INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de Programación II*				
TÍTULO DE LA PRÁCTICA:	*Arreglos de Objetos, Búsquedas y Ordenamientos*				
NÚMERO DE PRÁCTICA:	*4*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*II*
FECHA DE PRESENTACIÓN	*12/10/2024*	**HORA DE PRESENTACIÓN**	*15/20/00*		
INTEGRANTE (s) *Usiel Suriel Quispe Puma*				**NOTA (0-20)**	*Nota colocada por el docente*
DOCENTE(s): *Mg.Lino José Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

Ejercicio 1

Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3 3. Completar el Código de la clase DemoBatalla

Código:

- **Clase DemoBatalla (Prinicipal):**

```java
package Laboratorio_04;

//Laboratorio Nº4 - Ejercicio 1
//Autor: Usiel Suriel Quispe Puma

import java.util.*;

public class DemoBatalla {

    public static void main(String[] args) {
        Nave[] misNaves = new Nave[8];
        Scanner sc = new Scanner(source: 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(s: "Nombre: ");
            nomb = sc.next();
            System.out.print(s: "Fila :");
            fil = sc.nextInt();
            System.out.print(s: "Columna: ");
            col = sc.next();
            System.out.print(s: "Estado(true o false) : ");
            est = sc.nextBoolean();
            System.out.print(s: "Puntos: ");
            punt = sc.nextInt();

            misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves

            misNaves[i].setNombre(n: nomb);
            misNaves[i].setFila(f: fil);
            misNaves[i].setColumna(c: col);
            misNaves[i].setEstado(e: est);
            misNaves[i].setPuntos(p: punt);
            System.out.println(x: "");
        }

        System.out.println(x: "\n\tNaves creadas");
        mostrarNaves(Flota:misNaves);

        //Mostrar naves     que tenga el mismo nombre
        System.out.println(x: "\tNaves con el mismo nombre\n");
```

```java
package Laboratorio_04;

//Laboratorio Nº4 - Ejercicio 1
//Autor: Usiel Suriel Quispe Puma

import java.util.*;

public class DemoBatalla {

    public static void main(String[] args) {
        Nave[] misNaves = new Nave[8];
        Scanner sc = new Scanner(source: 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(s: "Nombre: ");
            nomb = sc.next();
            System.out.print(s: "Fila :");
            fil = sc.nextInt();
            System.out.print(s: "Columna: ");
            col = sc.next();
            System.out.print(s: "Estado(true o false) : ");
            est = sc.nextBoolean();
            System.out.print(s: "Puntos: ");
            punt = sc.nextInt();

            misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves

            misNaves[i].setNombre(n: nomb);
            misNaves[i].setFila(f: fil);
            misNaves[i].setColumna(c: col);
            misNaves[i].setEstado(e: est);
            misNaves[i].setPuntos(p: punt);
            System.out.println(x: "");
        }

        System.out.println(x: "\n\tNaves creadas");
        mostrarNaves(Flota:misNaves);

        //Mostrar naves     que tenga el mismo nombre
        System.out.println(x: "\tNaves con el mismo nombre\n");
```

```
        System.out.print(s: "Ingrese el nombre :");
        nomb = sc.next();
        mostrarPorNombre(flota:misNaves, nombre: nomb);

        //Mostrar naves que tengan puntos inferiores a los puntos ingresados
        System.out.println(x: "\n\tNaves que tienen puntos inferiores \n ");
        System.out.print(s: "Ingrese la cantidad de puntos minimo: ");
        punt = sc.nextInt();
        mostrarPorPuntos(flota:misNaves, puntos: punt);

        //Mostrar naves con el mayor puntaje
        System.out.println(x: "\n\tNave con mayor numero de puntos\n ");
        mostrarMayorPuntos(flota:misNaves);

        //leer un nombre
        //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
        System.out.println(x: "\n\tBusqueda lineal ");
        System.out.print(s: "\nIngrese el nombre de la nave que busca : ");
        nomb = sc.next();
        int pos = busquedaLinealNombre(flota:misNaves, s: nomb);
        if (pos != -1) {
            misNaves[pos].mostrarDatos();
        } else {
            System.out.println(x: "No econtrado");
        }

        System.out.println(x: "\n\tOrdenamiento por la cantidad de puntos (Burbuja) ");
        ordenarPorPuntosBurbuja(flota:misNaves);
        mostrarNaves(flota:misNaves);

        System.out.println(x: "\n\tOrdenamiento por nombre( Burbuja)");
        ordenarPorNombreBurbuja(flota:misNaves);
        mostrarNaves(flota:misNaves);

        //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
        System.out.println(x: "\n\tBusqueda binaria por nombre ");
        System.out.print(s: "Ingrese el nombre de la nave que busca : ");
        nomb = sc.next();
        pos = busquedaBinariaNombre(flota:misNaves, s: nomb);
        if (pos != -1) {
            misNaves[pos].mostrarDatos();
        } else {
            System.out.println(x: "No econtrado");
```

```
        }

        System.out.println( x: "\n\tOrdenamiento por puntos( Seleccion)");
        ordenarPorPuntosSeleccion( flota: misNaves);
        mostrarNaves( flota: misNaves);

        System.out.println( x: "\n\tOrdenamiento por nombres( Seleccion)");
        ordenarPorNombreSeleccion( flota: misNaves);
        mostrarNaves( flota: misNaves);

        System.out.println( x: "\n\tOrdenamiento por puntos descendentemente( Insercion)");
        ordenarPorPuntosInsercion( flota: misNaves);
        mostrarNaves( flota: misNaves);

        System.out.println( x: "\n\tOrdenamiento por nombre descendentemente (Insercion)");
        ordenarPorNombreInsercion( flota: misNaves);
        mostrarNaves( flota: misNaves);
    }

    //Método para mostrar todas las naves
    public static void mostrarNaves(Nave[] flota) {
        for (int i = 0; i < flota.length; i++) {
            flota[i].mostrarDatos();
            System.out.println( x: "");

        }

    }

    //Método para mostrar todas las naves de un nombre que se pide por teclado
    public static void mostrarPorNombre(Nave[] flota, String nombre) {
        for (int i = 0; i < flota.length; i++) {
            if (flota[i] != null && flota[i].getNombre().equals( anObject: nombre)) {

                flota[i].mostrarDatos();
                System.out.println( x: "");
            }

        }
    }
```

```java
    //Método para mostrar todas las naves con un número de puntos inferior o igual
    //al número de puntos que se pide por teclado
    public static void mostrarPorPuntos(Nave[] flota, int puntos) {
        for (int i = 0; i < flota.length; i++) {
            if (flota[i].getPuntos() <= puntos) {
                flota[i].mostrarDatos();
                System.out.println("");
            }

        }

    }

//Método que devuelve la Nave con mayor número de Puntos
    public static void mostrarMayorPuntos(Nave[] flota) {
        int mayorPuntos = 0;
        int nave = 0;
        for (int i = 0; i < flota.length; i++) {
            if (flota[i].getPuntos() > mayorPuntos) {
                mayorPuntos = flota[i].getPuntos();
                nave = i;

            }

        }
        flota[nave].mostrarDatos();

    }
    //Método para buscar la primera nave con un nombre que se pidió por teclado

    public static int busquedaLinealNombre(Nave[] flota, String s) {
        for (int i = 0; i < flota.length; i++) {
            if (flota[i] != null && flota[i].getNombre().equals(s)) {

                return i;
            }

        }
        return -1;

    }
    //Método que ordena por número de puntos de menor a mayor
```

```java
public static void ordenarPorPuntosBurbuja(Nave[] flota) {

    for (int i = 0; i < flota.length - 1; i++) {
        for (int j = 0; j < flota.length - 1 - i; j++) {
            if (flota[j].getPuntos() > flota[j + 1].getPuntos()) {
                Nave temp = flota[j];
                flota[j] = flota[j + 1];
                flota[j + 1] = temp;
            }
        }
    }

}

//Método que ordena por nombre de A a Z
public static void ordenarPorNombreBurbuja(Nave[] flota) {
    for (int i = 0; i < flota.length; i++) {
        for (int j = 0; j < flota.length - 1 - i; j++) {
            if (flota[j].getNombre().compareTo(anotherString: flota[j + 1].getNombre()) > 0) {
                Nave temp;
                temp = flota[j];
                flota[j] = flota[j + 1];
                flota[j + 1] = temp;

            }

        }

    }

}
//Método para buscar la primera nave con un nombre que se pidió por teclado

public static int busquedaBinariaNombre(Nave[] flota, String s) {
    int baja = 0;
    int alta = flota.length - 1;
    while (baja <= alta) {
        int media = (baja + alta) / 2;
        if (flota[media] != null && flota[media].getNombre().equals(anObject:s)) {
            return media;
        } else if (flota[media] != null && s.compareTo(anotherString: flota[media].getNombre()) < 0) {
            alta = media - 1;
        } else {
```

```
                baja = media + 1;
            }
        }
        return -1;
    }
    //Método que ordena por número de puntos de menor a mayor

    public static void ordenarPorPuntosSeleccion(Nave[] flota) {
        for (int i = 0; i < flota.length - 1; i++) {
            for (int j = i + 1; j < flota.length; j++) {
                if (flota[i].getPuntos() > flota[j].getPuntos()) {
                    Nave temp = flota[i];
                    flota[i] = flota[j];
                    flota[j] = temp;
                }

            }

        }

    }
    //Método que ordena por nombre de A a Z

    public static void ordenarPorNombreSeleccion(Nave[] flota) {

        for (int i = 0; i < flota.length - 1; i++) {
            for (int j = i + 1; j < flota.length; j++) {
                if (flota[i].getNombre().compareTo(anotherString: flota[j].getNombre()) > 0) {
                    Nave temp = flota[i];
                    flota[i] = flota[j];
                    flota[j] = temp;
                }

            }

        }

    }
    //Método que muestra las naves ordenadas por número de puntos de mayor a menor

    public static void ordenarPorPuntosInsercion(Nave[] flota) {
        for (int i = 1; i < flota.length; i++) {
            Nave temp = flota[i];
```

```
            int j = i - 1;

            while (j > -1 && flota[j].getPuntos() < temp.getPuntos()) {
                flota[j + 1] = flota[j];
                j -= 1;

            }
            flota[j + 1] = temp;

        }

    }
    //Método que muestra las naves ordenadas por nombre de Z a A
    public static void ordenarPorNombreInsercion(Nave[] flota) {
        for (int i = 1; i < flota.length; i++) {
            Nave temp = flota[i];
            int j = i - 1;

            while (j >= 0 && flota[j] != null && flota[j].getNombre().compareTo(anotherString: temp.getNombre()) < 0) {
                flota[j + 1] = flota[j];
                j--;
            }
            flota[j + 1] = temp;
        }
    }

}
```

- **Clase Nave:**

```
package Laboratorio_04;
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;
    }
    //Mètodo para mostrar los datos de cada nave
     public void mostrarDatos() {
        System.out.println("Nombre : " + nombre);
        System.out.println("fila: " + fila + "     Columna : " + columna);
        System.out.println("puntos : " + puntos);
        System.out.println("Estado : " + estado);
    }

}
```

II. PRUEBAS

```
Nave 1
Nombre: alpha
Fila :4
Columna: a
Estado(true o false) : true
Puntos: 20

Nave 2
Nombre: alpha
Fila :3
Columna: f
Estado(true o false) : true
Puntos: 50

Nave 3
Nombre: f23
Fila :10
Columna: g
Estado(true o false) : true
Puntos: 33

Nave 4
Nombre: max
Fila :1
Columna: c
Estado(true o false) : true
Puntos: 34

Nave 5
Nombre: rex
Fila :8
Columna: f
Estado(true o false) : true
Puntos: 70

Nave 6
Nombre: drins
Fila :6
Columna: d
Estado(true o false) : true
Puntos: 56

Nave 7
Nombre: yin23
Fila :4
Columna: e
Estado(true o false) : true
Puntos: 69

Nave 8
Nombre: juan
Fila :4
Columna: g
Estado(true o false) : true
Puntos: 8


        Naves creadas
Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true
```

```
Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true

Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true
```

```
Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

        Naves con el mismo nombre

Ingrese el nombre :alpha
Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true


        Naves que tienen puntos inferiores

Ingrese la cantidad de puntos minimo: 30
Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true
```

```
        Nave con mayor numero de puntos

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

        Busqueda lineal

Ingrese el nombre de la nave que busca : f33
No econtrado

        Ordenamiento por la cantidad de puntos (Burbuja)
Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true
```


```
        Nave con mayor numero de puntos

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

        Busqueda lineal

Ingrese el nombre de la nave que busca : f33
No econtrado

        Ordenamiento por la cantidad de puntos (Burbuja)
Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true
```

```
Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true
```

```
        Busqueda binaria por nombre
Ingrese el nombre de la nave que busca : max
Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

        Ordenamiento por puntos( Seleccion)
Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true
```

```
Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true

Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true


        Ordenamiento por nombres( Seleccion)
Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true

Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true

Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true
```

```
Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true


        Ordenamiento por puntos descendentemente( Insercion)
Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true
```

```
Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true

Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true

Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true

Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true
```

```
Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true


       Ordenamiento por nombre descendentemente (Insercion)
Nombre : yin23
fila: 4     Columna : e
puntos : 69
Estado : true

Nombre : rex
fila: 8     Columna : f
puntos : 70
Estado : true

Nombre : max
fila: 1     Columna : c
puntos : 34
Estado : true

Nombre : juan
fila: 4     Columna : g
puntos : 8
Estado : true

Nombre : f23
fila: 10     Columna : g
puntos : 33
Estado : true
```

```
Nombre : drins
fila: 6     Columna : d
puntos : 56
Estado : true

Nombre : alpha
fila: 3     Columna : f
puntos : 50
Estado : true

Nombre : alpha
fila: 4     Columna : a
puntos : 20
Estado : true
```

- **Commit:**

```
Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git add src/Laboratorio_04

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git commit -m "Se agrega el laboratorio 4 en la carpeta src"
[master 3cbd4d0] Se agrega el laboratorio 4 en la carpeta src
 2 files changed, 342 insertions(+)
 create mode 100644 src/Laboratorio_04/DemoBatalla.java
 create mode 100644 src/Laboratorio_04/Nave.java

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 2.71 KiB | 308.00 KiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/usiel33/Laboratorios_FP2.git
   3b7bba3..3cbd4d0  master -> master

Quisp@DESKTOP-HDBB5AV MINGW32 ~/OneDrive/Documentos/Mi repositorio/Laboratorios_FP2 (master)
$ git log
commit 3cbd4d09f3eb1cba274cd84c17f6f26050b5a81d (HEAD -> master, origin/master, origin/HEAD)
Author: usiel33 <uquispep@unsa.edu.pe>
Date:   Sat Oct 12 15:05:42 2024 -0500

    Se agrega el laboratorio 4 en la carpeta src
```

III. Rubrica

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		3	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4		2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2		2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2		2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2		2	
6. Fechas	Las fechas de modificación del código fuente	2		4	

	están dentro de los plazos de fecha de entrega establecidos.				
7. Ortografía	El documento no muestra errores ortográficos.	2		1	
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		2	
TOTAL		20		18	

CONCLUSIONES

En este laboratorio vimos el uso de la poo, el arreglo de objeto y ordenamientos , los cuales son importante en el desarrollo de programas ya que mejora la legibilidad y permite separar el código en métodos para reutilizar códigos , además implementar algoritmos de ordenamientos , que ya están definidas, hace que estos procesos sean más eficientes.

METODOLOGÍA DE TRABAJO

primero revise el código ya escrito para poder entender el funcionamiento y los requerimientos que necesita, luego complete con código cada método haciendo que cumpla con el requerimiento.
finalmente hice pruebas para verificar el correcto funcionamiento del código .

REFERENCIAS Y BIBLIOGRAFÍA

Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE