# ORDENACIÓN DE ELEMENTOS EN UNA LISTA

## ORDENAR UN ARRAY Y UN ARRAYLIST DE TIPOS SIMPLES

Si queremos ordenar un array de tipos simples en orden ascendente, cómo puede ser un array de enteros (int) o un array de cadenas (String), podemos usar el método estático *Array.sort(…)*.

```
String []nombres={"pepe","juan","maria","ana"};
Arrays.sort(nombres);   //Ordenamos el array

int[] enteros={6,9,1,4,5};
Arrays.sort(enteros);
```

Para hacerlo en orden descendente, podemos usar un segundo parámetro en el método *Collection.reverseOrder()*. Este tipo de ordenamiento solo funciona con objetos, por lo que no lo podemos usarlo con un array de int; pero, sí podemos usarlo con el tipo no primitivo asociado a int que es Integer.

```
String []nombres={"pepe","juan","maria","ana"};
//Ordenamos el array en orden descendente
Arrays.sort(nombres, Collections.reverseOrder());

Integer[] enteros={6,9,1,4,5};
Arrays.sort(enteros, Collections.reverseOrder());
```

Si en lugar de tener un array tenemos una colección del tipo ArrayList, la ordenación se hace con el método estático de la clase Collection. Este método también admite el segundo parámetro para orden descendente.

```
ArrayList<String> listaNombres=new ArrayList();
listaNombres.add("pepe");
listaNombres.add("juan");
listaNombres.add("maria");
listaNombres.add("ana");

Collections.sort(listaNombres); //orden ascendente

//Orden descendente
Collections.sort(listaNombres,Collections.reverseOrder());
```

## ORDENAR UNA LISTA DE OBJETOS NO SIMPLES

Pero, ¿que ocurre si intentamos ordenar un array (o un ArrayList) de un tipo no simple como podría ser el tipo Persona? ¿Como sabe JAVA que una persona es mayor que otra?

RESPUESTA: No lo sabe, se lo tenemos que explicar nosotros.

Imagina que tenemos definida la clase persona que tiene 3 atributos: nombre, edad y altura. ¿Cómo va a saber JAVA como ordenar dos personas? ¿Las ordenamos por nombre, por edad o por altura?

```
static class Persona {

    public String nombre;
    public int edad, altura;
    // Constructor de la clase
    public Persona(String nombre, int edad, int altura) {
        this.nombre = nombre;
        this.edad = edad;
        this.altura = altura;
    }
```

Para poder ordenar una colección de elementos no simples, debemos hacer que dichos elementos implementen la clase *Comparable*, y que programen el método *compareTo(…)*.

Este método compara el objeto que hace la llamada con el objeto pasado por parámetro. Según el criterio de ordenación que queramos aplicar, devolverá lo siguiente:

- Debe devolver -1 si el objeto llamador es menor que el objeto del parámetro
- Debe devolver 1 si el objeto llamador es mayor que el objeto del parámetro
- Debe devolver 0 si son iguales.

Vamos a ordenar las personas por su altura. Haremos que la clase Persona implemente la interfaz *Comparable<Persona>* (los símbolos <…> es para parametrizar una Clase/Interfaz) y que programe el método *compareTo(…)* en base al atributo altura.

```java
static class Persona implements Comparable<Persona> {

        public String nombre;
        public int edad, altura;

        public Persona(String nombre, int edad, int altura) {
            this.nombre = nombre;
            this.edad = edad;
            this.altura = altura;
        }

        @Override
        public int compareTo(Persona o) {
            if (altura < o.altura) {
                return -1;
            }
            if (altura > o.altura) {
                return 1;
            }
            return 0;
        }
    }
```

Una vez hecho, JAVA ya esta en disposición de saber ordenar objetos de tipo Persona (en este caso por su altura). Ya podemos ordenar un array de Personas tal y como lo hemos hecho en el primer apartado. (el método imprimeArrayPersonas(...) simplemente imprime el array).

```java
public static void main(String[] args) {
    Persona[] arrayPersonas = new Persona[5];
    arrayPersonas[0] = new Persona("Mario", 22, 187);
    arrayPersonas[1] = new Persona("Pepe", 52, 173);
    arrayPersonas[2] = new Persona("Manuel", 27, 158);
    arrayPersonas[3] = new Persona("David", 25, 164);
    arrayPersonas[4] = new Persona("Alberto", 80, 184);

    System.out.println("Array sin ordenar por altura");
    imprimeArrayPersonas(arrayPersonas);
    // Ordeno el array de personas por altura (de menor a mayor)
    Arrays.sort(arrayPersonas);
    System.out.println("Array ordenado por altura");
    imprimeArrayPersonas(arrayPersonas);
}
```

Dependiendo si estamos ordenando un array o un ArrayList, usaremos *Arrays.sort()* o *Collections.sort()* respectivamente.