Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Practica de Laboratorio 4:
Arreglos de Objetos, Búsquedas y Ordenamientos

Nombre: Jhonatan Benjamin Mamani Céspedes **CUI: 20232188**

1. Cree un Proyecto llamado Laboratorio4
2. Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3
3. Completar el Código de la clase DemoBatalla

Código del programa

DemoBatalla.java 5, U ● Nave.java U

DemoBatalla.java > ...

```java
// LABORATORIO N° 4 - EJERCICIO 1
// AUTOR: JHONATAN BENJAMIN MAMANI CÉSPEDES
// TIEMPO: 49 MINUTOS
import java.util.*;
public class DemoBatalla {
    Run | Debug
    public static void main(String [] args){
        Nave [] misNaves = new Nave[5];
        Scanner sc = new Scanner(System.in);
        String nomb, col, nombre;
        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 (Entre 1 y 10): ");
            do
                fil = sc.nextInt();
            while (fil > 10 || fil < 1);
            System.out.print(s:"Columna (De la A a la J): ");
            String abc = "ABCDEFGHIJ";
            do
                col = sc.next().toUpperCase();
            while (abc.indexOf(col) == -1);
            System.out.print(s:"Estado: ");
            est = sc.nextBoolean();
            System.out.print(s:"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);
        }
```

```
38          System.out.println(x:"\nGenerando naves...");
39          System.out.println(x:"\nNaves creadas:");
40          mostrarNaves(misNaves);
41          System.out.print(s:"\nIntroduzca el nombre de la nave que desea buscar: ");
42          mostrarPorNombre(misNaves);
43          System.out.print(s:"\nIntroduzca los puntos maximos que puede tener una nave: ");
44          mostrarPorPuntos(misNaves);
45          System.out.println(x:"\nEn su flota, la nave con el mayor numero de puntos es:");
46          mostrarMayorPuntos(misNaves);
47
48          System.out.println(x:"\nBUSQUEDA LINEAL:");
49          System.out.print(s:"\nIngrese el nombre de la nave que quiere buscar: ");
50          nombre = sc.next();
51          int pos = busquedaLinealNombre(misNaves, nombre);
52          if (pos != -1){
53              mostrarNaveEspecifica(misNaves, pos);
54          } else {
55              System.out.println(x:"No se ha encontrado la nave");
56          }
57
58          // BURBUJA
59          System.out.println(x:"\nOrdenamiento de las naves por el algoritmo Burbuja:");
60          System.out.println(x:"\nPor puntos:");
61          ordenarPorPuntosBurbuja(misNaves);
62          mostrarNaves(misNaves);
63          System.out.println(x:"\nPor nombres:");
64          ordenarPorNombreBurbuja(misNaves);
65          mostrarNaves(misNaves);
66
```

```
66
67          System.out.println(x:"\nSe han ordenado exitosamente las naves, puede buscar por la forma binaria:");
68          System.out.println(x:"\nBUSQUEDA BINARIA:");
69          System.out.print(s:"\nIngrese el nombre de la nave que quiere buscar: ");
70          nombre = sc.next();
71          int pos1 = busquedaBinariaNombre(misNaves, nombre);
72          if (pos1 != -1)
73              mostrarNaveEspecifica(misNaves, pos1);
74          else
75              System.out.println(x:"No se ha encontrado la nave");
76          System.out.println(x:"\nORDENAMIENTO POR PUNTOS (SELECCIÓN):");
77          ordenarPorPuntosSeleccion(misNaves);
78          mostrarNaves(misNaves);
79          System.out.println(x:"\nORDENAMIENTO POR NOMBRES (SELECCIÓN):");
80          ordenarPorNombreSeleccion(misNaves);
81          mostrarNaves(misNaves);
82          System.out.println(x:"\nORDENAMIENTO POR PUNTOS (INSERCIÓN):");
83          ordenarPorPuntosInsercion(misNaves);
84          mostrarNaves(misNaves);
85          System.out.println(x:"\nORDENAMIENTO POR NOMBRES (INSERCIÓN):");
86          ordenarPorNombreInsercion(misNaves);
87          mostrarNaves(misNaves);
88      }
89      public static void mostrarNaveEspecifica(Nave [] flota, int n){
90          System.out.println("\nNave " + (n + 1)+ ": " + flota[n].getNombre());
91          System.out.println("Ubicacion: " + flota[n].getColumna() + "-" + flota[n].getFila());
92          if (flota[n].getEstado() == true)
93              System.out.println(x:"Estado: Activo");
94          else
95              System.out.println(x:"Estado: Inactivo");
96          System.out.println("Puntos: " + flota[n].getPuntos());
97      }
```

```java
public static void mostrarNaves(Nave [] flota){
    for (int i = 0; i < flota.length; i++){
        System.out.println("\nNave " + (i + 1)+ ": " + flota[i].getNombre());
        System.out.println("Ubicacion: " + flota[i].getColumna() + "-" + flota[i].getFila());
        if (flota[i].getEstado() == true)
            System.out.println(x:"Estado: Activo");
        else
            System.out.println(x:"Estado: Inactivo");
        System.out.println("Puntos: " + flota[i].getPuntos());
    }
}
public static void mostrarPorNombre(Nave [] flota){
    Scanner sc = new Scanner(System.in);
    String nombre = sc.next();
    for (int i = 0; i < flota.length; i++)
        if (nombre.equals(flota[i].getNombre())){
            System.out.println("Nave " + (i + 1) + ": " + nombre);
            System.out.println("Ubicacion: " + flota[i].getColumna() + "-" + flota[i].getFila());
            if (flota[i].getEstado() == true)
                System.out.println(x:"Estado: Activo");
            else
                System.out.println(x:"Estado: Inactivo");
            System.out.println("Puntos: " + flota[i].getPuntos());
        }
}
```

```java
public static void mostrarPorPuntos(Nave [] flota){
    Scanner sc = new Scanner(System.in);
    int puntos = sc.nextInt();
    System.out.println("Lista de naves con puntos menores o iguales a " + puntos + ":");
    for (int i = 0; i < flota.length; i++)
        if (puntos >= flota[i].getPuntos()){
            System.out.println("Nave " + (i + 1) + ": " + flota[i].getNombre());
            System.out.println("Ubicacion: " + flota[i].getColumna() + "-" + flota[i].getFila());
            if (flota[i].getEstado() == true)
                System.out.println(x:"Estado: Activo");
            else
                System.out.println(x:"Estado: Inactivo");
            System.out.println("Puntos: " + flota[i].getPuntos());
        }
}
public static void mostrarMayorPuntos(Nave [] flota){
    int mayor = flota[0].getPuntos();
    int idx = 0;
    for (int i = 0; i < flota.length; i++)
        if (flota[i].getPuntos() > mayor){
            mayor = flota[i].getPuntos();
            idx = i;
        }
    System.out.println("Nave " + (idx + 1) + ": " + flota[idx].getNombre());
    System.out.println("Ubicacion: " + flota[idx].getColumna() + "-" + flota[idx].getFila());
    if (flota[idx].getEstado() == true)
        System.out.println(x:"Estado: Activo");
    else
        System.out.println(x:"Estado: Inactivo");
    System.out.println("Puntos: " + flota[idx].getPuntos());
}
```

```java
    public static int busquedaLinealNombre(Nave[] flota, String s){
        for (int i = 0; i < flota.length; i++)
            if (flota[i].getNombre().equalsIgnoreCase(s))
                return i;
        return -1;
    }
    public static void ordenarPorPuntosBurbuja(Nave[] flota){
        int n = flota.length;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (flota[j].getPuntos() > flota[j+1].getPuntos()) {
                    Nave temp = flota[j];
                    flota[j] = flota[j + 1];
                    flota[j + 1] = temp;
                }
    }
    public static void ordenarPorNombreBurbuja(Nave[] flota){
        int n = flota.length;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n - i - 1; j++)
                if (flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0) {
                    Nave temp = flota[j];
                    flota[j] = flota[j + 1];
                    flota[j + 1] = temp;
                }
    }
```

```java
    public static int busquedaBinariaNombre(Nave[] flota, String s){
        int left = 0;
        int right = flota.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            int comparison = flota[mid].getNombre().compareTo(s);
            if (comparison == 0)
                return mid;
            if (comparison < 0)
                left = mid + 1;
            else
                right = mid - 1;
        }
        return -1;
    }
    public static void ordenarPorPuntosSeleccion(Nave[] flota){
        int n = flota.length;
        for (int i = 0; i < n - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < n; j++) {
                if (flota[j].getPuntos() < flota[minIdx].getPuntos())
                    minIdx = j;
            }
            Nave temp = flota[minIdx];
            flota[minIdx] = flota[i];
            flota[i] = temp;
        }
    }
```

```java
    public static void ordenarPorNombreSeleccion(Nave[] flota){
        int n = flota.length;
        for (int i = 0; i < n - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < n; j++)
                if (flota[j].getNombre().compareTo(flota[minIdx].getNombre()) < 0)
                    minIdx = j;
            Nave temp = flota[minIdx];
            flota[minIdx] = flota[i];
            flota[i] = temp;
        }
    }
    public static void ordenarPorPuntosInsercion(Nave[] flota){
        int n = flota.length;
        for (int i = 1; i < n; ++i) {
            Nave key = flota[i];
            int j = i - 1;
            while (j >= 0 && flota[j].getPuntos() > key.getPuntos()) {
                flota[j + 1] = flota[j];
                j = j - 1;
            }
            flota[j + 1] = key;
        }
    }
```

```java
    public static void ordenarPorNombreInsercion(Nave[] flota){
        int n = flota.length;
        for (int i = 1; i < n; ++i) {
            Nave key = flota[i];
            int j = i - 1;
            while (j >= 0 && flota[j].getNombre().compareTo(key.getNombre()) > 0) {
                flota[j + 1] = flota[j];
                j = j - 1;
            }
            flota[j + 1] = key;
        }
    }
}

```

Ejecución del programa

```
Nave 1
Nombre: Jhon
Fila (Entre 1 y 10): 2
Columna (De la A a la J): H
Estado: true
Puntos: 8
Nave 2
Nombre: Abel
Fila (Entre 1 y 10): 4
Columna (De la A a la J): B
Estado: true
Puntos: 5
Nave 3
Nombre: Zamio
Fila (Entre 1 y 10): 4
Columna (De la A a la J): D
Estado: false
Puntos: 3
Nave 4
Nombre: Hierro
Fila (Entre 1 y 10): 8
Columna (De la A a la J): G
Estado: false
Puntos: 10
Nave 5
Nombre: Lila
Fila (Entre 1 y 10): 5
Columna (De la A a la J): D
Estado: true
Puntos: 7
```

```
Generando naves...

Naves creadas:

Nave 1: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 2: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 3: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

Nave 4: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10

Nave 5: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Introduzca el nombre de la nave que desea buscar: Lila
Nave 5: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7
```

```
Introduzca el nombre de la nave que desea buscar: Lila
Nave 5: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Introduzca los puntos maximos que puede tener una nave: 5
Lista de naves con puntos menores o iguales a 5:
Nave 2: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5
Nave 3: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

En su flota, la nave con el mayor numero de puntos es:
Nave 4: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10
```

```
BUSQUEDA LINEAL:

Ingrese el nombre de la nave que quiere buscar: Abo
No se ha encontrado la nave

Ordenamiento de las naves por el algoritmo Burbuja:

Por puntos:

Nave 1: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

Nave 2: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 3: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 4: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 5: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10
```

```
Por nombres:

Nave 1: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 2: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10

Nave 3: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 4: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 5: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

Se han ordenado exitosamente las naves, puede buscar por la forma binaria:
```

```
BUSQUEDA BINARIA:

Ingrese el nombre de la nave que quiere buscar: Lila

Nave 4: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

ORDENAMIENTO POR PUNTOS (SELECCI?N):

Nave 1: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

Nave 2: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 3: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 4: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 5: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10
```

```
ORDENAMIENTO POR NOMBRES (SELECCI?N):

Nave 1: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 2: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10

Nave 3: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 4: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 5: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3
```

```
ORDENAMIENTO POR PUNTOS (INSERCI?N):

Nave 1: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3

Nave 2: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 3: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 4: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 5: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10
```

```
ORDENAMIENTO POR NOMBRES (INSERCI?N):

Nave 1: Abel
Ubicacion: B-4
Estado: Activo
Puntos: 5

Nave 2: Hierro
Ubicacion: G-8
Estado: Inactivo
Puntos: 10

Nave 3: Jhon
Ubicacion: H-2
Estado: Activo
Puntos: 8

Nave 4: Lila
Ubicacion: D-5
Estado: Activo
Puntos: 7

Nave 5: Zamio
Ubicacion: D-4
Estado: Inactivo
Puntos: 3
```