

# Informe de Laboratorio 04

Tema: Arreglos de Objetos, Búsqueda y Ordenamientos

Nota			

Estudiante	Escuela	${f Asign atura}$
		Fundamentos de la
Christian Mestas Zegarra	Escuela Profesional de	Programación 2
cmestasz@unsa.edu.pe	Ingeniería de Sistemas	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

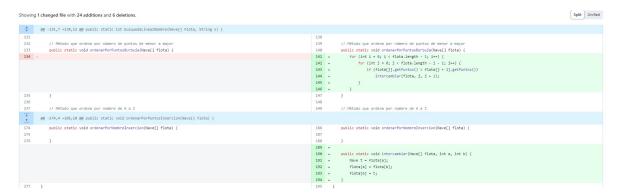
```
$ git add .
$ git commit -m "Metodo busquedaLinealNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo busquedaBinariaNombre()"
$ code DemoBatalla.java
$ git add DemoBatalla.java
$ git commit -m "Metodo ordenarPorPuntosBurbuja()"
$ code DemoBatalla.java
$ git add DemoBatalla.java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



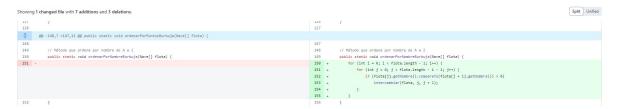


## Segundo Commit.

#### Tercer Commit.



Cuarto Commit.

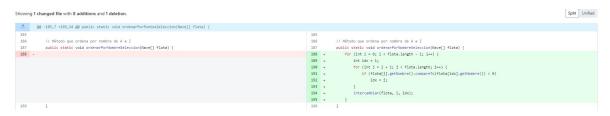


Quinto Commit.

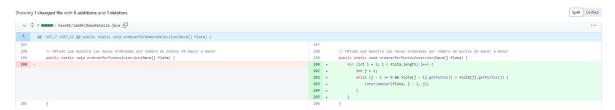




### Sexto Commit.



## Séptimo Commit.



# Octavo Commit.



Noveno Commit.





# 5. Código desarrollado

## Nave.java

```
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) {
10
           nombre = n;
12
       public void setFila(int f) {
           fila = f;
14
16
       public void setColumna(String c) {
17
18
           columna = c;
19
20
       public void setEstado(boolean e) {
21
           estado = e;
23
       public void setPuntos(int p) {
           puntos = p;
26
27
28
       // Metodos accesores
29
       public String getNombre() {
30
           return nombre;
32
33
       }
34
       public int getFila() {
35
36
           return fila;
39
       public String getColumna() {
40
           return columna;
41
42
       }
43
44
       public boolean getEstado() {
           return estado;
46
47
       }
48
       public int getPuntos() {
50
           return puntos;
```





- 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

```
// Laboratorio Nro 4 - Actividad 1
   // Autor: Christian Mestas
   // Colaboro: Marco Aedo, clases DemoBatalla y Nave
   import java.util.*;
6
   public class DemoBatalla {
       public static void main(String[] args) {
           Nave[] misNaves = new Nave[10];
           Scanner sc = new Scanner(System.in);
           String nomb, col;
11
           int fil, punt;
           boolean est:
13
           for (int i = 0; i < misNaves.length; i++) {</pre>
              System.out.println("Nave " + (i + 1));
              System.out.print("Nombre: ");
              nomb = sc.nextLine();
              System.out.print("Fila: ");
18
              fil = sc.nextInt();
19
              sc.nextLine();
20
              System.out.print("Columna: ");
21
              col = sc.nextLine();
              System.out.print("Estado: ");
              est = sc.nextBoolean();
24
              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);
30
              misNaves[i].setFila(fil);
31
              misNaves[i].setColumna(col);
              misNaves[i].setEstado(est);
33
              misNaves[i].setPuntos(punt);
              System.out.println();
36
           System.out.println("Naves creadas:");
37
           mostrarNaves(misNaves):
           mostrarPorNombre(misNaves);
           mostrarPorPuntos(misNaves);
           System.out.println("Nave con mayor numero de puntos: " + mostrarMayorPuntos(misNaves)
               + "\n");
42
           // leer un nombre
43
           // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
44
           // caso contrario
45
           System.out.println("Ingrese el nombre a buscar:");
           String nombre = sc.nextLine();
           int pos = busquedaLinealNombre(misNaves, nombre);
           if (pos == -1)
49
              System.out.println("Nave no encontrada.\n");
50
           else
              System.out.println("Nave encontrada: " + misNaves[pos] + "\n");
           System.out.println("Naves ordenadas por puntos:");
```





```
ordenarPorPuntosBurbuja(misNaves);
54
           mostrarNaves(misNaves);
           System.out.println("Naves ordenadas por nombre:");
56
           ordenarPorNombreBurbuja(misNaves);
           mostrarNaves(misNaves);
58
59
           // mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en
           // caso contrario
61
           System.out.println("Ingrese el nombre a buscar:");
62
           nombre = sc.nextLine();
63
           pos = busquedaBinariaNombre(misNaves, nombre);
64
           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:");
69
           ordenarPorPuntosSeleccion(misNaves);
70
           mostrarNaves(misNaves);
71
           System.out.println("Naves ordenadas por nombre:");
           ordenarPorNombreSeleccion(misNaves);
73
           mostrarNaves(misNaves);
74
           System.out.println("Naves ordenadas por puntos (Invertido):");
           ordenarPorPuntosInsercion(misNaves);
76
           mostrarNaves(misNaves);
           System.out.println("Naves ordenadas por nombre (Invertido):");
           ordenarPorNombreInsercion(misNaves);
           mostrarNaves(misNaves);
       }
81
82
        // Metodo para mostrar todas las naves
83
       public static void mostrarNaves(Nave[] flota) {
84
           for (Nave nave : flota)
85
               System.out.println(nave.toString());
           System.out.println();
       }
89
        // Metodo para mostrar todas las naves de un nombre que se pide por teclado
90
       public static void mostrarPorNombre(Nave[] flota) {
91
           Scanner sc = new Scanner(System.in);
           System.out.println("Ingrese el nombre a buscar:");
           String nombre = sc.nextLine();
           System.out.println("Naves con el nombre " + nombre + ":");
95
           for (Nave nave : flota) {
96
               if (nave.getNombre().equals(nombre))
97
                   System.out.println(nave.toString());
0.8
99
           System.out.println();
100
       }
        // Metodo para mostrar todas las naves con un numero de puntos inferior o igual
        // al numero de puntos que se pide por teclado
104
        public static void mostrarPorPuntos(Nave[] flota) {
           Scanner sc = new Scanner(System.in);
           System.out.println("Ingrese la cantidad de puntos maximos:");
           int puntos = sc.nextInt();
108
           sc.nextLine();
```





```
System.out.println("Naves con " + puntos + " puntos como maximo:");
            for (Nave nave : flota) {
111
                if (nave.getPuntos() <= puntos)</pre>
                   System.out.println(nave.toString());
113
114
            System.out.println();
        }
117
        // Metodo que devuelve la Nave con mayor numero de Puntos
118
        public static Nave mostrarMayorPuntos(Nave[] flota) {
119
            int maxIdx = 0;
            for (int i = 0; i < flota.length; i++) {</pre>
                if (flota[i].getPuntos() > flota[maxIdx].getPuntos())
                   maxIdx = i;
124
            return flota[maxIdx];
126
127
        // Metodo para buscar la primera nave con un nombre que se pidio por teclado
128
        public static int busquedaLinealNombre(Nave[] flota, String s) {
            for (int i = 0; i < flota.length; i++) {</pre>
130
                if (flota[i].getNombre().equals(s)) {
                   return i;
133
            }
            return -1;
        // Metodo que ordena por numero de puntos de menor a mayor
        public static void ordenarPorPuntosBurbuja(Nave[] flota) {
139
            for (int i = 0; i < flota.length - 1; i++) {</pre>
140
                for (int j = 0; j < flota.length - i - 1; <math>j++) {
141
                    if (flota[j].getPuntos() > flota[j + 1].getPuntos())
                       intercambiar(flota, j, j + 1);
143
144
            }
145
        }
146
147
        // Metodo que ordena por nombre de A a Z
        public static void ordenarPorNombreBurbuja(Nave[] flota) {
            for (int i = 0; i < flota.length - 1; i++) {</pre>
                for (int j = 0; j < flota.length - i - 1; <math>j++) {
                   if (flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0)
                       intercambiar(flota, j, j + 1);
153
               }
154
            }
        }
        // Metodo para buscar la primera nave con un nombre que se pidio por teclado
158
        public static int busquedaBinariaNombre(Nave[] flota, String s) {
159
            int baja = 0, alta = flota.length - 1, media;
160
            while (baja <= alta) {</pre>
                media = (baja + alta) / 2;
                String nombre = flota[media].getNombre();
                if (nombre.equals(s))
164
                   return media;
```





```
if (nombre.compareTo(s) < 0)</pre>
                    baja = media + 1;
167
                else
                    alta = media - 1;
            }
            return -1;
        }
173
        // Metodo que ordena por numero de puntos de menor a mayor
174
        public static void ordenarPorPuntosSeleccion(Nave[] flota) {
            for (int i = 0; i < flota.length - 1; i++) {</pre>
                int idx = i;
                for (int j = i + 1; j < flota.length; j++) {
                    if (flota[j].getPuntos() < flota[idx].getPuntos())</pre>
                        idx = j;
180
181
                intercambiar(flota, i, idx);
182
            }
183
        }
184
185
        // Metodo 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++) {</pre>
188
                int idx = i;
                for (int j = i + 1; j < flota.length; j++) {</pre>
                    if (flota[j].getNombre().compareTo(flota[idx].getNombre()) < 0)</pre>
                        idx = j;
                }
                intercambiar(flota, i, idx);
194
            }
195
        }
196
197
        // Metodo que muestra las naves ordenadas por numero de puntos de mayor a menor
        public static void ordenarPorPuntosInsercion(Nave[] flota) {
199
            for (int i = 1; i < flota.length; i++) {</pre>
200
                int j = i;
201
                while (j - 1 \ge 0 \&\& flota[j - 1].getPuntos() < flota[j].getPuntos()) {
202
                    intercambiar(flota, j - 1, j);
203
                    j--;
                }
            }
        }
207
208
        // Metodo que muestra las naves ordenadas por nombre de Z a A
209
        public static void ordenarPorNombreInsercion(Nave[] flota) {
210
            for (int i = 1; i < flota.length; i++) {</pre>
211
                int j = i;
212
                while (j - 1 >= 0 && flota[j - 1].getNombre().compareTo(flota[j].getNombre()) < 0)</pre>
213
                    intercambiar(flota, j - 1, j);
214
                    j--;
                }
            }
219
        public static void intercambiar(Nave[] flota, int a, int b) {
220
```





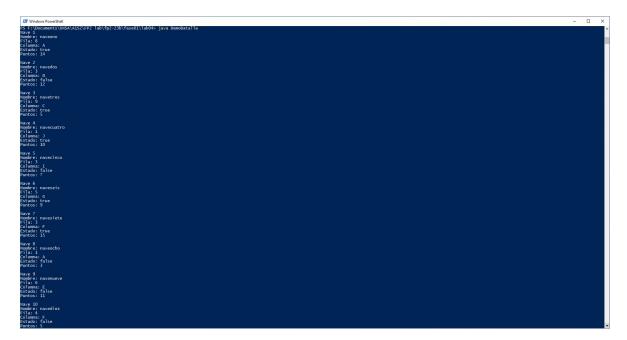
```
Nave t = flota[a];
flota[a] = flota[b];
flota[b] = t;

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 ordenar Por Puntos Burbuja() 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 ordenar Por PuntosSelección() 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.