INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	*FUNDAMENTOS DE PROGRAMACIÓN 2*				
TÍTULO DE LA PRÁCTICA:	*Arreglos Estándar.*				
NÚMERO DE PRÁCTICA:	*1*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*2024-B*
FECHA DE PRESENTACIÓN	*22/09/2024*	**HORA DE PRESENTACIÓN**	*18/30/12*		
INTEGRANTE (s) *Riveros Vilca Alberth Edwar*				**NOTA (0-20)**	
DOCENTE(s): *Ing. Lino Jose Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

Actividad 1: *Analice, complete y pruebe el Código de la clase DemoBatalla.*

CLASE Nave:

```java
public class Nave {  29 usages  new *
    private String nombre;  3 usages
    private int fila;  3 usages
    private String columna;  3 usages
    private boolean estado;  3 usages
    private int puntos;  3 usages
    // Metodos mutadores
    public void setNombre( String n){  1 usage  new *
        nombre = n;
    }
    public void setFila(int f){  1 usage  new *
        fila = f;
    }
    public void setColumna(String c){  1 usage  new *
        columna = c;
    }
    public void setEstado(boolean e){  1 usage  new *
        estado = e;
    }
    public void setPuntos(int p){  1 usage  new *
        puntos = p;
    }
    // Metodos accesores
    public String getNombre(){  10 usages  new *
        return nombre;
    }
    public int getFila(){  no usages  new *
        return fila;
    }
    public String getColumna(){  no usages  new *
        return columna;
    }
    public boolean getEstado(){  1 usage  new *
        return estado;
    }
    public int getPuntos(){  9 usages  new *
        return puntos;
    }
    @Override  new *
    public String toString(){
        return "[Nombre: "+nombre+"\tFila: "+fila+"\tColumna: "+columna+"\tEstado: "+estado+"\tPuntos: "+puntos
                +"]";
    }
}
```

CLASE DemoBatalla:

```java
import java.util.*;
public class DemoBatalla {  rivX241 *
    public static void main(String[] args) {  rivX241 *
        Nave[] misNaves = new Nave[8];
        Scanner sc = new Scanner(System.in);
        String nomb, col;
        int fil, punt;
        boolean est;

        // Creación de las naves
        for (int i = 0; i < misNaves.length; i++) {
            System.out.println("Nave " + (i + 1));
            System.out.print("Nombre: ");
            nomb = sc.next();
            System.out.print("Fila: ");
            fil = sc.nextInt();
            System.out.print("Columna: ");
            col = sc.next();
            System.out.print("Estado: ");
            est = sc.nextBoolean();
            System.out.print("Puntos: ");
            punt = sc.nextInt();
            misNaves[i] = new Nave();
            misNaves[i].setNombre(nomb);
            misNaves[i].setFila(fil);
            misNaves[i].setColumna(col);
            misNaves[i].setEstado(est);
            misNaves[i].setPuntos(punt);
        }

        System.out.println("\nNaves creadas:");
        mostrarNaves(misNaves);

        // Clonación para preservar el orden original
        Nave[] copiaNaves;

        // Ordenar por puntos (burbuja)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por puntos (burbuja):");
        ordenarPorPuntosBurbuja(copiaNaves);
        mostrarNaves(copiaNaves);

        // Ordenar por nombre (burbuja)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por nombre (burbuja):");
        ordenarPorNombreBurbuja(copiaNaves);
        mostrarNaves(copiaNaves);

        // Ordenar por puntos (selección)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por puntos (selección):");
        ordenarPorPuntosSeleccion(copiaNaves);
        mostrarNaves(copiaNaves);
```

```java
        // Ordenar por nombre (selección)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por nombre (selección):");
        ordenarPorNombreSeleccion(copiaNaves);
        mostrarNaves(copiaNaves);

        // Ordenar por puntos (inserción)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por puntos (inserción):");
        ordenarPorPuntosInsercion(copiaNaves);
        mostrarNaves(copiaNaves);

        // Ordenar por nombre (inserción)
        copiaNaves = misNaves.clone();
        System.out.println("\nOrdenando por nombre (inserción):");
        ordenarPorNombreInsercion(copiaNaves);
        mostrarNaves(copiaNaves);

        // Búsquedas para probar los métodos de búsqueda
        System.out.print("\nBuscar nave por nombre (búsqueda lineal): ");
        String nombreBusqueda = sc.next();
        int indexLineal = busquedaLinealNombre(copiaNaves, nombreBusqueda);
        System.out.println(indexLineal >= 0 ? "Nave encontrada en índice " + indexLineal : "Nave no encontrada");

        System.out.print("\nBuscar nave por nombre (búsqueda binaria): ");
        nombreBusqueda = sc.next();
        int indexBinaria = busquedaBinariaNombre(copiaNaves, nombreBusqueda);
        System.out.println(indexBinaria >= 0 ? "Nave encontrada en índice " + indexBinaria : "Nave no encontrada");
    }
    public static void mostrarNaves(Nave [] flota){  7 usages  rivX241
        for (Nave nave : flota) {
            System.out.println(nave.toString());
        }
    }
    public static void mostrarPorNombre(Nave [] flota){  no usages  rivX241
        Scanner sc = new Scanner(System.in);
        System.out.print("Ingrese el nombre de la nave: ");
                String name = sc.nextLine();
        for(Nave naveBuscada : flota){
            if(naveBuscada.getNombre().equals(name)){
                System.out.println(naveBuscada);
            }
        }
    }
    public static void mostrarPorPuntos(Nave [] flota){  no usages  rivX241
        Scanner sc = new Scanner(System.in);
        System.out.print("Ingrese el número de puntos para buscar las naves inferiores a este: ");
        int puntos = sc.nextInt();
        for(Nave naveFiltrada : flota){
            if(naveFiltrada.getPuntos()<=puntos){
                System.out.println(naveFiltrada);
            }
        }
    }
```

```java
public static String mostrarMayorPuntos(Nave [] flota){  no usages  rivX241
    int index = 0;
    for(int i=1;i< flota.length;i++){
        if(flota[i].getPuntos()>flota[i-1].getPuntos()&&flota[i].getEstado()){
            index = i;
        }
    }
    return flota[index].toString();
}
public static Nave[] nuevaFlota(Nave[] flota){  no usages  rivX241
    Nave[] nuevaFlota = new Nave[flota.length];
    Random rand = new Random();
    int randomIndex;
    for(int i = 0;i< flota.length;i++){
        randomIndex = rand.nextInt(flota.length);
        nuevaFlota[i] = flota[randomIndex];
    }
    return nuevaFlota;
}
//Método para buscar la primera nave con un nombre que se pidió por teclado
public static int busquedaLinealNombre(Nave[] flota, String s){  1 usage  rivX24
    for(int i=0;i<=flota.length;i++){
      if(flota[i].getNombre().equals(s)){
        return i;
      }
    }
    return -1;
}
//Método que ordena por número de puntos de menor a mayor
public static void ordenarPorPuntosBurbuja(Nave[] flota){  1 usage  rivX241
    for(int i=1;i<flota.length;i++){
        for(int j=0;j<flota.length-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){  1 usage  rivX241
    for(int i=1;i<flota.length;i++){
        for(int j=0;j<flota.length-i;j++){
            if(flota[j].getNombre().compareTo(flota[j+1].getNombre())>0){
                Nave 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
```

```java
public static int busquedaBinariaNombre(Nave[] flota, String s){  1 usage
    int alta=flota.length-1,baja=0,media;
    while(baja<=alta){
        media = (alta+baja)/2;
        if(flota[media].getNombre().equals(s)){
            return media;
        }else if(flota[media].getNombre().compareTo(s)>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){  1 usage  rivX2
    for(int i=0;i<flota.length;i++){
        int min = i;
        for(int j=i+1;j<flota.length;j++){
            if(flota[j].getPuntos()<flota[min].getPuntos()){
                min = j;
            }
        }
        if(i != min){
            Nave temp = flota[i];
            flota[i] = flota[min];
            flota[min] = temp;
        }
    }
}
//Método que ordena por nombre de A a Z
public static void ordenarPorNombreSeleccion(Nave[] flota){  1 usage  rivX2
    for(int i=0;i<flota.length;i++){
        int min = i;
        for(int j=i+1;j<flota.length;j++){
            if(flota[j].getNombre().compareTo(flota[min].getNombre())<0){
                min = j;
            }
        }
        if(i != min){
            Nave temp = flota[i];
            flota[i] = flota[min];
            flota[min] = temp;
        }
    }
}
```

```java
//Método que muestra las naves ordenadas por número de puntos de mayor a menor
    public static void ordenarPorPuntosInsercion(Nave[] flota){  1 usage  rivX241
        for(int i=1;i<flota.length;i++){
            Nave actual = flota[i];
            int j=i-1;
            while(j>=0 && flota[i].getPuntos()>actual.getPuntos()){
                flota[j+1] = flota[j];
                j++;
            }
            flota[j+1] = actual;
        }
    }
//Método que muestra las naves ordenadas por nombre de A a Z
    public static void ordenarPorNombreInsercion(Nave[] flota){  1 usage  rivX241 *
        for(int i=1;i<flota.length;i++){
            Nave actual = flota[i];
            int j=i-1;
            while(j>=0 && flota[j].getNombre().compareTo(actual.getNombre())>0){
                flota[j+1] = flota[j];
                j--;
            }
            flota[j+1] = actual;
        }
    }
}
```

II. PRUEBAS

¿Con que valores comprobaste que tu práctica estuviera correcta?

De acuerdo al problema con valores de tipo String que equivalen a cadenas de caracteres.

¿Qué resultado esperabas obtener para cada valor de entrada?

Que se asignara a una variable que representa el nombre del soldado,luego de ello buscar y recibir el resultado de la búsqueda con el arreglo previamente ordenado por los métodos de inserción,burbuja y selección y con la ayuda del método compareTo() .

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

Los valores almacenados se imprimieron por pontalla y se mostró los atributos de dicho soldado buscado.

EJECUCIÓN:

```
Nave 1
Nombre: Luis
Fila: 3
Columna: 5
Estado: true
Puntos: 13
Nave 2
Nombre: Alberto
Fila: 6
Columna: 4
Estado: true
Puntos: 22
Nave 3
Nombre: Juan
Fila: 5
Columna: 9
Estado: true
Puntos: 43
Nave 4
Nombre: Ricardo
Fila: 4
Columna: 1
Estado: true
Puntos: 29
Nave 5
Nombre: Rodrigo
Fila: 2
Columna: 8
Estado: true
Puntos: 20
Nave 6
Nombre: Edwar
Fila: 3
Columna: 6
Estado: true
Puntos: 52
Nave 7
Nombre: Pedro
Fila: 5
Columna: 6
Estado: true
Puntos: 45
Nave 8
Nombre: Gabriel
Fila: 9
Columna: 2
Estado: true
Puntos: 43
```

```
Naves creadas:
[Nombre: Luis   Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Juan   Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
```

```
Ordenando por puntos (burbuja):
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Juan   Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
[Nombre: Luis   Fila: 3 Columna: 5  Estado: true    Puntos: 13]

Ordenando por nombre (burbuja):
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
[Nombre: Juan   Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Luis   Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
```

```
Ordenando por puntos (selección):
[Nombre: Luis   Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Juan   Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]

Ordenando por nombre (selección):
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
[Nombre: Juan   Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Luis   Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
```

```
Ordenando por puntos (inserción):
[Nombre: Luis    Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Juan    Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]

Ordenando por nombre (inserción):
[Nombre: Alberto    Fila: 6 Columna: 4  Estado: true    Puntos: 22]
[Nombre: Edwar  Fila: 3 Columna: 6  Estado: true    Puntos: 52]
[Nombre: Gabriel    Fila: 9 Columna: 2  Estado: true    Puntos: 43]
[Nombre: Juan    Fila: 5 Columna: 9  Estado: true    Puntos: 43]
[Nombre: Luis    Fila: 3 Columna: 5  Estado: true    Puntos: 13]
[Nombre: Pedro  Fila: 5 Columna: 6  Estado: true    Puntos: 45]
[Nombre: Ricardo    Fila: 4 Columna: 1  Estado: true    Puntos: 29]
[Nombre: Rodrigo    Fila: 2 Columna: 8  Estado: true    Puntos: 20]
```

```
Buscar nave por nombre (búsqueda lineal): Gabriel
Nave encontrada en índice 2

Buscar nave por nombre (búsqueda binaria): Luis
Nave encontrada en índice 4
```

III. CUESTIONARIO:

CONCLUSIONES

El uso de algoritmos de ordenamientos aplicados a arreglos de objetos nos permite realizar búsquedas de los objetos y los atributos correspondientes que necesitamos y nos ayuda a organizar los datos de manera decreciente o creciente de acuerdo a los requisitos que tengamos.

METODOLOGÍA DE TRABAJO

1.Realice la lectura del problema vi lo requisitos y restricciones del mismo para plantear una solución.

2.Identifique las herramientas y la lógica requerida para la solución del mismo.
3.Codifique el problema para luego testearlo con los datos de entrada.
4.Probe la solución y corregí los errores que estuvieron presentes.

REFERENCIAS Y BIBLIOGRAFÍA

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

E. G. Castro Gutiérrez y M. W. Aedo López, *Fundamentos de programación 2: tópicos de programación orientada a objetos*, 1st ed. Arequipa, Perú: Universidad Nacional de San Agustín, 2021. ISBN: 978-612-5035-20-2. 170 p. [Enseñanza universitaria o superior]. Impreso, tapa blanda, 20.5 x 29 cm.

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	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	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
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	2	
TOTAL		20		18	