Informe de Laboratorio 04

Tema: Arreglos de Objetos, Bsqueda y Ordenamiento de Burbuja

Nota

Estudiante	Escuela	Asignatura
Hernan Andy Choquehuanca Zapana hchoquehuancaz@unsa.edu.pe	Escuela Profesional de Ingeniera de Sistemas	Fundamentos de la Programacin II Semestre: II Código: 20232191

Laboratorio	Tema	Duración
04	Arreglos de Objetos, Bsqueda y Ordenamiento de Burbuja	02 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 20 Septiembre 2023	Al 25 Septiembre 2023

1. Tarea

- Cree un Proyecto llamado Laboratorio4
- Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio 3
- Completar el Código de la clase DemoBatalla
- Utilizar Git para evidenciar su trabajo.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- VIM 9.0.
- Visual Studio Code.
- Git 2.41.1.
- Cuenta en GitHub con el correo institucional.
- Variables Simples
- Arreglos de Objetos.
- Métodos.
- Métodos de Búsqueda y Ordenamiento.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/hernanchoquehuanca/fp2-23b.git`
- URL para el laboratorio 04 en el Repositorio GitHub.
- `https://github.com/hernanchoquehuanca/fp2-23b/tree/main/fase01/lab04`

4. Actividades con el repositorio GitHub

4.1. Commits

4.1.1. Actividad 1 :

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 :

- Primero acomodamos y copiamos el código del lab03, para luego empezar a completarlo
- El código fue el siguiente:

Listing 1: DemoBatalla.java

```java
// Laboratorio Nro 02 - Ejercicio01
// Autor : Hernan Andy
// Colaboro : -
// Tiempo : -

import java.util.*;
public class DemoBatalla {
  public static void main(String [] args){
    Nave [] misNaves = new Nave[8];
    Scanner sc = new Scanner(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("Nombre: ");
      nomb = sc.next();
      System.out.println("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();
```

```
28
29      misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
30      misNaves[i].setNombre(nomb);
31      misNaves[i].setFila(fil);
32      misNaves[i].setColumna(col);
33      misNaves[i].setEstado(est);
34      misNaves[i].setPuntos(punt);
35    }
36    System.out.println("\nNaves creadas:");
37    mostrarNaves(misNaves);
38    mostrarPorNombre(misNaves);
39    mostrarPorPuntos(misNaves);
40    System.out.println("\nNave con mayor nmero de puntos: " + mostrarMayorPuntos(misNaves));
41
42    //leer un nombre
43    //mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en caso
          contrario
44    int pos=busquedaLinealNombre(misNaves,nombre);
45    ordenarPorPuntosBurbuja(misNaves);
46    mostrarNaves(misNaves);
47    ordenarPorNombreBurbuja(misNaves);
48    mostrarNaves(misNaves);
49
50    //mostrar los datos de la nave con dicho nombre, mensaje de no encontrado en caso
          contrario
51    pos=busquedaBinariaNombre(misNaves,nombre);
52    ordenarPorPuntosSeleccion(misNaves);
53    mostrarNaves(misNaves);
54    ordenarPorPuntosInsercion(misNaves);
55    mostrarNaves(misNaves);
56    ordenarPorNombreSeleccion(misNaves);
57    mostrarNaves(misNaves);
58    ordenarPorNombreInsercion(misNaves);
59    mostrarNaves(misNaves);
60  }
61  // Mtodo  para mostrar todas las naves
62  public static void mostrarNaves(Nave [] flota){
63    System.out.println("Naves ingresadas: ");
64    for (int i = 0; i < flota.length; i++){
65      mostrarNave(flota[i]);
66    }
67  }
68
69  // Mtodo  para mostrar todas las naves de un nombre que se pide por teclado
70  public static void mostrarPorNombre(Nave [] flota){
71    Scanner sc = new Scanner(System.in);
72    System.out.println("Ingrese un nombre: ");
73    String name = sc.next();
74
75    for (int i = 0; i < flota.length; i++){
76      if (flota[i].getNombre().equals(name))
77        mostrarNave(flota[i]);
78    }
79  }
80
81  // Mtodo  para mostrar todas las naves con un nmero de puntos inferior o igual
```

```
82  //al  nmero  de puntos que se pide por teclado
83  public static void mostrarPorPuntos(Nave [] flota){
84    Scanner sc = new Scanner(System.in);
85    System.out.println("Ingrese una cantidad de puntos: ");
86    int puntos = sc.nextInt();
87
88    for (int i = 0; i < flota.length; i++){
89      if (flota[i].getPuntos() <= puntos){
90        mostrarNave(flota[i]);
91      }
92    }
93  }
94  public static void mostrarNave(Nave nave){
95    System.out.println("\n Nave: " + nave.getNombre());
96    if (nave.getEstado())
97      System.out.println("Estado : Alive");
98      else
99      System.out.println("Estado : Dead");
100   System.out.println("Puntos : " + nave.getPuntos());
101
102 }
103
104 // Mtodo  que devuelve la Nave con mayor nmero de Puntos
105 public static Nave mostrarMayorPuntos(Nave [] flota){
106   int min = flota[0].getPuntos();
107   int positionNave = 0;
108   for (int i = 1; i < flota.length; i++){
109     if (flota[i].getPuntos() > min){
110       min = flota[i].getPuntos();
111       positionNave = i;
112     }
113   }
114   return flota[positionNave];
115 }
116
117 //Crear un mtodo que devuelva un nuevo arreglo de objetos con todos los objetos
        previamente ingresados
118 //pero aleatoriamente desordenados
119
120 public static Nave [] desordenarFlota(Nave flota []){
121   Nave [] flotaRandom = new Nave [flota.length];
122   for (int i = 0; i < flota.length; i++) {
123     int r = (int)Math.random() * 10;
124     while (flotaRandom[r] != null){
125       r = (int)Math.random() * 10;
126     }
127     flotaRandom [r] = flota[i];
128   }
129
130   return flotaRandom;
131 }
132
133 // Mtodo  para buscar la primera nave con un nombre que se pidi por teclado
134 public static int busquedaLinealNombre(Nave[] flota, String s){
135 }
136 // Mtodo  que ordena por nmero de puntos de menor a mayor
```

```
  public static void ordenarPorPuntosBurbuja(Nave[] flota){
  }
  // Mtodo  que ordena por nombre de A a Z
  public static void ordenarPorNombreBurbuja(Nave[] flota){
  }
  // Mtodo  para buscar la primera nave con un nombre que se pidi por teclado
  public static int busquedaBinariaNombre(Nave[] flota, String s){
  }
  // Mtodo  que ordena por nmero de puntos de menor a mayor
  public static void ordenarPorPuntosSeleccion(Nave[] flota){
  }
  // Mtodo  que ordena por nombre de A a Z
  public static void ordenarPorNombreSeleccion(Nave[] flota){
  }
  // Mtodo  que muestra las naves ordenadas por nmero de puntos de mayor a menor
  public static void ordenarPorPuntosInsercion(Nave[] flota){
  }
  // Mtodo  que muestra las naves ordenadas por nombre de Z a A
  public static void ordenarPorNombreInsercion(Nave[] flota){
  }
}
```

Listing 2: Commit: Reutilizando el código del lab03 para la actividad

```
$ git add .
$ git commit -m "Reutilizando el cdigo del lab03 para la actividad"
$ git push -u origin main
```

- Luego Implementamos el método busquedaLinealNombre haciendo uso de un simple bucle for, dentro una condificonal if que imprimirá aquellas naves con el mismo nombre ingresados.
- El código fue el siguiente:

Listing 3: DemoBatalla.java

```
// Mtodo  para buscar la primera nave con un nombre que se pidi por teclado
public static int busquedaLinealNombre(Nave[] flota, String s){
  for (int i = 0; i < flota.length; i++){
    if (flota[i].getNombre().equals(s))
      return i;
  }
  System.out.println("No se encontr");
  return -1;
}
```

Listing 4: Commit: Implementando el método busquedaLinealNombre y completando el main para recibir el nombre

```
$ git add .
$ git commit -m "Implementando el mtodo busquedaLinealNombre y completando el main para
    recibir el nombre"
$ git push -u origin main
```

- Adicionalmente a los métodos planteados, se creó un método (crearArreglo) para realizar copias del arreglo enviado como argumento y que no sea modificado.
- El código fue el siguiente:

Listing 5: DemoBatalla.java

```
public static Nave[] crearArreglo(Nave[] flota){
    Nave[] flotaNueva = new Nave[flota.length];
    System.arraycopy(flota, 0, flotaNueva, 0, flota.length);
    return flotaNueva;
  }
}
```

Listing 6: Commit: Implementando un nuevo método para copiar los arreglos antes de usarlos en los futuros métodos

```
$ git add .
$ git commit -m "Implementando un nuevo mtodo para copiar los arreglos antes de usarlos
    en los futuros mtodos"
$ git push -u origin main
```

- Se creó otro método llamado intercambiar, el cual recibe como parámetros un arreglo, y dos enteros los cuales nos sirven para intercambiar posiciones dentro de un arreglo.
- El código fue el siguiente:

Listing 7: DemoBatalla.java

```
public static void intercambiar(Nave[] flota, int i, int j){
  Nave temp = flota[i];
  flota[i] = flota[j];
  flota[j] = temp;
}
```

Listing 8: Commit: Implementando el método intercambiar para usarlo en el ordenarPorNombreBurbuja

```
$ git add .
$ git commit -m "Implementando el mtodo intercambiar para usarlo en el
    ordenarPorNombreBurbuja"
$ git push -u origin main
```

- Utilizando el algoritmo de ordenación por Burbuja, se implementó el método con bucles anidados y una condicional de ordenamiento.
- El código fue el siguiente:

Listing 9: DemoBatalla.java

```
public static void ordenarPorPuntosBurbuja(Nave[] flota){
  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())
        intercambiar(flota, j, j+1);
}
```

Listing 10: Commit: Implementando el método ordenarPorPuntosBurbuja usando el método intercambiar y el algoritmo de ordenamiento por burbuja

```
$ git add .
$ git commit -m "Implementando el mtodo ordenarPorPuntosBurbuja usando el mtodo
    intercambiar y el algoritmo de ordenamiento por burbuja"
$ git push -u origin main
```

- De la misma manera que el método anterior, se realizó el odenamiento por Burbuja, pero utilizando el primer caracter de cada nombre para ordenarlos.
- El código fue el siguiente:

Listing 11: DemoBatalla.java

```
public static void ordenarPorNombreBurbuja(Nave[] flota){
  for (int i = 1; i < flota.length; i++)
    for (int j = 0; j < flota.length - i; j++)
      if ((int)flota[j].getNombre().charAt(0) > (int)flota[j+1].getNombre().charAt(0))
        intercambiar(flota, j, j+1);
}
```

Listing 12: Commit: Implementando el método ordenarPorNombreBurbuja usando un cast a la primera letra de los nombre para asi ordenarlos alfabéticamente

```
$ git add .
$ git commit -m "Implementando el mtodo ordenarPorNombreBurbuja usando un cast a la
    primera letra de los nombre para asi ordenarlos alfabticamente"
$ git push -u origin main
```

- Implementando el método busquedaBinariaNombre utilizando el algoritmo proporcionado.
- Se creó un nuevo método para comparar nombres por sus caracteres.
- Así de esta manera la condicional de este algoritmo (else if) será más práctica al ejecutarse.
- El código fue el siguiente:

Listing 13: DemoBatalla.java

```
public static int busquedaBinariaNombre(Nave[] flota, String s){
  ordenarPorNombreBurbuja(flota);
  int baja = 0; int media; int alta = flota.length;

  while (baja <= alta){
    media = (alta + baja) / 2;
    if (flota[media].getNombre().equals(s))
      return media;
```

```
    else if (compararNombresMayor(s, flota[media].getNombre()))
      alta = media - 1;
      else
        baja = media + 1;
  }
  return -1;
}
```

Listing 14: Commit: Implementando el método busquedaBinariaNombre usando un nuevo método creado para compara valores de dos strings (nombres)

```
$ git add .
$ git commit -m "Implementando el mtodo busquedaBinariaNombre usando un nuevo mtodo
    creado para compara valores de dos strings (nombres)"
$ git push -u origin main
```

- Usando como guía el algoritmo dado sobre Selección, se utilizó aquello para convertirlo a código.
- Recorriendo el arreglo de Naves, conforme se selecciona aquel que tiene la menor cantidad de puntos, para luego relizar su intercambio de posición.
- El código fue el siguiente:

Listing 15: DemoBatalla.java

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

Listing 16: Commit: Implementando el método ordenarPorPuntosSeleccion usando el algoritmo proporcionado y el método intercambiar

```
$ git add .
$ git commit -m "Implementando el mtodo ordenarPorPuntosSeleccion usando el algoritmo
    proporcionado y el mtodo intercambiar"
$ git push -u origin main
```

- Siguiendo la misma lógica del método anterior, pero esta vez para ordenarPorNombreSeleccion se utilizó el método ya mencionado (compararNombresMayor) para tomar en cuenta los valores de las letras que conforman el nombre.
- El código fue el siguiente:

Listing 17: DemoBatalla.java

```
public static void ordenarPorNombreSeleccion(Nave[] flota){
  for (int i = 0; i < flota.length - 1; i++) {
    int nombreMin = i;
    for (int j = i + 1; j < flota.length; j++) {
        if (compararNombresMayor(flota[j].getNombre(), flota[i].getNombre())) {
            nombreMin = j;
        }
    }
    intercambiar(flota, nombreMin, i);
  }
}
```

Listing 18: Commit: Implementando el método ordenarPorNombreSeleccion usando el algoritmo dado, además del método compararNombresMayor creado anteriormente

```
$ git add .
$ git commit -m "Implementando el mtodo ordenarPorNombreSeleccion usando el algoritmo
    dado, adems del mtodo compararNombresMayor creado anteriormente"
$ git push -u origin main
```

- Haciendo uso del algoritmo por inserción brindado, se convirtió a código haciendo uso de dos bucles (while dentro de for), además que el while nos permite encontrar aquel Nave con mayor puntuación que la varible puntos, para luego realizar su intercambio.
- Posteriormente se imprimen el nuevo arreglo de Naves ordenadas por puntos.
- El código fue el siguiente:

Listing 19: DemoBatalla.java

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

    while (j >= 0 && flota[j].getPuntos() > puntos) {
        flota[j + 1] = flota[j];
        j--;
    }
    flota[j + 1].setPuntos(puntos);
  }

  System.out.println("Flota ordenada por puntos (insercion): ");
  for (int i = 0; i < flota.length; i++){
    System.out.println(flota[i].getNombre() + ": " + flota[i].getPuntos());
  }
}
```

Listing 20: Commit: Implementando el método ordenarPorPuntosInsercion haciendo uso del algoritmo proporcionado y adaptándolo al código

```
$ git add .
```

```
$ git commit -m "Implementando el mtodo ordenarPorPuntosInsercion haciendo uso del
    algoritmo proporcionado y adaptndolo al cdigo"
$ git push -u origin main
```

- Finalmente siguiendo la lógica anterior, esta vez con los nombres se hizo uso del método compararNombresMayor para reordenar las Naves según el orden jerárquico de sus nombres.
- El código fue el siguiente:

Listing 21: DemoBatalla.java

```
1  public static void ordenarPorNombreInsercion(Nave[] flota) {
2    for (int i = 1; i < flota.length; i++) {
3      Nave naveActual = flota[i];
4      int j = i - 1;
5
6      while (j >= 0 && compararNombresMayor(flota[j].getNombre(), naveActual.getNombre())) {
7        flota[j + 1] = flota[j];
8        j--;
9      }
10
11     flota[j + 1] = naveActual;
12   }
13
14   System.out.println("Flota ordenada por nombres (insercin): ");
15   for (int i = 0; i < flota.length; i++) {
16       System.out.println((i + 1) + ". " + flota[i].getNombre());
17   }
18 }
```

Listing 22: Commit: Implementando el método ordenarPorNombreInsercion utilizando elmétodo compararNombresMayor que fue creado anteriormente, además del algoritmo proporcionado

```
$ git add .
$ git commit -m "Implementando el mtodo ordenarPorNombreInsercion utilizando elmtodo
    compararNombresMayor que fue creado anteriormente, adems del algoritmo
    proporcionado"
$ git push -u origin main
```

4.2. Estructura de laboratorio 03

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab04
|   DemoBatalla.java
|   Nave.java
|
+------latex
    |   Informe_Lab04.pdf
    |   Informe_Lab04.tex
    |
    |------img
    |       logo_abet.png
    |       logo_episunsa.png
    |       logo_unsa.jpg
    |
    +------src
            BinariaNombre.java
            BusquedaLineal.java
            CodigoDado.java
            CopiarArreglos.java
            Intercambiar.java
            NombreBurbuja.java
            NombresInsercion.java
            PuntosBurbuja.java
            PuntosInsercion.java
            PuntosSeleccion.java
            Selection.java
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio 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 lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: 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

Tabla 3: Rúbrica para contenido del Informe y demostración

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	3	
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			
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		15	

6. Referencias

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html
- https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html