

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la programación II				
TÍTULO DE LA PRÁCTICA:	Arreglos de Objetos, Búsquedas y Ordenamientos				
NÚMERO DE PRÁCTICA:	04	AÑO LECTIVO:	2024	NRO. SEMESTRE:	02
FECHA DE PRESENTACIÓN	13/10/2024	HORA DE PRESENTACIÓN	11/59/00		
INTEGRANTE (s) <i>José León Enrique Hatches Curo</i>				NOTA (0-20)	<i>Nota colocada por el docente</i>
DOCENTE(s): <i>Ing. Lino Pinto Oppe</i>					

RESULTADOS Y PRUEBAS
<p><b>I. EJERCICIOS RESUELTOS:</b></p> <p><i>El estudiante coloca la evidencia de los ejercicios propuestos realizados en la sesión de laboratorio, en el tiempo o duración indicado por el docente.</i></p> <p><i>El docente debe colocar la retroalimentación por cada ejercicio que el estudiante/grupo ha presentado.</i></p> <p>Repositorio en GitHub:</p> <p><a href="https://github.com/LeonHatches/Laboratorio-04">https://github.com/LeonHatches/Laboratorio-04</a></p>

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 2</p>

### *DEMO BATALLA NUEVOS MÉTODOS:*

*Analice, complete y pruebe el Código de la clase DemoBatalla.*

- *Primero, en este es el main donde se muestra el orden de ejecución de los métodos.*

```
// Leer nombre
String nombre = ingresarNombre();

//mostrar los datos de la nave con dicho nombre,
//mensaje de "no encontrado" en caso contrario
int pos = busquedaLinealNombre(misNaves, nombre);
mostrarUna (misNaves, pos, nombre);

// Ordenamiento BURBUJA POR PUNTOS
ordenarPorPuntosBurbuja (misNaves);
mostrarNaves (misNaves);

// Ordenamiento BURBUJA POR A - Z
ordenarPorNombreBurbuja (misNaves);
mostrarNaves (misNaves);

// Leer nombre
nombre = ingresarNombre();

// Mostrar los datos de la nave con dicho nombre,
// mensaje de "no encontrado" en caso contrario.
pos=busquedaBinariaNombre(misNaves, nombre);
mostrarUna (misNaves, pos, nombre);
```

```
// Ordenamiento SELECCION POR PUNTOS
ordenarPorPuntosSeleccion(misNaves);
mostrarNaves(misNaves);

// Ordenamiento SELECCION POR NOMBRE
ordenarPorNombreSeleccion(misNaves);
mostrarNaves(misNaves);

// Ordenamiento INSERCIÓN POR PUNTOS
ordenarPorPuntosInsercion(misNaves);
mostrarNaves(misNaves);

// Ordenamiento INSERCIÓN POR NOMBRE
ordenarPorNombreInsercion(misNaves);
mostrarNaves(misNaves);
```

- *Luego, estos métodos en conjunto hacen que se ingrese nombre y se muestre según lo encontrado en las búsquedas.*

```
//Leer un nombre
public static String ingresarNombre()
{
    System.out.print("\t| BUSQUEDA |\nIngrese un nombre a buscar: ");
    return sc.next();
}

// Metodo que muestra una sola nave o no si no existe
public static void mostrarUna(Nave [] flota, int pos, String n)
{
    if (pos == -1)
        System.out.println("No se encontró la nave.\n");
    else
        System.out.println("\t| NAVE CON EL NOMBRE : "+n+" |\n"+flota[pos]+\n");
}
```

- *Luego, este método hace la búsqueda lineal.*

```
//Método para buscar la primera nave con un nombre que se pidió por teclado
public static int busquedaLinealNombre(Nave[] flota, String s)
{
    System.out.println ("| BUSQUEDA LINEAL |");
    for (int i = 0 ; i < flota.length ; i++)
    {
        if ( s.equals( flota [i].getNombre() ) )
            return i;
    }
    return -1;
}
```

- Después, se muestran los métodos que hacen el ordenamiento burbuja por puntos y nombres.

```
//Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosBurbuja(Nave[] flota)
{
    Nave intercambio;
    System.out.println ("| ORDENAMIENTO BURBUJA MENOR A MAYOR |");

    // Siendo un "i" las iteraciones y el maximo posible despues de cada
    // pasada, se establece este algoritmo
    for (int i = flota.length - 1 ; i > 0 ; i--)
        for (int n = 0 ; n < i ; n++)
        {
            if ( flota[n].getPuntos() > flota[n+1].getPuntos() )
            {
                intercambio = flota[n];
                flota[n]     = flota[n+1];
                flota[n+1]   = intercambio;
            }
        }
}
```

```
//Método que ordena por nombre de A a Z
public static void ordenarPorNombreBurbuja(Nave[] flota)
{
    Nave intercambio;
    System.out.println ("| ORDENAMIENTO BURBUJA A - Z |");

    // Siendo un "i" las iteraciones y el maximo posible despues de cada
    // pasada, se establece este algoritmo
    for (int i = flota.length - 1 ; i > 0 ; i--)
        for (int n = 0 ; n < i ; n++)
        {
            int valor = flota[n].getNombre().compareTo(flota[n+1].getNombre());
            if ( valor > 0 )
            {
                intercambio = flota[n];
                flota[n]     = flota[n+1];
                flota[n+1]   = intercambio;
            }
        }
}
```

- *Luego, este método muestra la búsqueda binaria.*

```
//Método para buscar la primera nave con un nombre que se pidió por teclado
public static int busquedaBinariaNombre(Nave[] flota, String s)
{
    int bajo = 0, media, alto = flota.length - 1;

    System.out.println ("| BUSQUEDA BINARIA - NOMBRE |");
    while (bajo <= alto)
    {
        media = (alto + bajo)/2;

        int valor = s.compareTo( flota[media].getNombre() );



        if (valor == 0) return media;
        else if (valor > 0) bajo = media + 1;
        else alto = media - 1;
    }
    return -1;
}
```

- *Después, se muestran los métodos que usa ordenación por selección.*

```
//Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosSeleccion(Nave[] flota)
{
    int menor = 0;
    Nave intercambio;

    System.out.println ("| ORDENAMIENTO SELECCION POR PUNTOS |");
    //ITERACIONES
    for (int i = 0 ; i < flota.length-1 ; i++)
    {
        menor = i;
        //RECORRIDO (La "i" como inicio, actualizado en cada iteracion)
        for (int r = i+1 ; r < flota.length ; r++)
        {
            if ( flota[menor].getPuntos() > flota[r].getPuntos() )
                menor = r;
        }

        // INTERCAMBIO
        intercambio = flota[menor];
        flota[menor] = flota [i];
        flota [i] = intercambio;
    }
}
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 6</p>

```
//Método que ordena por nombre de A a Z
public static void ordenarPorNombreSeleccion(Nave[] flota)
{
    int menor;
    Nave intercambio;



    System.out.println ("| ORDENAMIENTO SELECCION A - Z |");
    //ITERACIONES
    for (int i = 0 ; i < flota.length-1 ; i++)
    {
        menor = i;
        //RECORRIDO (La "i" como inicio, actualizado en cada iteracion)
        for (int r = i+1 ; r < flota.length ; r++)
        {
            int valor = flota[menor].getNombre().compareTo( flota[r].getNombre() );
            if ( valor > 0 )
                menor = r;
        }

        // INTERCAMBIO
        intercambio = flota[menor];
        flota[menor] = flota [i];
        flota [i] = intercambio;
    }
}
```

- Por último, se crean métodos por ordenación de inserción.

```
//Método que muestra las naves ordenadas por número de puntos de mayor a menor
public static void ordenarPorPuntosInsercion(Nave[] flota)
{
    Nave intercambio;

    System.out.println ("| ORDENAMIENTO INSERCIÓN POR PUNTOS |");
    // ITERACIONES
    for (int i = 1 ; i < flota.length ; i++)
    {
        // RECORRIDO (Con detección de recorridos innecesarios)
        for (int r = i ; r > 0 && flota[r-1].getPuntos() < flota[r].getPuntos() ; r--)
        {
            intercambio = flota [r-1];
            flota [r-1] = flota [r];
            flota [r] = intercambio;
        }
    }
}
```

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 7</p>

```
//Método que muestra las naves ordenadas por nombre de Z a A
public static void ordenarPorNombreInsercion(Nave[] flota)
{
    Nave intercambio;

    System.out.println ("| ORDENAMIENTO INSERCIÓN A - Z |");
    // ITERACIONES
    for (int i = 1 ; i < flota.length ; i++)
    {
        // RECORRIDO (Con detección de recorridos innecesarios)
        for
        (
            int r = i ;
            r > 0 && flota[r-1].getNombre().compareTo( flota[r].getNombre() ) < 0;
            r--
        )
        {
            intercambio = flota [r-1];
            flota [r-1] = flota  [r];
            flota  [r] = intercambio;
        }
    }
}
```

## II. PRUEBAS

### EJECUCIÓN:

#### GIT:

- Aquí hice mi primer commit que tiene “Nave.java” y “DemoBatalla.java”, en donde tienen los métodos hasta antes del algoritmo burbuja de A - Z.

```
Usuario248@DESKTOP-TBA9GB6 MINGW64 ~/desktop/git (master)
$ git commit -m "Primer Commit - Hasta 0. Burbuja A-Z"
[master (root-commit) 8bb1ee0] Primer Commit - Hasta 0. Burbuja A-Z
2 files changed, 210 insertions(+)
create mode 100644 DemoBatalla.java
create mode 100644 Nave.java

Usuario248@DESKTOP-TBA9GB6 MINGW64 ~/desktop/git (master)
$ git remote add origin https://github.com/LeonHatches/Laboratorio-04.git

Usuario248@DESKTOP-TBA9GB6 MINGW64 ~/desktop/git (master)
$ git branch -M main

Usuario248@DESKTOP-TBA9GB6 MINGW64 ~/desktop/git (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 2.13 KiB | 2.13 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/LeonHatches/Laboratorio-04.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 8</p>

- *Este es mi 2do commit, en los cuales tienen las resoluciones de los métodos de ordenamiento.*

```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/git/repositorios/laboratorio04/Laboratorio-04 (main)
$ git commit -m "Segundo commit, todos los metodos de ordenamiento realizados"
[main 0ebf04b] Segundo commit, todos los metodos de ordenamiento realizados
1 file changed, 219 insertions(+), 48 deletions(-)

Equipo@DESKTOP-VTEJTT1 MINGW32 ~/git/repositorios/laboratorio04/Laboratorio-04 (main)
$ git branch -vv
* main 0ebf04b [origin/main: ahead 1] Segundo commit, todos los metodos de ordenamiento realizad
os

Equipo@DESKTOP-VTEJTT1 MINGW32 ~/git/repositorios/laboratorio04/Laboratorio-04 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.87 KiB | 318.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/LeonHatches/Laboratorio-04.git
8bb1ee0..0ebf04b main -> main
```

- *2do Commit de nave con un comentario de sin cambios.*

```
Equipo@DESKTOP-VTEJTT1 MINGW32 ~/git/repositorios/laboratorio04/Laboratorio-04 (main)
$ git commit -m "Segundo commit, nave sin cambios"
[main bd215af] Segundo commit, nave sin cambios
1 file changed, 1 insertion(+), 1 deletion(-)

Equipo@DESKTOP-VTEJTT1 MINGW32 ~/git/repositorios/laboratorio04/Laboratorio-04 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 373 bytes | 93.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/LeonHatches/Laboratorio-04.git
0ebf04b..bd215af main -> main
```

### EJECUCIÓN:

- *Búsqueda lineal:*

```
| BUSQUEDA LINEAL |
Ingrese un nombre a buscar: A
| NAVE CON EL NOMBRE : A |
NOMBRE:      A
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      22
```



- *Ordenamiento burbuja:*

```
| ORDENAMIENTO BURBUJA MENOR A MAYOR |
| NAVE 1 |
NOMBRE:      C
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      21

| NAVE 2 |
NOMBRE:      A
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      22

| NAVE 3 |
NOMBRE:      B
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      23
```

```
| ORDENAMIENTO BURBUJA A - Z |
| NAVE 1 |
NOMBRE:      A
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      22

| NAVE 2 |
NOMBRE:      A
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      25

| NAVE 3 |
NOMBRE:      B
FILA:         1
COLUMNA:     1
ESTADO:      false
PUNTOS:      23

| NAVE 4 |
NOMBRE:      C
```

- *Búsqueda binaria:*

```
Ingrese un nombre a buscar: C
|| BUSQUEDA BINARIA - NOMBRE |
| NAVE CON EL NOMBRE : C |
NOMBRE:      C
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      21
```

- *Ordenamiento por selección:*

```
| ORDENAMIENTO SELECCION POR PUNTOS |
| NAVE 1 |
NOMBRE:      C
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      21

| NAVE 2 |
NOMBRE:      A
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      22

| NAVE 3 |
NOMBRE:      B
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      23
```

```
| ORDENAMIENTO SELECCION A - Z |
| NAVE 1 |
NOMBRE:      A
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      22

| NAVE 2 |
NOMBRE:      A
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      25

| NAVE 3 |
NOMBRE:      B
FILA:        1
COLUMNA:     1
ESTADO:      false
PUNTOS:      23
```

- *Ordenamiento por inserción:*

```
| ORDENAMIENTO INSERCIÓN POR PUNTOS |  
| NAVE 1 |  
NOMBRE: A  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 25  
  
| NAVE 2 |  
NOMBRE: D  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 24  
  
| NAVE 3 |  
NOMBRE: B  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 23
```

```
| ORDENAMIENTO INSERCIÓN A - Z |  
| NAVE 1 |  
NOMBRE: D  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 24  
  
| NAVE 2 |  
NOMBRE: C  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 21  
  
| NAVE 3 |  
NOMBRE: B  
FILA: 1  
COLUMNA: 1  
ESTADO: false  
PUNTOS: 23
```

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 12

### III. CUESTIONARIO:

*Colocar la evidencia de las respuestas realizadas al cuestionario enunciado en la guía práctica de laboratorio.*

*Preguntas:*

*¿Con qué valores comprobaste que tu práctica estuviera correcta?*

- *Con valores de entrada tipo String (cadena), que son representadas por nombres; por valores booleanos (boolean) como estados o con valores de entrada tipo entero (int) como las filas, columnas y puntos.*

*¿Qué resultado esperabas obtener para cada valor de entrada?*

- *Esperaba unos mismos resultados de salida, pero con diferente algoritmo de ordenamiento.*

*¿Qué valor o comportamiento obtuviste para cada valor de entrada?*

- *Obtuve los valores que deseaba para satisfacer el enunciado.*

## CONCLUSIONES

*En conclusión, para que el código del enunciado pueda tener un buen funcionamiento, el uso de arreglos será necesario de implementar en los datos de las naves como soldados, así también para el uso de los métodos que devuelvan los arreglos ordenados de diferentes formas, así mostremos gran optimización, accesibilidad y orden al momento de programar. Así también, la importancia de los métodos al instante de trabajar y tener un código que tenga una facilidad de observación al momento de las correcciones del código o en este caso, la visualización de algoritmos de ordenamiento.*

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 13

### METODOLOGÍA DE TRABAJO

*Colocar la metodología de trabajo que ha utilizado el estudiante o el grupo para resolver la práctica, es decir el procedimiento/secuencia de pasos en forma general.*

- *La secuencia que utilicé fue, primero, analizar el código base ya proporcionado o que había realizado y tratar de darle parte de la solución en mi mente. Segundo, me dediqué a armar la estructura de los algoritmos faltantes, los distintos métodos. Tercero, analizar errores y formas de optimizar el código para que se visualice de una manera entendible. Cuarto y último, compilar o correr el código para visualizar su funcionamiento o en dado caso no funcione, volver al paso 3 para corregir.*

### REFERENCIAS Y BIBLIOGRAFÍA

*Ninguna referencia externa.*

## RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Tabla 1: 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

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	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1	
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 están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
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	4	
TOTAL		20		18	

Tabla 2: Rúbrica para contenido del Informe y demostración