Informe de Laboratorio 04

Tema: Arreglos de Objetos, Búsqueda y Ordenamiento de Burbuja

Nota

Estudiante	Escuela	Asignatura
Julio Rubén Chura Acabana jchuraaca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	F. de Programción 2 Semestre: I Código: 20230472

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

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

1. Tarea

- En esta práctica se le dará un código incompleto donde deberá completar los métodos. Se le solicita utilizar los distintos tipos de búsqueda como la lineal y la binaria. También deberá hacer uso de los algoritmos de ordenamiento (Selección, inserción y burbuja)
- Usted debe realizar varios commits y al término de la actividad deberá elaborar un informe

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows
- VIM 9.0.
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.
- Arreglos Estándar

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JulioChura/fp2-23b.git`
- URL para el laboratorio 01 en el Repositorio GitHub.
- `https://github.com/JulioChura/fp2-23b/tree/main/fase01/lab04`

4. Actividades con el repositorio GitHub

Listing 1: Preparando los archivos, para ello se copiará el código del lab03 al lab04

```
mkdir lab04
cd lab01
Copy-Item -Path "Nave.java" -Destination "..\lab04"
Copy-Item -Path "DemoBatalla" -Destination "..\lab04"
cd ..
cd lab04
```

Listing 2: Se copian los métodos que faltan al código DemoBatalla.java, ahora solo hay que completar los métodos

```
vim DemoBatalla.java
```

Listing 3: Commit: Codigo de la practica de lab04 sin llenar los nuevos metodos

```
git add DemoBatalla.java
git commit -m "Codigo de la practica de lab04 sin llenar los nuevos metodos"
git push -u origin main
```

Listing 4: Implementado el método busquedaLinealNombre

```
vim DemoBatalla.java
```

- El método se encarga de retornar el índice donde se encuentre la palabra solicitada por el usuario, en caso de no encontrar se retorna un -1.

```
122     //Método para buscar la primera nave con un nombre que se pidió por teclado
123     public static int busquedaLinealNombre(Nave[] flota, String s){
124         for(int i = 0; i < flota.length; i++){
125             if(s.equalsIgnoreCase(flota[i].getNombre())){
126                 return i;
127             }
128         }
129         return -1;
130     }
```

- Para ver el funcionamiento del método, se implementa en el main lo necesario. Lo más interesante son las condicionales, en caso el número sea -1, se imprime un mensaje de que no fue encontrada la nave, caso contrario se imprime todos los datos de esta.

```
41          System.out.println("Ingrese algun nombre de una nave");
42          nombre = sc.next();
43          int pos=busquedaLinealNombre(misNaves,nombre);
44          if(pos==-1)
45              System.out.println("Nave no encontrada");
46          else
47              System.out.println("Nave encontrada"+ misNaves[pos].toString());
```

- Para poder probar el método, se cambio el tamaño del arreglo siendo ahora de 2, además que algunas líneas fueron puestas como comentarios para evitar que al momento de imprimir, sea algo extenso

Listing 5: Probando el metodo busquedaLinealNombre

```
javac DemoBatalla.java
java DemoBatalla.java
Nave 1
Nombre: alfa
Fila
3
Columna: 5
Estado: true
Puntos: 43
Nave 2
Nombre: beta
Fila
43
Columna: 5
Estado: true
Puntos: 43
Ingrese algun nombre de una nave
beta
Nave encontradaNombre: beta, Fila: 43, Columna: 5, Estado; true, Puntos: 43
```

Listing 6: Commit: Metodo busquedaLinealNombre completo

```
git add DemoBatalla.java
git commit -m "Metodo busquedaLinealNombre completo"
git push -u origin main
```

Listing 7: Implementadno el método de ordenarPorPuntosBurbuja

```
vim DemoBatalla.java
```

- En este método se pretende ordenar los puntos que tiene cada nave usando el ordenamiento burbuja

```java
    //Método que ordena por número de puntos de menor a mayor
    public static void ordenarPorPuntosBurbuja(Nave[] flota){
        for(int i = 0; i < flota.length-1; i++){
            for(int j = 0; j < flota.length-1; j++){
                if(flota[j].getPuntos() > flota[j+1].getPuntos()){
                    Nave auxiliar = flota[j];
                    flota[j] = flota[j+1];
                    flota[j+1] = auxiliar;
                }
            }
        }
    }
```

Listing 8: Probando el metodo ordenarPorPuntosBurbuja

```
javac DemoBatalla.java
java DemoBatalla.java
Nave 1
Nombre: alfa
Fila
34
Columna: 5
Estado: true
Puntos: 34
Nave 2
Nombre: beta
Fila
43
Columna: 5
Estado: true
Puntos: 6
Nave 3
Nombre: coraza
Fila
3
Columna: 6
Estado: true
Puntos: 55
Nave 1: Nombre: beta, Fila: 43, Columna: 5, Estado; true, Puntos: 6
Nave 2: Nombre: alfa, Fila: 34, Columna: 5, Estado; true, Puntos: 34
Nave 3: Nombre: coraza, Fila: 3, Columna: 6, Estado; true, Puntos: 55
```

Listing 9: Commit: Metodo ordenarPorPuntosBurbuja fue culminado

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorPuntosBurbuja"
git push -u origin main
```

Listing 10: Implementado el método ordenarPorNombreBurbuja

```
vim DemoBatalla.java
```

- El método se encarga de ordenar de forma alfabética los Strings de un arreglo. Se hace uso del método compareTo(String s) el cual retorna un número negativo si s es mayor que el otro string, retorna un valor positivo cuando s es menor que el otro string y cero cuando ambos strings son iguales. Luego se sigue la misma lógica del ordenamiento Burbuja

```
145     public static void ordenarPorNombreBurbuja(Nave[] flota){
146         for(int i = 0; i < flota.length-1;i++){
147             for(int j = 0; j < flota.length-1; j++){
148                 if(flota[j].getNombre().compareTo(flota[j+1].getNombre())>0){
149                     Nave aux = flota[j];
150                     flota[j] = flota[j+1];
151                     flota[j+1] = aux;
152                 }
153             }
154         }
155     }
```

- Para poder probar el método, se cambio el tamaño del arreglo siendo ahora de 3, además que algunas líneas fueron puestas como comentarios para evitar que al momento de imprimir, sea algo extenso

Listing 11: Probando el metodo ordenarPorNombreBurbuja

```
Nave 1
Nombre: caza
Fila
5
Columna: 6
Estado: true
Puntos: 34
Nave 2
Nombre: alfa
Fila
23
Columna: 5
Estado: true
Puntos: 67
Nave 3
Nombre: alcon
Fila
3
Columna: 6
Estado: true
Puntos: 87
Nave 1: Nombre: alcon, Fila: 3, Columna: 6, Estado; true, Puntos: 87
Nave 2: Nombre: alfa, Fila: 23, Columna: 5, Estado; true, Puntos: 67
Nave 3: Nombre: caza, Fila: 5, Columna: 6, Estado; true, Puntos: 34
```

Listing 12: Commit:"Método ordenarPorNombreBurbuja culminado"

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorNombreBurbuja culminado"
git push -u origin main
```

Listing 13: Implementado el método busquedaBinariaNombre

```
vim DemoBatalla.java
```

- El método se encarga de hacer la búsqueda de un string ingresado por el usuario. Se hace uso del algoritmo de búsqueda binaria. Se usa los métodos equals para determinar si el String en la posición media es igual al solicitado y también se hace uso del método compareTo para determinar las posiciones alta y baja para seguir con el flujo algoritmo. Si no se encuentran coincidencias, se retorna el valor de -1. Se añadieron líneas en el main para que el método reciba los parámetros y pueda hacer su búsqueda. Una vez el método retorne un valor de tipo entero, dicho valor será evaluado en una estructura condicional y de acuerdo a las condiciones se imprime un mensaje.

```
161     public static int busquedaBinariaNombre(Nave[] flota, String s){
162         int baja = 0;
163         int alta = flota.length-1;
164         while(baja <= alta ){
165             int media = (baja+alta)/2;
166                 if (flota[media].getNombre().equals(s)){
167                     return media;
168                 }else{
169                     if(flota[media].getNombre().compareTo(s)  < 0){
170                         alta = media-1;
171                     }else{
172                         baja = media+1;
173                     }
174                 }
175         }
176         return -1;
```

- Fragmento de código en el main

```
54          System.out.println("Ingrese la nave que desea buscar");
55          nombre = sc.next();
56          pos=busquedaBinariaNombre(misNaves,nombre);
57          if(pos==-1)
58              System.out.println("Nave no encontrada");
59          else
60              System.out.println("Nave encontrada"+ misNaves[pos].toString());
```

Listing 14: Probando el metodo busquedaBinariaNombre

```
javac DemoBatalla.java
java DemoBatalla
Nave 1
Nombre: alfa
Fila
5
Columna: 6
Estado: true
Puntos: 34
Nave 2
```

```
Nombre: beta
Fila
3
Columna: 4
Estado: true
Puntos: 8
Nave 3
Nombre: gama
Fila
3
Columna: 4
Estado: true
Puntos: 32
Ingrese la nave que desea buscar
beta
Nave encontradaNombre: beta, Fila: 3, Columna: 4, Estado; true, Puntos: 8
```

Listing 15: Commit:"Metodo busquedaBinariaNombre"

```
git add DemoBatalla.java
git commit -m "Metodo busquedaBinariaNombre"
git push -u origin main
```

Listing 16: Implementado el método ordenarPorPuntosSeleccion

```
vim DemoBatalla.java
```

- El método se encarga de hacer un ordenamiento usando el algoritmo por selección, el cual consiste en empezar a tomar el elemento de la posición 0 (se irá recorriendo hasta llegar a la penúltima posición) y lo va comparando con los elementos que se encuentran a la derecha, si hay un elemento menor que él, se intercambian de posición, de modo que lo de la izquierda ya queda ordenado. En código usamos dos bucles for, el primero recorre los elementos del arreglo Naves, en el ámbito de este bucle declaramos una variable de tipo entera, la cual almacenará la posición donde se encuentra el elemento mínimo, luego se pone otro bucle for junto con la estructura condicional que servirán para hacer las comparaciones, si el elemento de la posición i+1 es menor que el de la posición i, ocurre una actualización, siendo ahora que minIndex se le asigna i+1 y así sucesivamente según la condición. Culminado el segundo bucle for, se hacen los respectivos cambios

```
181     public static void ordenarPorPuntosSeleccion(Nave[] flota){
182         for(int i = 0; i < flota.length-1; i++){
183             int minIndex = i;
184             for(int j = i+1; j < flota.length; j++){
185                 if(flota[j].getPuntos() < flota[minIndex].getPuntos()){
186                     minIndex = j;
187                 }
188             }
189             Nave auxiliar = flota[minIndex];
190             flota[minIndex] = flota[i];
191             flota[i] = auxiliar;
192         }
193     }
```

Listing 17: Probando el metodo ordenarPorPuntosSeleccion

```
javac DemoBatalla.java
java DemoBatalla
Nave 1
Nombre: Hydra
Fila
3
Columna: 4
Estado: true
Puntos: 43
Nave 2
Nombre: Caza
Fila
4
Columna: 6
Estado: true
Puntos: 76
Nave 3
Nombre: Misilera
Fila
1
Columna: 6
Estado: true
Puntos: 53
Nave 1: Nombre: Hydra, Fila: 3, Columna: 4, Estado; true, Puntos: 43
Nave 2: Nombre: Misilera, Fila: 1, Columna: 6, Estado; true, Puntos: 53
Nave 3: Nombre: Caza, Fila: 4, Columna: 6, Estado; true, Puntos: 76
```

Listing 18: Commit: "Metodo ordenarPorPuntosSeleccion culminado"

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorPuntosSeleccion culminado"
git push -u origin main
```

Listing 19: Implementado el método ordenarPorNombreSeleccion

```
vim DemoBatalla.java
```

- El método se encarga de hacer un ordenamiento de Strings usando el algoritmo por selección,básicamente la explicación es la misma que se hizo en el método de ordenarPuntosSeleccion , solo que aquí lo única que varía es la condición del if. Como en esta línea se están comparando Strings, se usa el método compareTo, esto con el fin de ordenar de la 'a' hasta la 'z'

```
194     //Método que ordena por nombre de A a Z
195     public static void ordenarPorNombreSeleccion(Nave[] flota){
196         for(int i = 0; i < flota.length-1; i++){
197             int minIndex = i;
198             for(int j = i+1; j < flota.length;j++){
199                 if(flota[j].getNombre().compareTo( flota[minIndex].getNombre())<0){
200                         minIndex = j;
201                 }
202             }
203         Nave auxiliar = flota[minIndex];
204         flota[minIndex] = flota[i];
205         flota[i] = auxiliar;
206         }
207     }
```

Listing 20: Probando el metodo ordenarPorNombreSeleccion

```
javac DemoBatalla.java
java DemoBatalla
Nave 1
Nombre: alfa
Fila
3
Columna: 5
Estado: true
Puntos: 56
Nave 2
Nombre: xtreme
Fila
7
Columna: 8
Estado: true
Puntos: 54
Nave 3
Nombre: huascar
Fila
3
Columna: 5
Estado: true
Puntos: 87
Nave 1: Nombre: alfa, Fila: 3, Columna: 5, Estado; true, Puntos: 56
Nave 2: Nombre: huascar, Fila: 3, Columna: 5, Estado; true, Puntos: 87
Nave 3: Nombre: xtreme, Fila: 7, Columna: 8, Estado; true, Puntos: 54
```

Listing 21: Commit: "Metodo ordenarPorNombreSeleccion completado"

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorNombreSeleccion completado"
git push -u origin main
```

Listing 22: Implementado el método ordenarPorPuntosInsercion

```
vim DemoBatalla.java
```

- El método se encarga de hacer un ordenamiento de Puntos usando el algoritmo de inserción.

Básicamente se extrae un elemento de la posición inicial , y se le compara con los elementos de su izquierda, de manera que el lado izquierdo va quedando ordenado

```
209     public static void ordenarPorPuntosInsercion(Nave[] flota){
210         int posicion;
211         Nave auxiliar;
212         for(int i = 0; i < flota.length; i++){
213             posicion = i;
214             auxiliar = flota[i];
215             while(0 < posicion && auxiliar.getPuntos() < flota[posicion-1].getPuntos()){
216                 flota[posicion] = flota[posicion-1];
217                 posicion--;
218             }
219             flota[posicion] = auxiliar;
220         }
221     }
```

Listing 23: Probando el metodo ordenarPorPuntoSeleccion

```
javac DemoBatalla.java
java DemoBatalla
Nave 1
Nombre: epsilon
Fila
2
Columna: 4
Estado: true
Puntos: 459
Nave 2
Nombre: omega
Fila
4
Columna: 5
Estado: true
Puntos: 450
Nave 3
Nombre: red
Fila
3
Columna: 7
Estado: true
Puntos: 100
Nave 1: Nombre: red, Fila: 3, Columna: 7, Estado; true, Puntos: 100
Nave 2: Nombre: omega, Fila: 4, Columna: 5, Estado; true, Puntos: 450
Nave 3: Nombre: epsilon, Fila: 2, Columna: 4, Estado; true, Puntos: 459
```

Listing 24: Commit: "Metodo ordenarPorPuntosInserccion culminado"

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorPuntosInserccion culminado"
git push -u origin main
```

Listing 25: Implementado el método ordenarPorNombreInsercion y dándole detalles al código

```
vim DemoBatalla.java
```

- El método se encarga de hacer un ordenamiento de Nombres usando el algoritmo de inserción. Básicamente se extrae un elemento de la posición inicial , y se le compara con los elementos de su izquierda, de manera que el lado izquierdo va quedando ordenado. Se usa compareTo ya que se trata de strings.

```
223     public static void ordenarPorNombreInsercion(Nave[] flota){
224         int posicion;
225         Nave auxiliar;
226         for(int i = 0; i < flota.length; i++){
227             posicion = i;
228             auxiliar = flota[i];
229             while(posicion > 0 && auxiliar.getNombre().compareTo(flota[posicion-1].getNombre())<0){
230                 flota[posicion] = flota[posicion-1];
231                 posicion--;
232             }
233             flota[posicion] = auxiliar;
234         }
235     }
```

Listing 26: Probando el metodo ordenarPorNombreInserción

```
javac DemoBatalla.java
java DemoBatalla
Nave 1
Nombre:
coraza
Fila
4
Columna: 5
Estado: true
Puntos: 5
Nave 2
Nombre: venganza
Fila
4
Columna: 6
Estado: true
Puntos: 65
Nave 3
Nombre: alfa
Fila
4
Columna: 5
Estado: true
Puntos: 56
Nave 4
Nombre: destructor
Fila
4
Columna: 6
Estado: true
Puntos: 32
Nave 1: Nombre: alfa, Fila: 4, Columna: 5, Estado; true, Puntos: 56
Nave 2: Nombre: coraza, Fila: 4, Columna: 5, Estado; true, Puntos: 5
Nave 3: Nombre: destructor, Fila: 4, Columna: 6, Estado; true, Puntos: 32
Nave 4: Nombre: venganza, Fila: 4, Columna: 6, Estado; true, Puntos: 65
```

Listing 27: Commit: "Metodo ordenarPorNombreSelecion culminado"

```
git add DemoBatalla.java
git commit -m "Metodo ordenarPorNombreSelecion culminado"
git push -u origin main
```

- * Hubo un error al momento del commit, realmente se está subiendo el método ordenarPorNombreInsercion

Listing 28: Se borran los comenatrios y el numero de elementos del arreglo vuelve a ser de 10 para probarlo

```
vim DemoBatalla.java
```

- Ahora lo que se hace para un mejor seguimiento de la ejecución del código es aumentar líneas que imprimen un mensaje el cual indica lo que se está mostrando.

Listing 29: Probando DemoBatalla.java

```
javac DemoBatalla.java
java DemoBatalla
 Nave 1
 Nombre: coraza
 Fila
 3
 Columna: 4
 Estado: true
 Puntos: 54
 Nave 2
 Nombre: alcon
 Fila
 4
 Columna: 7
 Estado: true
 Puntos: 43
 Nave 3
 Nombre: venganza
 Fila
 2
 Columna: 4
 Estado: false
 Puntos: 32
 Nave 4
 Nombre: huascar
 Fila
 2
 Columna: 9
 Estado: true
 Puntos: 80
 Nave 5
 Nombre: peregrino
 Fila
 1
 Columna: 7
 Estado: true
```

```
Puntos: 49
Nave 6
Nombre: alfa
Fila
2
Columna: 5
Estado: true
Puntos: 15
Nave 7
Nombre: alcon
Fila
4
Columna: 7
Estado: true
Puntos: 53
Nave 8
Nombre: milenario
Fila
8
Columna: 4
Estado: false
Puntos: 30
Nave 9
Nombre: destructor
Fila
3
Columna: 6
Estado: true
Puntos: 47
Nave 10
Nombre: omega
Fila
2
Columna: 3
Estado: false
Puntos: 56

Naves creadas:
Nave 1: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 2: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 3: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Nave 4: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Nave 5: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 6: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 7: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 8: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 9: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 10: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Ingrese el nombre de la nave que desea buscar
huascar
Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Ingrese los puntos para hacer la busqueda de resultados menores o igual
33
Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
```

```
Nave con mayor numero de puntos: Nombre: huascar, Fila: 2, Columna: 9, Estado; true,
    Puntos: 80
Se usara busqueda lineal para encontrar el nombre solicitado
Ingrese algun nombre de una nave
alcon
Nave encontrada Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Se usara ordenamiento burbuja para ordenar los puntos
Nave 1: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 2: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 3: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Nave 4: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 7: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 8: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 9: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 10: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Se usara ordenamiento burbuja para ordenar los nombres
Nave 1: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 2: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 3: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 4: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Nave 7: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 8: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 9: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 10: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Se usara busqueda binaria para encontrar el nombre de la nave a buscar
Ingrese la nave que desea buscar
utopia
Nave no encontrada
Se empleara orden por seleccion en los puntos
Nave 1: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 2: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 3: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Nave 4: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 7: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 8: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 9: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 10: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Se empleara orden por insercion en los puntos
Nave 1: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 2: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 3: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Nave 4: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 7: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 8: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 9: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 10: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
```

```
Se empleara orden por seleccion en los nombres
Nave 1: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 2: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 3: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 4: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Nave 7: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 8: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 9: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 10: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
Se empleara orden por insercion en los nombres
Nave 1: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 43
Nave 2: Nombre: alcon, Fila: 4, Columna: 7, Estado; true, Puntos: 53
Nave 3: Nombre: alfa, Fila: 2, Columna: 5, Estado; true, Puntos: 15
Nave 4: Nombre: coraza, Fila: 3, Columna: 4, Estado; true, Puntos: 54
Nave 5: Nombre: destructor, Fila: 3, Columna: 6, Estado; true, Puntos: 47
Nave 6: Nombre: huascar, Fila: 2, Columna: 9, Estado; true, Puntos: 80
Nave 7: Nombre: milenario, Fila: 8, Columna: 4, Estado; false, Puntos: 30
Nave 8: Nombre: omega, Fila: 2, Columna: 3, Estado; false, Puntos: 56
Nave 9: Nombre: peregrino, Fila: 1, Columna: 7, Estado; true, Puntos: 49
Nave 10: Nombre: venganza, Fila: 2, Columna: 4, Estado; false, Puntos: 32
```

Listing 30: Commit: "Subiendo DemoBatalla.java del lab04 en su version terminada"

```
git add DemoBatalla.java
git commit -m "Subiendo DemoBatalla.java del lab04 en su version terminada"
git push -u origin main
```

4.1. Estructura de laboratorio 01

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

```
lab04
|   DemoBatalla.java
|   Nave.java
|
|----latex
|      programacion_lab04_rescobedoq_v1.0.pdf
|      programacion_lab04_rescobedoq_v1.0.tex
|
|      img
|           1.jpg
|           10.jpg
|           11.jpg
|           12.jpg
|           2.jpg
|           3.jpg
|           4.jpg
|           5.jpg
|           6.jpg
|           7.jpg
|           8.jpg
|           9.jpg
```

```
|          logo_abet.png
|          logo_episunsa.png
|          logo_unsa.jpg
|
     src
```

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	2	
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	1	
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 estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	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	X	4	
Total		20		16	

6. Referencias

- https://drive.google.com/file/d/1CoQAKeKW-QDYRmHLrBdbSopFB1Z_Qmk3/view?usp=drive_link