INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	*Fundamentos de la Programación 2*				
TÍTULO DE LA PRÁCTICA:	*Practica de Laboratorio 4: Arreglos de Objetos, Búsquedas y Ordenamientos*				
NÚMERO DE PRÁCTICA:	*4*	**AÑO LECTIVO:**	*2024*	**NRO. SEMESTRE:**	*Segundo*
FECHA DE PRESENTACIÓN	*13/10/2024*	**HORA DE PRESENTACIÓN**	*23:59*		
INTEGRANTE (s) *Santiago Alonso Quintanilla Chávez*				**NOTA (0-20)**	
DOCENTE(s): *Ing. Lino Jose Pinto Oppe*					

RESULTADOS Y PRUEBAS

I. EJERCICIOS RESUELTOS:

Cree un Proyecto llamado Laboratorio4

Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java creadas en el Laboratorio3

1. Completar el código de la clase DemoBatalla
   a. **Código clase "Nave":**

```
public class Nave {
        private String nombre;
        private int fila;
        private String columna;
        private boolean estado;
        private int puntos;
        public void setNombre(String n){
            nombre=n;
        }
        public void setFila(int f){
            fila=f;
        }
        public void setColumna(String c){
            columna=c;
        }
        public void setEstado(boolean e){
            estado=e;
        }
        public void setPuntos(int p){
            puntos=p;
        }
        public String getNombre(){
            return nombre;
        }
        public int getFila(){
            return fila;
        }
        public String getColumna(){
            return columna;
        }
        public boolean getEstado(){
            return estado;
        }
        public int getPuntos(){
            return puntos;
        }
        public String toString() {
         return "-Nombre: "+nombre+"\t-Posicion: ("+fila+","+columna+")\t-Estado: "+estado+"\t-Puntos: "+puntos;
        }
}
```

b. Código clase "DemoBatalla":

```
import java.util.*;
public class DemoBatalla {
    public static void main(String[] args){
        Nave [] misNaves=new Nave[5];
        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.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);
        mostrarPorNombre(misNaves);
        mostrarPorPuntos(misNaves);
        System.out.println("\nNave con mayor número de puntos: \n-->"+mostrarMayorPuntos(misNaves));

        System.out.println("Ingrese el nombre a buscar: ");
        String nombre=sc.next();

        int pos=busquedaLinealNombre(misNaves,nombre);
        if (pos==-1) {
         System.out.println("no encontrado");
        } else {
         System.out.println(misNaves[pos]);
        }
```

```
46
47          ordenarPorPuntosBurbuja(misNaves);
48          mostrarNaves(misNaves);
49          ordenarPorNombreBurbuja(misNaves);
50          mostrarNaves(misNaves);
51
52          ordenarPorPuntosSeleccion(misNaves);
53          mostrarNaves(misNaves);
54          ordenarPorNombreSeleccion(misNaves);
55          mostrarNaves(misNaves);
56          ordenarPorPuntosInsercion(misNaves);
57          mostrarNaves(misNaves);
58          ordenarPorNombreInsercion(misNaves);
59          mostrarNaves(misNaves);
60
61      }
62      public static void mostrarNaves(Nave[] flota){
63          System.out.println("Naves de la flota: ");
64          for (int i=0;i<flota.length;i++){
65              System.out.println("-->"+flota[i]);
66          }
67      }
68      public static void mostrarPorNombre(Nave[] flota){
69          Scanner scan=new Scanner(System.in);
70          System.out.print("Nombre de la nave: ");
71          String nomb=scan.next();
72          Boolean verif=true;
73          for (int i=0;i<flota.length;i++){
74              if (nomb.equals(flota[i].getNombre())){
75                  System.out.println("-->"+flota[i]);
76                  verif=false;
77              }
78          }
79          if (verif){
80              System.out.println("No se encontro nave");
81          }
82      }
```

```java
public static void mostrarPorPuntos(Nave[] flota){
    Scanner scan=new Scanner(System.in);
    System.out.print("Ingrese límite de puntos: ");
    int puntos=scan.nextInt();
    Boolean verif=true;
    for (int i=0;i<flota.length;i++){
        int comparacion=flota[i].getPuntos();
        if (comparacion<=puntos){
            System.out.println("-->"+flota[i]);
            verif=false;
        }
    }
    if (verif){
        System.out.println("No se encontro nave");
    }
}
public static Nave mostrarMayorPuntos(Nave[] flota){
    int mayor=0, id=0;
    for (int i=0;i<flota.length;i++){
        int comparacion=flota[i].getPuntos();
        if (comparacion>mayor){
            mayor=comparacion;
            id=i;
        }
    }
    Nave naveMayor=flota[id];
    return naveMayor;
}
public static Nave[] metodoAleatorio(Nave[] flota){
 Random rand=new Random();
 Nave [] Aleatorio=new Nave[flota.length];
 ArrayList<Integer> indices=new ArrayList<Integer>();
 for (int i=0;i<flota.length;i++) {
     int indice=rand.nextInt(flota.length);
     while (indices.contains(indice)) {
         indice=rand.nextInt(flota.length);
     }
     indices.add(indice);
     Aleatorio[i]=flota[indice];
 }
 return Aleatorio;
}
```

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

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

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

- **Explicación del código:**
    - Los cambios realizados a la clase “DemoBatalla” son en la implementación de los métodos de búsqueda y de ordenamiento expuestos en el marco teórico: dos métodos de búsqueda (**Linea**l: donde se comparan uno a uno los elementos del arreglo y **Binario**: se comparan por mitades del arreglo que se van reduciendo, pero requiere que este ordenado), y tres métodos de ordenamiento (**Burbuja**: se comparan objetos adyacentes en el arreglo y se intercambian según el criterio de orden, **Por Selección**: cada objeto del arreglo se compara con todos los objetos hasta encontrar el menor y se intercambia, sucesivamente hasta llegar al último objeto del arreglo, **Por Inserción**: se ordenan los objetos por secciones cada vez más grandes del arreglo)

c. Ejecución del código:

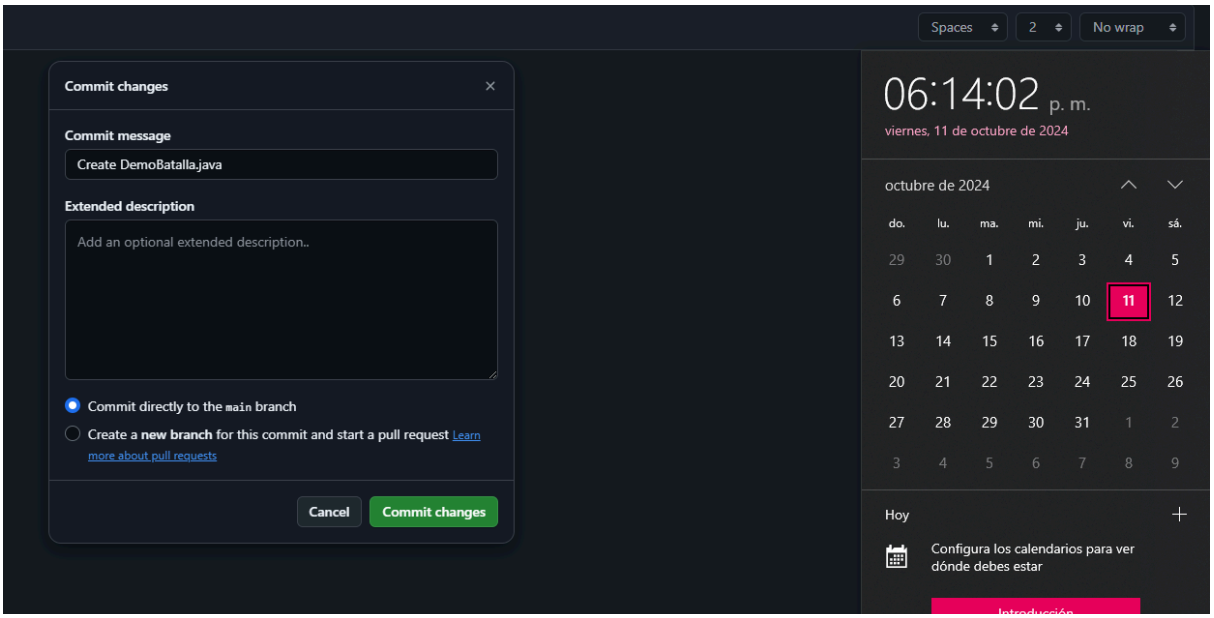
```
Nave 1
Nombre: lucas
Fila: 3
Columna: 7
Estado: true
Puntos: 31
Nave 2
Nombre: bruno
Fila: 9
Columna: 4
Estado: false
Puntos: 48
Nave 3
Nombre: mateo
Fila: 3
Columna: 9
Estado: false
Puntos: 58
Nave 4
Nombre: david
Fila: 2
Columna: 1
Estado: false
Puntos: 10
Nave 5
Nombre: jesus
Fila: 6
Columna: 6
Estado: true
Puntos: 90

Naves creadas:
Naves de la flota:
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
```

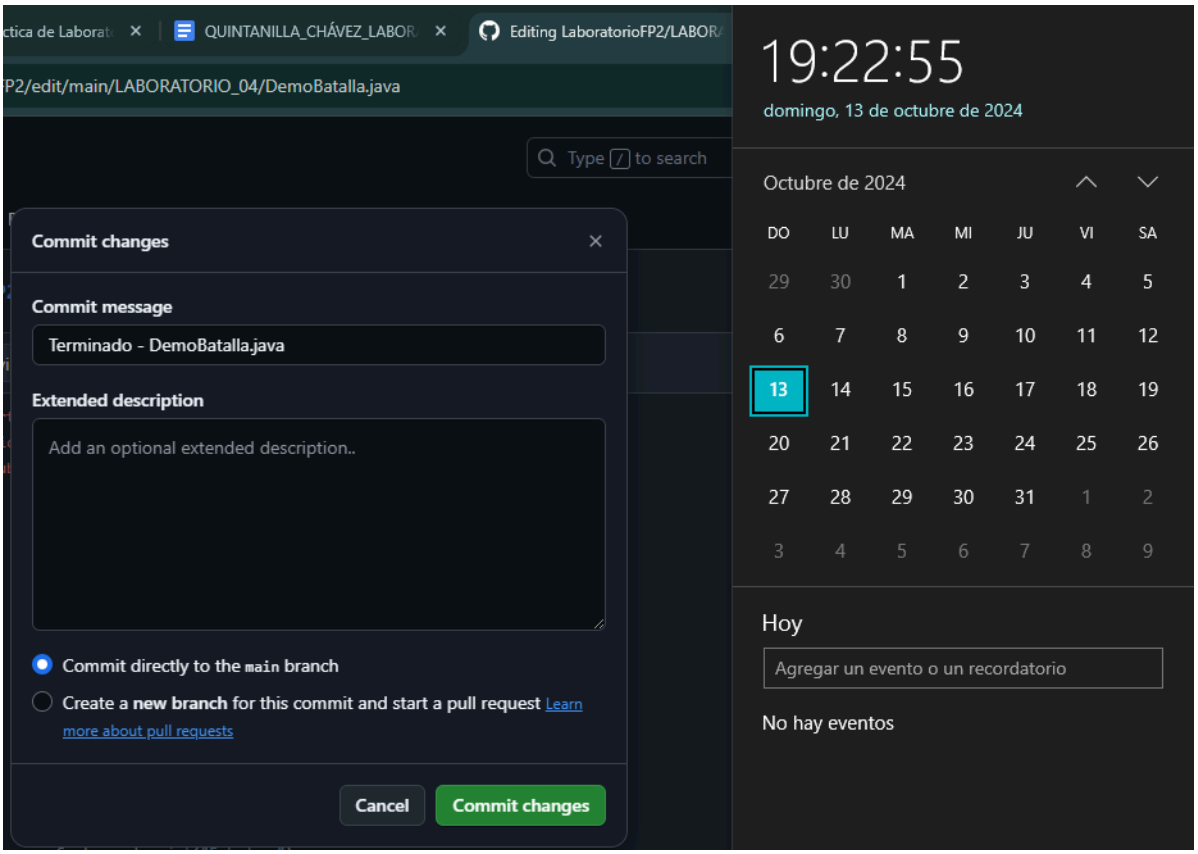
```
Nombre de la nave: bruno
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
Ingrese límite de puntos: 50
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10

Nave con mayor número de puntos:
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
Ingrese el nombre a buscar:
mateo
-Nombre: mateo  -Posicion: (3,9)        -Estado: false  -Puntos: 58
```

```
Naves de la flota:
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
Naves de la flota:
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
Naves de la flota:
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
Naves de la flota:
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
Naves de la flota:
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
Naves de la flota:
-->-Nombre: bruno       -Posicion: (9,4)        -Estado: false  -Puntos: 48
-->-Nombre: david       -Posicion: (2,1)        -Estado: false  -Puntos: 10
-->-Nombre: jesus       -Posicion: (6,6)        -Estado: true   -Puntos: 90
-->-Nombre: lucas       -Posicion: (3,7)        -Estado: true   -Puntos: 31
-->-Nombre: mateo       -Posicion: (3,9)        -Estado: false  -Puntos: 58
```

- **Explicación de la ejecución:** Las primeras dos capturas muestran la ejecución ya vista en el laboratorio 3, donde se insertan datos, se busca un objeto en específico, se filtran según un límite de puntos, y se muestra el objeto con mayor puntaje. En la tercera captura se muestran 6 ejecuciones del método mostrarNaves según el ordenamiento burbuja, ordenamiento por selección y ordenamiento por inserción (se van alternando entre ordenamiento por puntaje y por nombre, respectivamente)

- **Evidencia de los Commits del código DemoBatalla.java** (el código de Nave.java no sufrio cambios, por lo que se reutilizo el utilizado en el Laboratorio3)





	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. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4		X	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2		X	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2		X	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2		X	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2		X	
7. Ortografía	El documento no muestra errores ortográficos.	2		X	
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			
TOTAL		20			