Informe de Laboratorio 04

Tema: Arreglos de Objetos, Búsqueda y Ordenamiento de Burbuja

Nota

Estudiante	Escuela	Asignatura
Mikhail Gabino Velasque Arcos mvelasquea@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Laboratorio FUNDAMENTOS DE LA PROGRAMACION II Semestre: II Código: 20214260

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 Setiembre 2023	Al 27 Setiembre 2023

1. Actividades

- Cree un Proyecto llamado Laboratorio 4
- Usted podrá reutilizar las dos clases Nave.java y DemoBatalla.java. creadas en Laboratorio
- Completar el Código de la clase DemoBatallas

2. SOLUCIONARIO

- Se hace el uso de arreglo de objetos como el de nave.java y DemoBatalla.java para completar la actividad como se muestra en la siguiente seccion con la base ya planteada en el aterior laboratorio

2.1. CODIGO FUENTE

- Se crea la clase Nave.java

- Se crea la clase DemoBatalla$_v$2.*java* :

Listing 1: Creando la clase Nave

```
vim Nave.java
  vim DemoBatalla_v2.java
```

Listing 2: Creando la clase Nave

```
      public class Nave {
      /*
   laboratorio Nro 3 ejercicio 1
   Autor :Mikhail Gabino Velasque Arcos
  colaboro:---
  tiempo:
   */
      private String nombre;
      private int fila;
      private String columna;
      private boolean estado;
      private int puntos;
      // Metodos mutadores
      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;
      }
      // Metodos accesores
      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;
  }
      }
```

Listing 3: Creando la clase principal de DemoBatalla$_v$2.*java*

```
  public class DemoBatalla {

  public static void main(String [] args){
/*
   laboratorio Nro 3 ejercicio 1
   Autor :Mikhail Gabino Velasque Arcos
```

```
colaboro:---
tiempo:
 */
   Nave [] misNaves = new Nave[10];
   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();
   misNaves[i] = new Nave(); //Se crea un objeto Nave y se asigna su referencia a misNaves
   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 nmero de puntos: " + mostrarMayorPuntos(misNaves));
   }
   // Mtodo  para mostrar todas las naves
   public static void mostrarNaves(Nave [] flota){
      for (int i = 0; i < flota.length; i++) {
         System.out.println("NOMBRE: " +flota[i].getNombre());
         System.out.println("FILA: " +flota[i].getFila());
         System.out.println("COLUMNA: " +flota[i].getColumna());
         if (flota[i].getEstado()==true) {
         System.out.println("vivo");
         } else {
         System.out.println("muerto");
         }
         System.out.println("PUNTOS DE VIDA: "+flota[i].getPuntos());
         }}
   // Mtodo  para mostrar todas las naves de un nombre que se pide por teclado
   public static void mostrarPorNombre(Nave [] flota){
      String nombre;
      System.out.println("INGRESAR EL NOMBRE DEL QUE SE BUSCA");
      Scanner objeto= new Scanner(System.in);
      nombre=objeto.nextLine();
      for (int i = 0; i < flota.length; i++) {
      if (flota[i].getNombre().equals(nombre)) {
      System.out.println("COINCIDENCIA CON: "+nombre);
      System.out.println("NOMBRE: " +flota[i].getNombre());
```

```
      System.out.println("FILA: " +flota[i].getFila());
      System.out.println("COLUMNA: " +flota[i].getColumna());
      if (flota[i].getEstado()==true) {
      System.out.println("vivo");
      } else {
      System.out.println("esta muerto");
      }
      System.out.println("PUNTOS DE VIDA: "+flota[i].getPuntos());
      break;
      }else{
      System.out.println("---");
      }}}
   // Mtodo  para mostrar todas las naves con un nmero de puntos inferior o igual
   //al  nmero  de puntos que se pide por teclado
   public static void mostrarPorPuntos(Nave [] flota){
      for (int i = 0; i < flota.length; i++) {
         int puntoComparacion;
         System.out.println("INGRESE LA CANTIDAD DE VIDA QUE DESEA COMPARAR");
         Scanner objeto= new Scanner(System.in);
         puntoComparacion=objeto.nextInt();
         for (int j = 0; j < flota.length; j++) {
         if (flota[i].getPuntos()<=puntoComparacion) {
         System.out.println("NOMBRE: " +flota[i].getNombre());
         System.out.println("FILA: " +flota[i].getFila());
         System.out.println("COLUMNA: " +flota[i].getColumna());
         if (flota[i].getEstado()==true) {
         System.out.println("vivo");
         } else {
         System.out.println("muerto");
         }
         System.out.println("PUNTOS DE VIDA: "+flota[i].getPuntos());
         }else{
         System.out.println("---");
         }}}}
   // Mtodo  que devuelve la Nave con mayor nmero de Puntos

      public static Nave mostrarMayorPuntos(Nave[] flota) {
          Nave naveMayor = flota[0]; // Supongamos que la primera nave tiene la mayor
              cantidad de puntos inicialmente
          for (int i = 1; i < flota.length; i++) {
              if (flota[i].getPuntos() > naveMayor.getPuntos()) {
                  naveMayor = flota[i];
              }
          }
          return naveMayor;
      }
   }
```

2.2. EJERCICIO 02 (Uso de arreglos de objetos en el ejercicio 4 del lab 01)

- Se crea la clase Soldado.
- Se crea la clase principal:

Listing 4: Creando la clase Soldado

```
   public class Soldado {
      /*
 laboratorio Nro 3 ejercicio 2
 Autor :Mikhail Gabino Velasque Arcos
colaboro:---
tiempo:
 */
      private String nombre;
      private int nivelVida;

      public Soldado(String nombre, int nivelVida) {
          this.nombre = nombre;
          this.nivelVida = nivelVida;
      }

      public String getNombre() {
          return nombre;
      }

      public int getNivelVida() {
          return nivelVida;
      }

      public void setNivelVida(int nivelVida) {
          this.nivelVida = nivelVida;
      }

      public String toString() {
          return "Soldado:" + nombre + " con vidad de " + nivelVida;
      }

}
```

Listing 5: Creando la clase PRINCIPAL$_soldados$

```
public class PRINCIPAL_soldados {


   /*
    clase principal del ejercicio 05 del lab 1 aplicando arreglo de objetos
    */
    /*
 laboratorio Nro 3 ejercicio 2
 Autor :Mikhail Gabino Velasque Arcos
colaboro:---
tiempo:
 */
   public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        Soldado[] soldados = new Soldado[5];

        for (int i = 0; i < soldados.length; i++) {
```

```
            System.out.println("Soldado " + (i + 1));
            System.out.print("Nombre: ");
            String nombre = entrada.next();
            System.out.print("Nivel de Vida: ");
            int nivelVida = entrada.nextInt();

            soldados[i] = new Soldado(nombre, nivelVida);
        }

        System.out.println("\nLOS SOLDADOS CREADOS SON");
        for (Soldado soldado : soldados) {
            System.out.println(soldado);
        }
    }


}
```

2.3. EJERCICIO 03 (Uso de arreglos de objetos en el ejercicio 5 del lab 01)

- Se usa la clase Soldado que ya s etenia anteriormente.

- Se crea la clase principal EJERCITO$_V S$:

Listing 6: Creando la clase soldado

```
  public class Soldado {
   private String nombre;

   public Soldado(String nombre) {
       this.nombre = nombre;
   }

   public String getNombre() {
       return nombre;
   }
}
```

Listing 7: Creando la clase Ejercito$_v s$

```
  import java.util.*;
public class Ejercito_vs {

  public static void main(String[] args) {
     Soldado[] ejercito1 = inicioPROGRA();
        Soldado[] ejercito2 = inicioPROGRA();

        System.out.println("Ejrcito 1:");
        mostrar_Ejercito(ejercito1);

        System.out.println("\nEjrcito 2:");
        mostrar_Ejercito(ejercito2);

        int cantidad_ejercito1 = ejercito1.length;
```

```
            int cantidad_ejercito2 = ejercito2.length;

            if (cantidad_ejercito1 > cantidad_ejercito2) {
                System.out.println("\nEl Ejrcito 1 gana");
            } else if (cantidad_ejercito2 > cantidad_ejercito1) {
                System.out.println("\nEl Ejrcito 2 gana");
            } else {
                System.out.println("\nEmpate");
            }
        }

        public static Soldado[] inicioPROGRA() {
            Random r = new Random();
            int cantidadSoldados = r.nextInt(5) + 1;
            Soldado[] ejercito = new Soldado[cantidadSoldados];

            for (int i = 0; i < cantidadSoldados; i++) {
                ejercito[i] = new Soldado("Soldado" + i);
            }

            return ejercito;
        }
        public static void mostrar_Ejercito(Soldado[] ejercito) {
            for (int i = 0; i < ejercito.length; i++) {
                System.out.println("Soldado " + (i + 1) );
            }
        }

    }import java.util.*;

public class DemoBatalla_v2 {

    public static void main(String[] args) {
        Nave[] misNaves = new Nave[3];
        Scanner entrada = 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 = entrada.next();
            System.out.println("Fila: ");
            fil = entrada.nextInt();
            System.out.print("Columna: ");
            col = entrada.next();
            System.out.print("Estado: ");
            est = entrada.nextBoolean();
            System.out.print("Puntos: ");
            punt = entrada.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);

        // Encontrar la nave con mayor nmero de puntos
        Nave naveMayorPuntos = mostrarMayorPuntos(misNaves);
        System.out.println("\nNave con mayor nmero de puntos:");
        mostrarNave(naveMayorPuntos);

        // Bsqueda y ordenamientos
        System.out.print("\nIngrese el nombre de la nave a buscar: ");
        String nombreBusqueda = entrada.next();
        int pos = busquedaLinealNombre(misNaves, nombreBusqueda);
        if (pos != -1) {
            System.out.println("Nave encontrada en la posicin " + pos + ":");
            mostrarNave(misNaves[pos]);
        } else {
            System.out.println("Nave no encontrada.");
        }

        ordenarPorPuntosBurbuja(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Burbuja):");
        mostrarNaves(misNaves);

        ordenarPorNombreBurbuja(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Burbuja):");
        mostrarNaves(misNaves);

        pos = busquedaBinariaNombre(misNaves, nombreBusqueda);
        if (pos != -1) {
            System.out.println("\nNave encontrada en la posicin " + pos + ":");
            mostrarNave(misNaves[pos]);
        } else {
            System.out.println("Nave no encontrada.");
        }

      ordenarPorPuntosSeleccion(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Seleccin):");
        mostrarNaves(misNaves);

        ordenarPorPuntosInsercion(misNaves);
        System.out.println("\nNaves ordenadas por puntos (Insercin):");
        mostrarNaves(misNaves);

        ordenarPorNombreSeleccion(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Seleccin):");
        mostrarNaves(misNaves);

        ordenarPorNombreInsercion(misNaves);
        System.out.println("\nNaves ordenadas por nombre (Insercin):");
        mostrarNaves(misNaves);
    }
```

```
public static void mostrarNaves(Nave[] flota) {
    for (int i = 0; i < flota.length; i++) {
        mostrarNave(flota[i]);
    }
}

public static void mostrarNave(Nave nave) {
    System.out.println("NOMBRE: " + nave.getNombre());
    System.out.println("FILA: " + nave.getFila());
    System.out.println("COLUMNA: " + nave.getColumna());
    if (nave.getEstado()) {
        System.out.println("Estado: Vivo");
    } else {
        System.out.println("Estado: Muerto");
    }
    System.out.println("PUNTOS DE VIDA: " + nave.getPuntos());
}

public static void mostrarPorNombre(Nave[] flota) {
    String nombre;
    System.out.println("\nIngrese el nombre de la nave a buscar:");
    Scanner objeto = new Scanner(System.in);
    nombre = objeto.nextLine();
    int pos = busquedaLinealNombre(flota, nombre);
    if (pos != -1) {
        System.out.println("Nave encontrada en la posicin " + pos + ":");
        mostrarNave(flota[pos]);
    } else {
        System.out.println("Nave no encontrada.");
    }
}

public static void mostrarPorPuntos(Nave[] flota) {
    System.out.println("\nIngrese la cantidad de vida para comparar:");
    Scanner objeto = new Scanner(System.in);
    int puntoComparacion = objeto.nextInt();
    for (int i = 0; i < flota.length; i++) {
        if (flota[i].getPuntos() <= puntoComparacion) {
            System.out.println("Nave con menos de " + puntoComparacion + " puntos de
                vida:");
            mostrarNave(flota[i]);
        }
    }
}

public static Nave mostrarMayorPuntos(Nave[] flota) {
    Nave naveMayor = flota[0];
    for (int i = 1; i < flota.length; i++) {
        if (flota[i].getPuntos() > naveMayor.getPuntos()) {
            naveMayor = flota[i];
        }
    }
    return naveMayor;
}
```

```
public static int busquedaLinealNombre(Nave[] flota, String nombre) {
    for (int i = 0; i < flota.length; i++) {
        if (flota[i].getNombre().equalsIgnoreCase(nombre)) {
            return i; // Retorna la posicin de la nave encontrada
        }
    }
    return -1; // Nave no encontrada
}

public static void ordenarPorPuntosBurbuja(Nave[] flota) {
    boolean intercambiado;
    do {
        intercambiado = false;
        for (int i = 0; i < flota.length - 1; i++) {
            if (flota[i].getPuntos() > flota[i + 1].getPuntos()) {
                // Intercambiar las naves si estn fuera de orden
                Nave temp = flota[i];
                flota[i] = flota[i + 1];
                flota[i + 1] = temp;
                intercambiado = true;
            }
        }
    } while (intercambiado);
}

public static void ordenarPorNombreBurbuja(Nave[] flota) {
    boolean intercambiado;
    do {
        intercambiado = false;
        for (int i = 0; i < flota.length - 1; i++) {
            if (flota[i].getNombre().compareTo(flota[i + 1].getNombre()) > 0) {
                // Intercambiar las naves si estn fuera de orden
                Nave temp = flota[i];
                flota[i] = flota[i + 1];
                flota[i + 1] = temp;
                intercambiado = true;
            }
        }
    } while (intercambiado);
}

public static int busquedaBinariaNombre(Nave[] flota, String nombre) {
    Arrays.sort(flota, Comparator.comparing(Nave::getNombre)); // Asegurar que el arreglo
        est ordenado por nombre
    int inicio = 0;
    int fin = flota.length - 1;
    while (inicio <= fin) {
        int medio = inicio + (fin - inicio) / 2;
        int comparacion = nombre.compareTo(flota[medio].getNombre());
        if (comparacion == 0) {
            return medio; // Se encontr la nave
        } else if (comparacion < 0) {
            fin = medio - 1;
        } else {
            inicio = medio + 1;
        }
```

```
    }
    return -1; // Nave no encontrada
}

public static void ordenarPorPuntosSeleccion(Nave[] flota) {
    for (int i = 0; i < flota.length - 1; i++) {
        int indiceMinimo = i;
        for (int j = i + 1; j < flota.length; j++) {
            if (flota[j].getPuntos() < flota[indiceMinimo].getPuntos()) {
                indiceMinimo = j;
            }}
         Nave temp = flota[i];
        flota[i] = flota[indiceMinimo];
        flota[indiceMinimo] = temp;
    }}
public static void ordenarPorPuntosInsercion(Nave[] flota) {
    for (int i = 1; i < flota.length; i++) {
        Nave naveActual = flota[i];
        int j = i - 1;
        while (j >= 0 && flota[j].getPuntos() > naveActual.getPuntos()) {
            flota[j + 1] = flota[j];
            j--;
        }
        flota[j + 1] = naveActual;
    } }

public static void ordenarPorNombreSeleccion(Nave[] flota) {
    for (int i = 0; i < flota.length - 1; i++) {
        int indiceMinimo = i;
        for (int j = i + 1; j < flota.length; j++) {
            if (flota[j].getNombre().compareTo(flota[indiceMinimo].getNombre()) < 0) {
                indiceMinimo = j;
            }
        }
        // Intercambiar la nave actual con la nave de menor nombre
        Nave temp = flota[i];
        flota[i] = flota[indiceMinimo];
        flota[indiceMinimo] = temp;
    }}
public static void ordenarPorNombreInsercion(Nave[] flota) {
    for (int i = 1; i < flota.length; i++) {
        Nave naveActual = flota[i];
        int j = i - 1;
        while (j >= 0 && naveActual.getNombre().compareTo(flota[j].getNombre()) < 0) {
            flota[j + 1] = flota[j];
            j--;
        }
        flota[j + 1] = naveActual;
    }}}
```