

Clase Nave:

```
1 package L4;
2 public class Nave {
3     private String nombre;
4     private int fila;
5     private String columna;
6     private boolean estado;
7     private int puntos;
8     // Metodos mutadores
9     public void setNombre( String n){
10    nombre = n;
11    }
12    public void setFila(int f){
13    fila = f;
14    }
15    public void setColumna(String c){
16    columna = c;
17    }
18    public void setEstado(boolean e){
19    estado = e;
20    }
21    public void setPuntos(int p){
22    puntos = p;
23    }
24    // Metodos accesorios
25    public String getNombre(){
26    return nombre;
27    }
28    public int getFila(){
29    return fila;
30    }
31    public String getColumna(){
32    return columna;
33    }
34    public boolean getEstado(){
35    return estado;
36    }
37    public int getPuntos(){
38    return puntos;
39    }
40    // Completar con otros métodos necesarios
41 }
42
```

Clase DemoBatalla - main():

```
1 package L4;
2 import java.util.*;
3 public class DemoBatalla {
4     public static void main(String [] args){
5         Nave [] misNaves = new Nave[2];
6         Scanner sc = new Scanner(System.in);
7         String nomb = "", col = "";
8         int fil, punt = 0;
9         boolean est;
10        for (int i = 0; i < misNaves.length; i++) {
11            System.out.println("\nNave " + (i + 1));
12            System.out.print("Nombre: ");
13            nomb = sc.next();
14            System.out.print("Fila: ");
15            fil = sc.nextInt();
16            System.out.print("Columna: ");
17            col = sc.next();
18            System.out.print("Estado: ");
19            est = sc.nextBoolean();
20            System.out.print("Puntos: ");
21            punt = sc.nextInt();
22            misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
23            misNaves[i].setNombre(nomb);
24            misNaves[i].setFila(fil);
25            misNaves[i].setColumna(col);
26            misNaves[i].setEstado(est);
27            misNaves[i].setPuntos(punt);
28        }
29        System.out.println("\nNaves creadas:");
30        mostrarNaves(misNaves);
31        System.out.println();
32        System.out.print("Ingrese el nombre de la nave(s) a mostrar: ");
33        String elNombre = sc.next();
34        sc.nextLine();
35        mostrarPorNombre(misNaves, elNombre);
36        System.out.print("\nIngrese el limite de puntos de la(s) nave(s) a mostrar: ");
37        int puntosPedidos = sc.nextInt();
38        sc.nextLine();
39
40        mostrarPorPuntos(misNaves, puntosPedidos);
41        System.out.println("\nLa nave con mayor número de puntos es: ");
42        mostrarDatosNave(mostrarMayorPuntos(misNaves));
43        System.out.println();
44        Nave[] arrAleatorio = devuelveArray(misNaves);
45        System.out.println("Aleatorio");
46        mostrarNaves(arrAleatorio);
47        int pos=busquedaLinealNombre(misNaves, nomb);
48        System.out.println("Ordenando por puntos BURBUJA");
49        ordenarPorPuntosBurbuja(misNaves);
50        mostrarNaves(misNaves);
51        System.out.println("\n\nDESORDENANDO LAS NAVES");
52        devuelveArray(misNaves);
53        mostrarNaves(misNaves);
54        System.out.println("Ordenando por nombre BURBUJA");
55        ordenarPorNombreBurbuja(misNaves);
56        mostrarNaves(misNaves);
57        //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
58    }
```

```
59
60 System.out.print("\n\nIngrese el nombre de la nave a buscar: ");
61 nomb = sc.nextLine();
62 pos=busquedaBinariaNombre(misNaves, nomb);
63 System.out.println("La nave con nombre "+ nomb + " esta en la posicion " +
64     pos + " del arreglo de naves\n");
65 System.out.println("\n\nDESORDENANDO LAS NAVES");
66 devuelveArray(misNaves);
67 mostrarNaves(misNaves);
68
69 System.out.println("Ordenando por puntos SELECCION");
70 ordenarPorPuntosSeleccion(misNaves);
71 mostrarNaves(misNaves);
72 System.out.println("\n\nDESORDENANDO LAS NAVES");
73 devuelveArray(misNaves);
74 mostrarNaves(misNaves);
75 System.out.println("Ordenando por nombres SELECCION");
76 ordenarPorNombreSeleccion(misNaves);
77 mostrarNaves(misNaves);
78
79 System.out.println("\n\nDESORDENANDO LAS NAVES");
80 devuelveArray(misNaves);
81 mostrarNaves(misNaves);
82 System.out.println("Ordenando por puntos INSERION");
83 ordenarPorPuntosInsercion(misNaves);
84 mostrarNaves(misNaves);
85 System.out.println("\n\nDESORDENANDO LAS NAVES");
86 devuelveArray(misNaves);
87 mostrarNaves(misNaves);
88 System.out.println("Ordenando por nombre INSERION");
89 ordenarPorNombreInsercion(misNaves);
90 mostrarNaves(misNaves);
91
92 }
93 //Método para mostrar todas las naves
94 public static void mostrarNaves(Nave[] flota){
95     for (Nave misNaves : flota){
96         System.out.println("\tNombre: " + misNaves.getNombre() + "\tFila: "
97             + misNaves.getFila() + "\tColumna: " + misNaves.getColumna() +
98             "\tEstado: " + misNaves.getEstado() + "\t Puntos: " + misNaves.getPuntos());
99     }
100
101 }
102
103 //Método para mostrar todas las naves de un nombre que se pide por teclado
104 public static void mostrarPorNombre(Nave[] flota, String nombre){
105     for (int i = 0; i < flota.length; i++) {
106         if(flota[i].getNombre().equalsIgnoreCase(nombre)){
107             System.out.println("\tNombre: " + flota[i].getNombre() + "\tFila: "
108                 + flota[i].getFila() + "\tColumna: " + flota[i].getColumna() +
109                 "\tEstado: " + flota[i].getEstado() + "\t Puntos: " + flota[i].getPuntos());
110
111         }
112     }
113 }
114
```

```
114
115 //Método para mostrar todas las naves con un número de puntos inferior o igual
116 //al número de puntos que se pide por teclado
117 public static void mostrarPorPuntos(Nave [] flota, int puntosPedidos){
118     for (int i = 0; i < flota.length; i++) {
119         if(flota[i].getPuntos() <= puntosPedidos)
120             System.out.println("\tNombre: " + flota[i].getNombre() + "\tFila: "
121                 + flota[i].getFila() + "\tColumna: " + flota[i].getColumna() +
122                 "\tEstado: " + flota[i].getEstado() + "\t Puntos: " + flota[i].getPuntos());
123     }
124 }
125
126 //Método que devuelve la Nave con mayor número de Puntos
127 public static Nave mostrarMayorPuntos(Nave [] flota){
128     int indexMayor = 0;
129     for (int i = 0; i < flota.length; i++) {
130         if(flota[i].getPuntos() > flota[indexMayor].getPuntos())
131             indexMayor = i;
132     }
133     return flota[indexMayor];
134 }
135
136 //Crear un método que devuelva un nuevo arreglo de objetos con todos los
137 //objetos previamente ingresados
138 //pero aleatoriamente desordenados
139 public static Nave[] devuelveArray(Nave[] flota) {
140     Random rand = new Random();
141     int r1, r2;
142     Nave auxiliar;
143     Nave[] arrAleatorio = new Nave[flota.length];
144     for (int i = 0; i < flota.length; i++) {
145         arrAleatorio[i] = flota[i];
146     }
147     for(int i = 0; i < arrAleatorio.length; i++) {
148         r1 = rand.nextInt(arrAleatorio.length);
149         r2 = rand.nextInt(arrAleatorio.length);
150         auxiliar = arrAleatorio[r1];
151         arrAleatorio[r1] = arrAleatorio[r2];
152         arrAleatorio[r2] = auxiliar;
153     }
154     return arrAleatorio;
155 }
156
157 public static void mostrarDatosNave(Nave naveParticular) {
158     System.out.println("\tNombre: " + naveParticular.getNombre() + "\tFila: "
159         + naveParticular.getFila() + "\tColumna: " + naveParticular.getColumna() +
160         "\tEstado: " + naveParticular.getEstado() + "\t Puntos: "
161         + naveParticular.getPuntos());
162 }
163
164 //Método para buscar la primera nave con un nombre que se pidió por teclado
165 public static int busquedaLinealNombre(Nave[] flota, String s){
166     for (int i = 0; i < flota.length; i++) {
167         if (flota[i].getNombre().equals(s)){
168             return i;
169         }
170     }
171     return -1;
172 }
```

```
173
174 //Método que ordena por número de puntos de menor a mayor
175 public static void ordenarPorPuntosBurbuja(Nave[] flota){
176     for (int i = 0; i < flota.length; i++) {
177         for (int j = 0; j < flota.length - 1 - i; j++) {
178             if(flota[j].getPuntos() > flota[j + 1].getPuntos()) {
179                 Nave temp = flota[j];
180                 flota[j] = flota[j + 1];
181                 flota[j + 1] = temp;
182             }
183         }
184     }
185 }
186
187
188 //Método que ordena por nombre de A a Z
189 public static void ordenarPorNombreBurbuja(Nave[] flota){
190     for (int i = 0; i < flota.length; i++) {
191         for (int j = 0; j < flota.length - 1 - i; j++) {
192             if(flota[j].getNombre().compareTo(flota[j + 1].getNombre()) > 0){
193                 Nave temp = flota[j];
194                 flota[j] = flota[j + 1];
195                 flota[j + 1] = temp;
196             }
197         }
198     }
199 }
200
201 //Método para buscar la primera nave con un nombre que se pidió por teclado
202 public static int busquedaBinariaNombre(Nave[] flota, String s){
203     int alta, baja, media;
204     baja = 0;
205     alta = flota.length - 1;
206     while (baja <= alta) {
207         media = (alta + baja) / 2;
208         if (flota[media].getNombre().equals(s))
209             return media;
210         else if (s.compareTo(flota[media].getNombre()) < 0)
211             alta = media - 1;
212         else
213             baja = media + 1;
214     }
215     return -1;
216 }
217
218 //Método que ordena por número de puntos de menor a mayor
219 public static void ordenarPorPuntosSeleccion(Nave[] flota) {
220     for (int i = 0; i < flota.length - 1; i++) {
221         int indexMenor = i;
222         for (int j = i + 1; j < flota.length; j++) {
223             if (flota[j].getPuntos() < flota[indexMenor].getPuntos()) {
224                 indexMenor = j;
225             }
226         }
227         if (indexMenor != i) {
228             Nave temp = flota[i];
229             flota[i] = flota[indexMenor];
230             flota[indexMenor] = temp;
231         }
232     }
233 }
```

```
234
235
236 //Método que ordena por nombre de A a Z
237 public static void ordenarPorNombreSeleccion(Nave[] flota){
238     for (int i = 0; i < flota.length - 1; i++) {
239         int indexMenor = i;
240         for (int j = i + 1; j < flota.length; j++) {
241             if (flota[j].getNombre().compareTo(flota[indexMenor].getNombre()) < 0) {
242                 indexMenor = j;
243             }
244         }
245         if (indexMenor != i) {
246             Nave temp = flota[i];
247             flota[i] = flota[indexMenor];
248             flota[indexMenor] = temp;
249         }
250     }
251 }
252
253 //Método que muestra las naves ordenadas por número de puntos de mayor a menor
254 public static void ordenarPorPuntosInsercion(Nave[] flota) {
255     Nave aux;
256     int cont1, cont2;
257     for (cont1 = 1; cont1 < flota.length; cont1++) {
258         aux = flota[cont1];
259         for (cont2 = cont1 - 1; cont2 >= 0 && flota[cont2].getPuntos() > aux.getPuntos(); cont2--) {
260             flota[cont2 + 1] = flota[cont2];
261             flota[cont2] = aux;
262         }
263     }
264 }
265
266 //Método que muestra las naves ordenadas por nombre de Z a A
267 public static void ordenarPorNombreInsercion(Nave[] flota){
268     Nave aux;
269     int cont1, cont2;
270     for (cont1 = 1; cont1 < flota.length; cont1++) {
271         aux = flota[cont1];
272         for (cont2 = cont1 - 1; cont2 >= 0 && flota[cont2].getNombre().compareTo(aux.getNombre()) > 0; cont2--) {
273             flota[cont2 + 1] = flota[cont2];
274             flota[cont2] = aux;
275         }
276     }
277 }
278 }
```

II. PRUEBAS

- *Primero ingrese los datos de cada nave, reducí el tamaño a 5 con fines prácticos, pero debería funcionar para las n naves.*
- *Como los métodos tratan de ordenar los arreglos de objetos de acuerdo a un atributo específico, para demostrar que los métodos funcionan, desordene el array después de cada ordenamiento.*
- *Me asegure que para los métodos de búsqueda como el binario el array primero este ordenado según el parametro que busca la búsqueda binaria (en este caso por nombres).*
- *Lo que esperaba obtener era mi arreglo ordenado, el desordenado, y la posición del nombre que estaba buscando.*

Ingreso datos naves:

```
Nave 1
Nombre: BRAYAN
Fila: 1
Columna: 1
Estado: FALSE
Puntos: 1
```

```
Nave 2
Nombre: CARLOS
Fila: 2
Columna: 2
Estado: FALSE
Puntos: 2
```

```
Nave 3
Nombre: MARCOS
Fila: 3
Columna: 3
Estado: TRUE
Puntos: 3
```

```
Nave 4
Nombre: DANIEL
Fila: 4
Columna: 4
Estado: TRUE
Puntos: 4
```

```
Nave 5
Nombre: FIORELA
Fila: 5
Columna: 5
Estado: FALSE
```


Los siguientes métodos comienzan a trabajar:

Naves creadas:

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

Ingrese el nombre de la nave(s) a mostrar: DANIEL

Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
----------------	---------	------------	--------------	-----------

Ingrese el limite de puntos de la(s) nave(s) a mostrar: 3

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3

La nave con mayor número de puntos es:

Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
-----------------	---------	------------	---------------	-----------

Aleatorio

Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1

Ordenando por puntos BURBUJA

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

DESORDENANDO LAS NAVES

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

Ordenando por nombre BURBUJA

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3

Ingrese el nombre de la nave a buscar:

Ingrese el nombre de la nave a buscar: DANIEL

La nave con nombre DANIEL esta en la posicion 2 del arreglo de naves

DESORDENANDO LAS NAVES

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3

Ordenando por puntos SELECCION

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

DESORDENANDO LAS NAVES

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

Ordenando por nombres SELECCION

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3

DESORDENANDO LAS NAVES

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3

Ordenando por puntos INSERION

Nombre: BRAYAN	Fila: 1	Columna: 1	Estado: false	Puntos: 1
Nombre: CARLOS	Fila: 2	Columna: 2	Estado: false	Puntos: 2
Nombre: MARCOS	Fila: 3	Columna: 3	Estado: true	Puntos: 3
Nombre: DANIEL	Fila: 4	Columna: 4	Estado: true	Puntos: 4
Nombre: FIORELA	Fila: 5	Columna: 5	Estado: false	Puntos: 5

```
DESORDENANDO LAS NAVES
Nombre: BRAYAN Fila: 1 Columna: 1 Estado: false Puntos: 1
Nombre: CARLOS Fila: 2 Columna: 2 Estado: false Puntos: 2
Nombre: MARCOS Fila: 3 Columna: 3 Estado: true Puntos: 3
Nombre: DANIEL Fila: 4 Columna: 4 Estado: true Puntos: 4
Nombre: FIORELA Fila: 5 Columna: 5 Estado: false Puntos: 5
Ordenando por nombre INSERION
Nombre: BRAYAN Fila: 1 Columna: 1 Estado: false Puntos: 1
Nombre: CARLOS Fila: 2 Columna: 2 Estado: false Puntos: 2
Nombre: DANIEL Fila: 4 Columna: 4 Estado: true Puntos: 4
Nombre: FIORELA Fila: 5 Columna: 5 Estado: false Puntos: 5
Nombre: MARCOS Fila: 3 Columna: 3 Estado: true Puntos: 3
PS C:\Users\Hogar\Documents\BRAYAN\FP2 - Laboratories>
```

Upload to github:

Laboratorio-4 had recent pushes 26 minutes ago

Compare & pull request

Laboratorio-4

4 Branches 0 Tags

Go to file

+

<> Code

This branch is 11 commits ahead of main

Contribute

BCarlosAC Update Nave.java

c2e5663 · 26 minutes ago 15 Commits

DemoBatalla.java

Update DemoBatalla.java

27 minutes ago

Nave.java

Update Nave.java

26 minutes ago

History for AUCCACUSI_BRAYAN_LABORATORIOS / DemoBatalla.java on Laboratorio-4

Commits on Oct 13, 2024

Update DemoBatalla.java

BCarlosAC authored 27 minutes ago

Verified

Commits on Oct 11, 2024

Add files via upload

BCarlosAC authored 2 days ago

Verified

End of commit history for this file

Laboratorio-4

Go to file

- DemoBatalla.java
- Nave.java

BCarlosAC Update DemoBatalla.java

Code Blame 278 lines (255 loc) · 11.5 KB

```

1  package L4;
2  import java.util.*;
3  public class DemoBatalla {
4  public static void main(String [] args){
5      Nave [] misNaves = new Nave[2];
6      Scanner sc = new Scanner(System.in);
7      String nomb = "", col = "";
8      int fil, punt = 0;
9      boolean est;
10     for (int i = 0; i < misNaves.length; i++) {
11         System.out.println("\nNave " + (i + 1));
12         System.out.print("Nombre: ");

```

Commits

History for AUCCACUSI_BRAYAN_LABORATORIOS / Nave.java on Laboratorio-4

- Commits on Oct 13, 2024

Update Nave.java

BCarlosAC authored 28 minutes ago

Verify
- Commits on Oct 11, 2024

Add files via upload

BCarlosAC authored 2 days ago

Verify

Delete Nave.java

BCarlosAC authored 2 days ago

Verify

Add files via upload

BCarlosAC authored 2 days ago

Verify

Link Repositorio_rama: Laboratorio4

https://github.com/BCarlosAC/AUCCACUSI_BRAYAN_LABORATORIOS.git

III. CUESTIONARIO:

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

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	7	18	

CONCLUSIONES

Hay varios algoritmos de ordenamiento de arreglos, los objetos en el arreglo pueden tener diferentes atributos, sin embargo, a la hora de ordenarlos la estructura del método es el mismo solo que se usan diferentes getters para comparar los elementos del método; hay métodos especiales para comparar tipos de atributos como por ejemplo el "compareTo" que directamente me compara String de acuerdo al orden alfabético considerando que si se repiten letras pasa a la siguiente.

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>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: 14</p>

METODOLOGÍA DE TRABAJO

1. *Este es la continuación de un trabajo anterior por lo que solo se pasó a completar los métodos correspondientes.*
2. *Se resolvió los métodos según la guía y razonando, pero si tomaba mucho tiempo se recurre a información externa.*
3. *El código y mediante el análisis de los errores se refinó el código hasta llegar a la versión final.*

REFERENCIAS Y BIBLIOGRAFÍA

<https://www.youtube.com/watch?v=O4iuk9VhqYs&t=547s>