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`

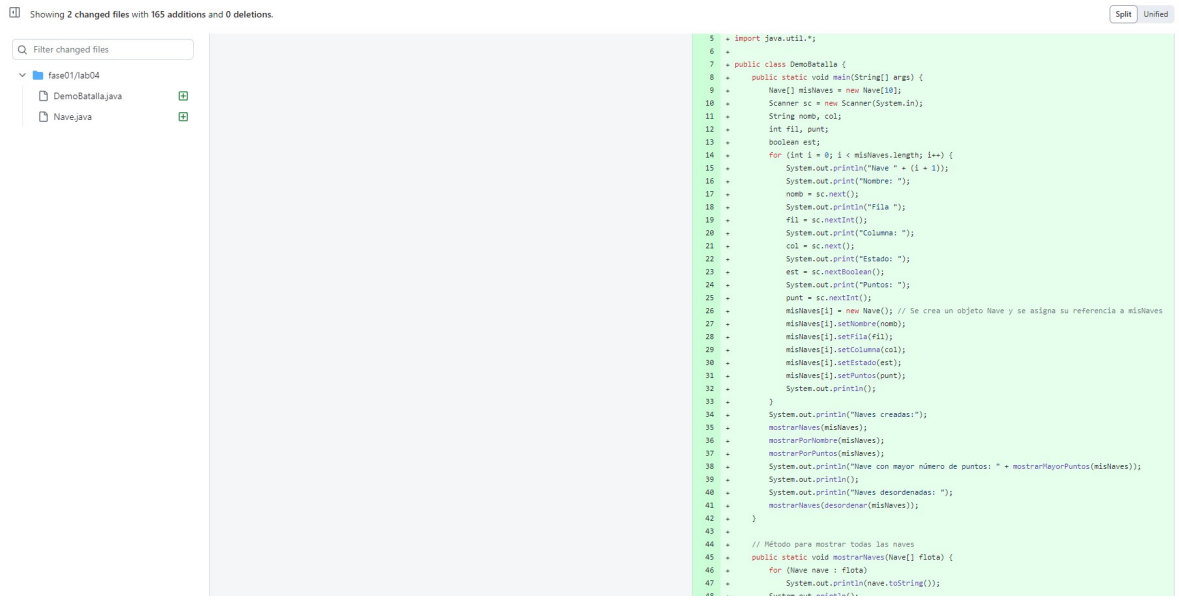
4. Actividades con el repositorio GitHub

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
```



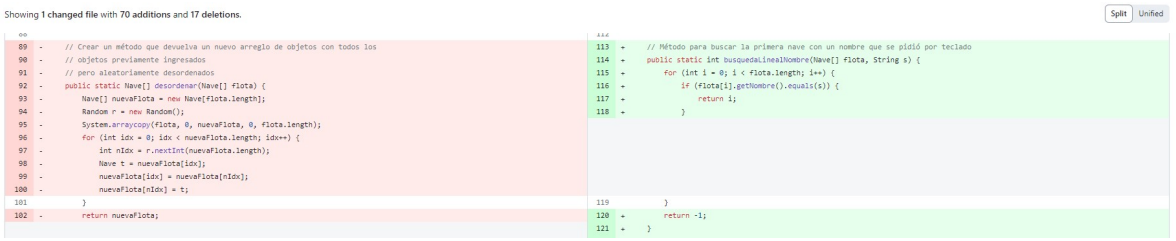
Primer Commit.

Actualizando DemoBatalla.java

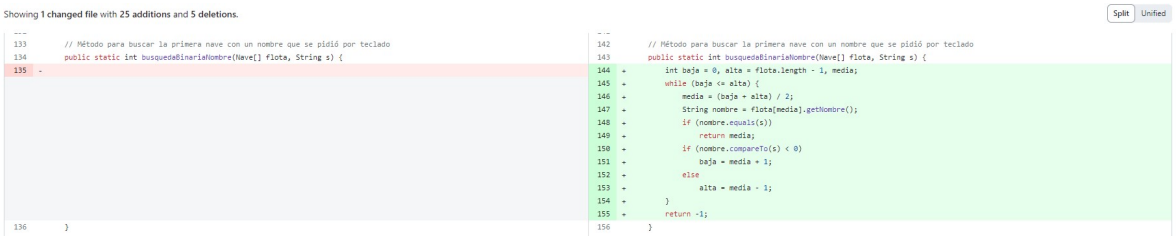
```
$ code DemoBatalla.java
```

Segundo - Noveno Commit / DemoBatalla.java

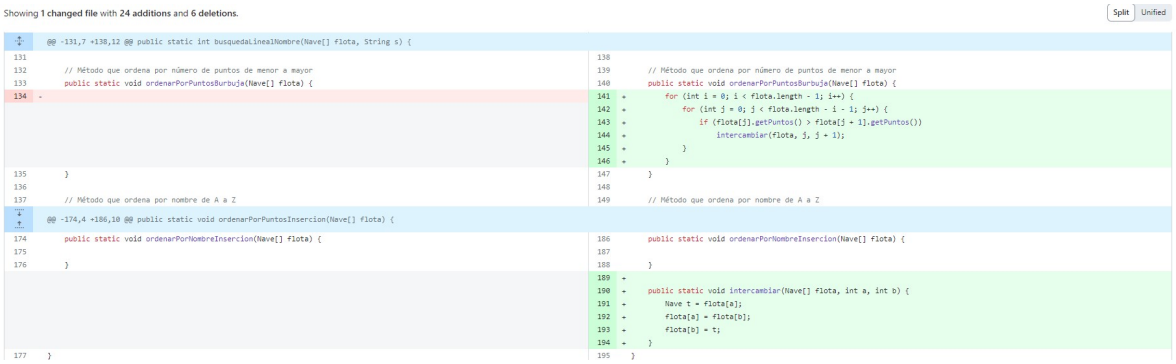
```
$ git add .
$ git commit -m "Metodo busquedaLinealNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo busquedaBinariaNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosBurbuja()"
$ code DemoBatalla.java
$ git add DemoBatalla.java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



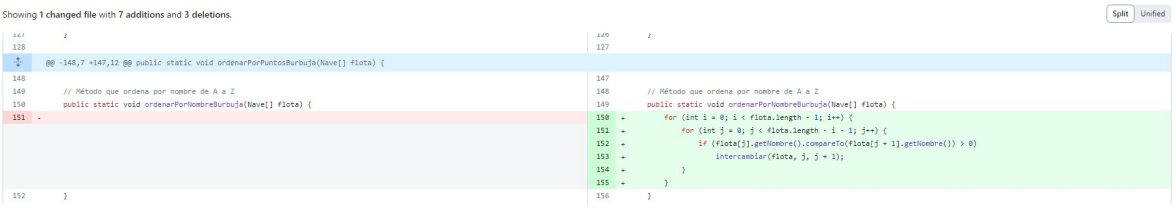
Segundo Commit.



Tercer Commit.

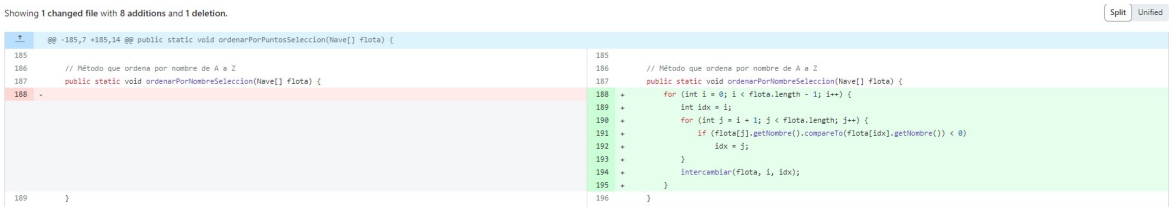


Cuarto Commit.

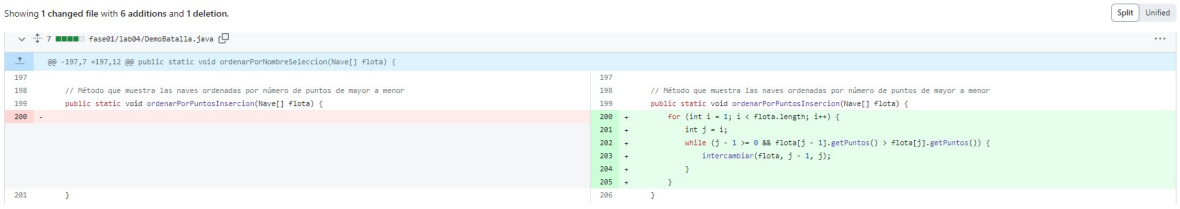


Quinto Commit.

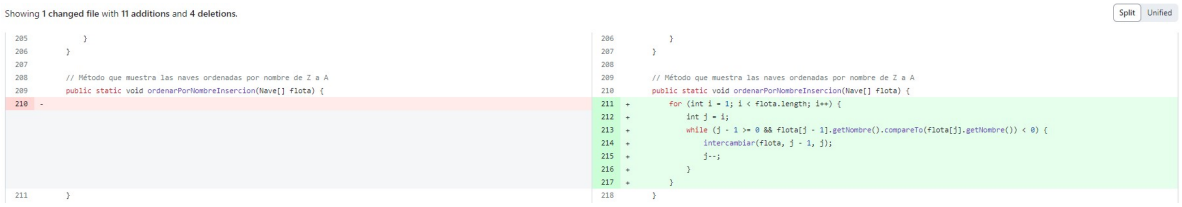
Sexto Commit.



Séptimo Commit.



Octavo Commit.



Noveno Commit.

5. Código desarrollado

Nave.java

```java
public class Nave {
    private String nombre;
    private int fila;
    private String columna;
    private boolean estado;
    private int puntos;

    // Metodos mutadores
    public void setNombre(String n) {
        nombre = n;
    }

    public void setFila(int f) {
        fila = f;
    }

    public void setColumna(String c) {
        columna = c;
    }

    public void setEstado(boolean e) {
        estado = e;
    }

    public void setPuntos(int p) {
        puntos = p;
    }

    // Metodos accesores
    public String getNombre() {
        return nombre;

    }

    public int getFila() {
        return fila;

    }

    public String getColumna() {
        return columna;

    }

    public boolean getEstado() {
        return estado;

    }

    public int getPuntos() {
        return puntos;

```

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

- 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

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
            System.out.println("Nave encontrada: " + misNaves[pos] + "\n");
        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();
```

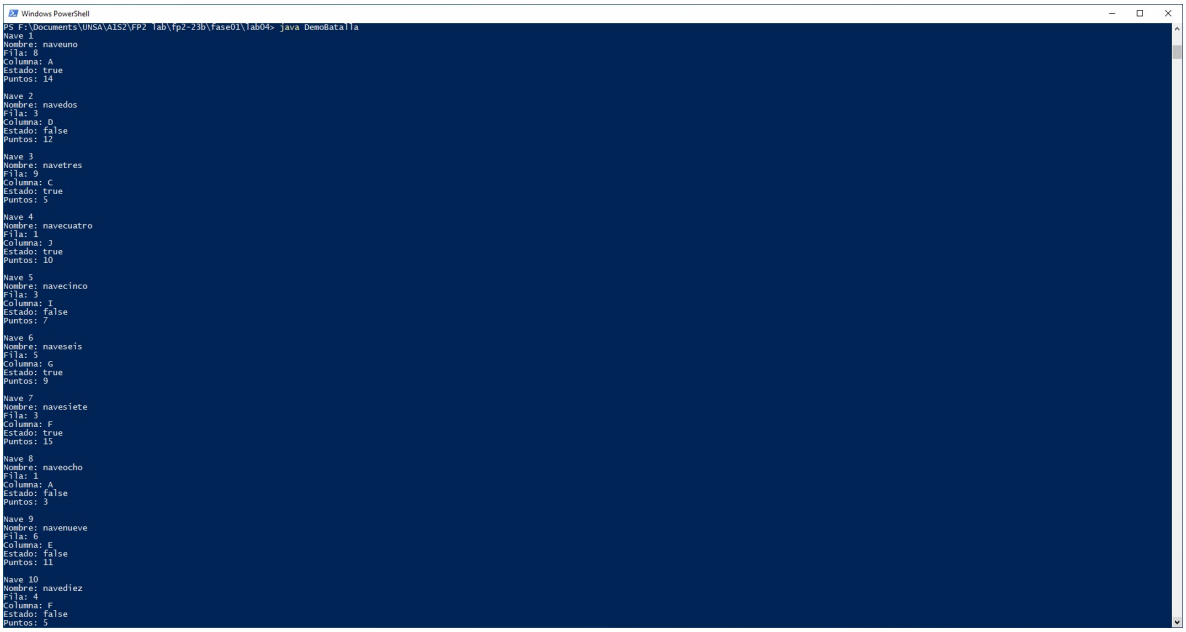
```
110         System.out.println("Naves con " + puntos + " puntos como maximo:");
111         for (Nave nave : flota) {
112             if (nave.getPuntos() <= puntos)
113                 System.out.println(nave.toString());
114         }
115         System.out.println();
116     }
117 
118     // 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                 intercambiar(flota, j - 1, j);
215                 j--;
216             }
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



DemoBatalla.java

DemoBatalla.java

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 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.