

**Universidad Nacional de San Agustín**  
**Escuela Profesional de Ingeniería de Sistemas**  
**Fundamentos de Programación II**  
**Practica de Laboratorio 4:**  
**Arreglos de Objetos, Búsquedas y Ordenamientos**

**I**

---

**OBJETIVOS**

- Crear e inicializar arreglos de objetos
- Realizar Búsquedas secuencial y binaria en un arreglo de objetos
- Implementar método de ordenamiento de burbuja en arreglos de objetos.
- Arreglos de Objetos: Búsquedas y Ordenamiento

**II**

---

**MARCO TEORICO**

**Arreglos de Objetos**

Persona
<code>String nombre</code> <code>int edad</code> <code>char genero</code>
<code>void setNombre(String elNombre)</code> <code>void setEdad(int laEdad)</code> <code>void setGenero(char elGenero)</code> <code>String getNombre()</code> <code>int getEdad()</code> <code>char getGenero()</code>

Figura 1: Clase Persona

```
Persona[ ] misAmigos;  
misAmigos = new Persona[20];  
misAmigos[0] = new Persona( );  
misAmigos[0].setName("Juancito");
```

Figura 2: Ejemplo de creación de un arreglo de objetos

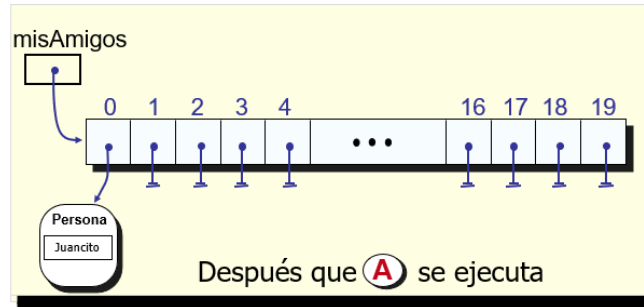


Figura 3: Representación gráfica del arreglo de objetos.

## Métodos de Búsqueda y Ordenamiento

Una de las operaciones más realizadas son la búsqueda de información y el ordenamiento de la misma. Existen muchos algoritmos, unos más eficientes que otros dependiendo del problema y la situación de los datos.

A continuación, se presenta el código de algunos de ellos.

```
public class Fundamentos2 {
    public static void main(String[] args){
        int num, indice;
        int[] estudiantesIds = {5, 18, 22, 17, 38};

        Scanner scan = new Scanner(System.in);
        System.out.println("Ingrese el id a buscar:");
        num = scan.nextInt();

        indice = encontrarEstudiante(estudiantesIds, num);

        if(indice != -1)
            System.out.println("encontrado en posicion:" + indice);
        else
            System.out.println("no se encontro");
    }
    public static int encontrarEstudiante(int[] ids, int id){
        for(int i = 0; i < ids.length; i++){
            if(ids[i] == id)
                return i;
        }
        return -1;
    }
}
```

Figura 4: Búsqueda Lineal

```

public static int buscarBinaria(int[] ids, int id){
    int alta, baja, media;
    baja=0;
    alta=ids.length-1;
    while(baja<=alta){
        media=(alta+baja)/2;
        if(ids[media]==id)
            return media;
        else if(id<ids[media])
            alta=media-1;
        else
            baja=media+1;
    }
    return -1;
}

```

Figura 5: Búsqueda Binaria

```

public static void ordenarBurbuja(int[] list){
    for (int i=1; i<list.length; i++){
        for(int j=0; j<list.length-i; j++){
            if(list[j]>list[j+1])
                intercambiar(list, j, j+1);
        }
    }
    private static void intercambiar(int[] list, int i, int j){
        int temp;
        temp = list[i];
        list[i] = list[j];
        list[j] = temp;
    }
}

```

Figura 6: Ordenamiento por burbuja

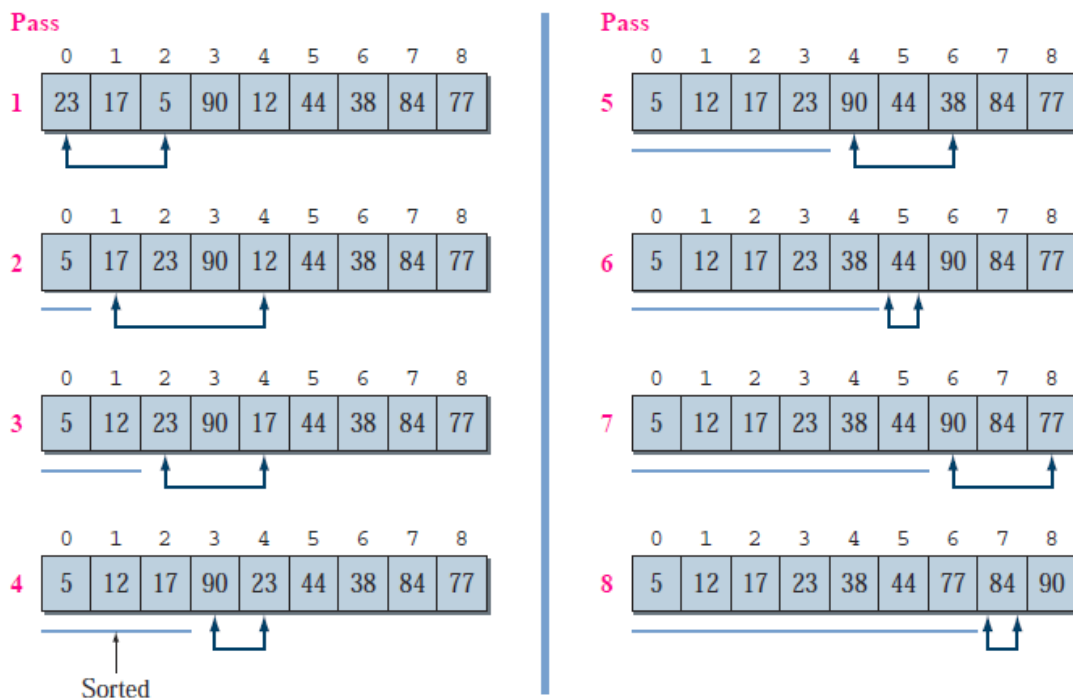


Figura 7: Ordenamiento por Selección

Region ordenada

	0	1	2	3	4
V	10	5	7	1	6

V	5	10	7	1	6
---	---	----	---	---	---

Region ordenada

Region ordenada

	0	1	2	3	4
V	5	10	7	1	6

V	5	7	10	1	6
---	---	---	----	---	---

Region Ordenada

Region Ordenada

	0	1	2	3	4
V	5	7	10	1	6

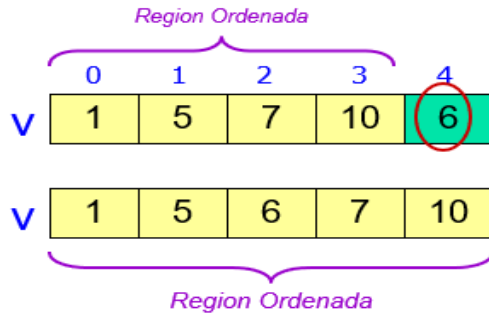
V	1	5	7	10	6
---	---	---	---	----	---

Region Ordenada

¿En dónde insertar  $v[1]$  en la región ordenada de su izquierda?

¿En dónde insertar  $v[2]$  en la región ordenada de su izquierda?

¿En dónde insertar  $v[3]$  en la región ordenada de su izquierda?



¿En dónde insertar  $v[4]$  en la región ordenada de su izquierda?

## Ordenamiento Completo.

Figura 8: Ordenamiento por Inserción

### III ACTIVIDADES

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

```
public class Nave {

    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 accesorios
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;
}
// Completar con otros métodos necesarios
}

```

```

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();

            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 número de puntos: " + mostrarMayorPuntos(misNaves));

        //leer un nombre
        //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
        int pos=busquedaLinealNombre(misNaves,nombre);

        ordenarPorPuntosBurbuja(misNaves);
        mostrarNaves(misNaves);
        ordenarPorNombreBurbuja(misNaves);
        mostrarNaves(misNaves);
    }
}

```

```

        //mostrar los datos de la nave con dicho nombre, mensaje de "no encontrado" en caso contrario
        pos=busquedaBinariaNombre(misNaves,nombre);

        ordenarPorPuntosSeleccion(misNaves);
        mostrarNaves(misNaves);
        ordenarPorNombreSeleccion(misNaves);
        mostrarNaves(misNaves);
        ordenarPorPuntosInsercion(misNaves);
        mostrarNaves(misNaves);
        ordenarPorNombreInsercion(misNaves);
        mostrarNaves(misNaves);
    }

    //Método para mostrar todas las naves
    public static void mostrarNaves(Nave[] flota){
        //REUTILIZAR
    }

    //Método para mostrar todas las naves de un nombre que se pide por teclado
    public static void mostrarPorNombre(Nave[] flota){
        //REUTILIZAR
    }

    //Método para mostrar todas las naves con un número de puntos inferior o igual
    //al número de puntos que se pide por teclado
    public static void mostrarPorPuntos(Nave[] flota){
        //REUTILIZAR
    }

    //Método que devuelve la Nave con mayor número de Puntos
    public static Nave mostrarMayorPuntos(Nave[] flota){
        //REUTILIZAR
    }

    //Método para buscar la primera nave con un nombre que se pidió por teclado
    public static int busquedaLinealNombre(Nave[] flota, String s){
    }

    //Método que ordena por número de puntos de menor a mayor
    public static void ordenarPorPuntosBurbuja(Nave[] flota){
    }

    //Método que ordena por nombre de A a Z
    public static void ordenarPorNombreBurbuja(Nave[] flota){
    }

    //Método para buscar la primera nave con un nombre que se pidió por teclado
    public static int busquedaBinariaNombre(Nave[] flota, String s){
    }

    //Método que ordena por número de puntos de menor a mayor
    public static void ordenarPorPuntosSeleccion(Nave[] flota){
    }

    //Método que ordena por nombre de A a Z
    public static void ordenarPorNombreSeleccion(Nave[] flota){
    }

    //Método que muestra las naves ordenadas por número de puntos de mayor a menor
    public static void ordenarPorPuntosInsercion(Nave[] flota){
    }

    //Método que muestra las naves ordenadas por nombre de Z a A
    public static void ordenarPorNombreInsercion(Nave[] flota){
    }
}

```