna: D
Estado: true
Puntos: 9

Nave 5
Nombre: navecinco
Fila: 6
Columna: F
Estado: false
Puntos: 11

Nave 6
Nombre: naveseis
Fila: 2
Columna: B
Estado: false
Puntos: 7

Nave 7
Nombre: navesiete
Fila: 3
Columna: F
Estado: true
Puntos: 2

Nave 8
Nombre: naveocho
Fila: 4
```

```
Columna: A
Estado: true
Puntos: 10

Nave 9
Nombre: navenueve
Fila: 3
Columna: I
Estado: false
Puntos: 13

Nave 10
Nombre: navediez
Fila: 2
Columna: E
Estado: true
Puntos: 7

Naves creadas:
naveuno: (5, G). Estado: true. Puntos: 6
navedos: (2, A). Estado: false. Puntos: 7
navetres: (8, J). Estado: true. Puntos: 11
navecuatro: (8, D). Estado: true. Puntos: 9
navecinco: (6, F). Estado: false. Puntos: 11
naveseis: (2, B). Estado: false. Puntos: 7
navesiete: (3, F). Estado: true. Puntos: 2
naveocho: (4, A). Estado: true. Puntos: 10
navenueve: (3, I). Estado: false. Puntos: 13
navediez: (2, E). Estado: true. Puntos: 7

Ingrese el nombre a buscar:
navecinco
Naves con el nombre navecinco:
navecinco: (6, F). Estado: false. Puntos: 11

Ingrese la cantidad de puntos maximos:
6
Naves con 6 puntos como maximo:
naveuno: (5, G). Estado: true. Puntos: 6
navesiete: (3, F). Estado: true. Puntos: 2

Nave con mayor numero de puntos: navenueve: (3, I). Estado: false. Puntos: 13

Ingrese el nombre a buscar:
naveonce
Nave no encontrada.

Naves ordenadas por puntos:
navesiete: (3, F). Estado: true. Puntos: 2
naveuno: (5, G). Estado: true. Puntos: 6
navedos: (2, A). Estado: false. Puntos: 7
naveseis: (2, B). Estado: false. Puntos: 7
navediez: (2, E). Estado: true. Puntos: 7
navecuatro: (8, D). Estado: true. Puntos: 9
naveocho: (4, A). Estado: true. Puntos: 10
navetres: (8, J). Estado: true. Puntos: 11
```

```
navecinco: (6, F). Estado: false. Puntos: 11
navenueve: (3, I). Estado: false. Puntos: 13

Naves ordenadas por nombre:
navecinco: (6, F). Estado: false. Puntos: 11
navecuatro: (8, D). Estado: true. Puntos: 9
navediez: (2, E). Estado: true. Puntos: 7
navedos: (2, A). Estado: false. Puntos: 7
navenueve: (3, I). Estado: false. Puntos: 13
naveocho: (4, A). Estado: true. Puntos: 10
naveseis: (2, B). Estado: false. Puntos: 7
navesiete: (3, F). Estado: true. Puntos: 2
navetres: (8, J). Estado: true. Puntos: 11
naveuno: (5, G). Estado: true. Puntos: 6

Ingrese el nombre a buscar:
navedos
Nave encontrada: navedos: (2, A). Estado: false. Puntos: 7

Naves ordenadas por puntos:
navesiete: (3, F). Estado: true. Puntos: 2
naveuno: (5, G). Estado: true. Puntos: 6
navediez: (2, E). Estado: true. Puntos: 7
navedos: (2, A). Estado: false. Puntos: 7
naveseis: (2, B). Estado: false. Puntos: 7
navecuatro: (8, D). Estado: true. Puntos: 9
naveocho: (4, A). Estado: true. Puntos: 10
navecinco: (6, F). Estado: false. Puntos: 11
navetres: (8, J). Estado: true. Puntos: 11
navenueve: (3, I). Estado: false. Puntos: 13

Naves ordenadas por nombre:
navecinco: (6, F). Estado: false. Puntos: 11
navecuatro: (8, D). Estado: true. Puntos: 9
navediez: (2, E). Estado: true. Puntos: 7
navedos: (2, A). Estado: false. Puntos: 7
navenueve: (3, I). Estado: false. Puntos: 13
naveocho: (4, A). Estado: true. Puntos: 10
naveseis: (2, B). Estado: false. Puntos: 7
navesiete: (3, F). Estado: true. Puntos: 2
navetres: (8, J). Estado: true. Puntos: 11
naveuno: (5, G). Estado: true. Puntos: 6

Naves ordenadas por puntos (Invertido):
navenueve: (3, I). Estado: false. Puntos: 13
navecinco: (6, F). Estado: false. Puntos: 11
navetres: (8, J). Estado: true. Puntos: 11
naveocho: (4, A). Estado: true. Puntos: 10
navecuatro: (8, D). Estado: true. Puntos: 9
navediez: (2, E). Estado: true. Puntos: 7
navedos: (2, A). Estado: false. Puntos: 7
naveseis: (2, B). Estado: false. Puntos: 7
naveuno: (5, G). Estado: true. Puntos: 6
navesiete: (3, F). Estado: true. Puntos: 2

Naves ordenadas por nombre (Invertido):
```

```
naveuno: (5, G). Estado: true. Puntos: 6
navetres: (8, J). Estado: true. Puntos: 11
navesiete: (3, F). Estado: true. Puntos: 2
naveseis: (2, B). Estado: false. Puntos: 7
naveocho: (4, A). Estado: true. Puntos: 10
navenueve: (3, I). Estado: false. Puntos: 13
navedos: (2, A). Estado: false. Puntos: 7
navediez: (2, E). Estado: true. Puntos: 7
navecuatro: (8, D). Estado: true. Puntos: 9
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

	Nivel			
Puntos	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.