

Programación 2

TUDAI

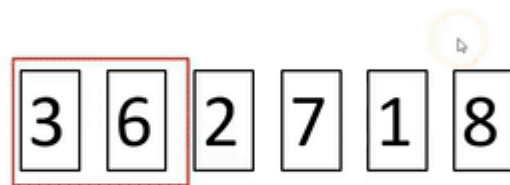
**Comparación de Objetos:
Comparable y Comparator**

Comparación de Objetos

```
public void ordenar(int[] arr) {  
    int n = arr.length;  
    int temp = 0;  
    for(int i=0; i < n; i++){  
        for(int j=1; j < (n-i); j++){  
            //si el elemento siguiente es mayor al actual  
            if(arr[j-1] > arr[j]){  
                //intercambio los elementos  
                temp = arr[j-1];  
                arr[j-1] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```

¿Y si el
arreglo no
es de
enteros?

Bubble Sort:

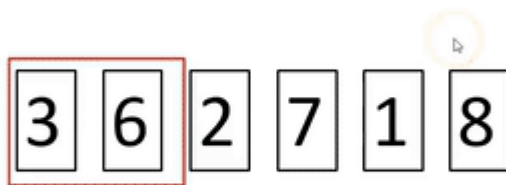


Comparación de Objetos

```
public void ordenar(Object[] arr) {  
    int n = arr.length;  
    int temp = 0;  
    for(int i=0; i < n; i++){  
        for(int j=1; j < (n-i); j++){  
            //si el elemento siguiente es mayor al actual  
            if(arr[j-1] > arr[j]){  
                //intercambio los elementos  
                temp = arr[j-1];  
                arr[j-1] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```

¿Y si el
arreglo no
es de
enteros?

Bubble Sort:



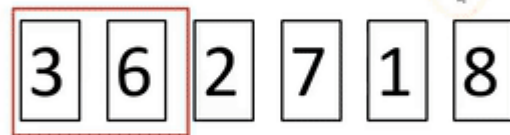
¿Cómo comparo dos Objetos?

Comparación de Objetos

```
public void ordenar(Object[] arr) {  
    int n = arr.length;  
    int temp = 0;  
    for(int i=0; i < n; i++){  
        for(int j=1; j < (n-i); j++){  
            //si el elemento siguiente es mayor al actual  
            if(arr[j-1] > arr[j]){  
                //intercambio los elementos  
                temp = arr[j-1];  
                arr[j-1] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```

¿Y si el
arreglo no
es de
enteros?

Bubble Sort:



A un objeto siempre se
le envían mensajes


¿Cómo comparo dos Objetos?



Comparación de Objetos

— — —

```
public void ordenar(Object[] arr) {  
    int n = arr.length;  
    int temp = 0;  
    for(int i=0; i < n; i++){  
        for(int j=1; j < (n-i); j++){  
            //si el elemento siguiente es mayor al actual  
            if( arr[j-1].esMenor(arr[j]) ){  
                //intercambio los elementos  
                temp = arr[j-1];  
                arr[j-1] = arr[j];  
                arr[j] = temp;  
            }  
        }  
    }  
}
```



Pero Object
no tiene ese
método...

Comparable



- Cuando una clase implementa la interfaz Comparable está obligada a definir el método:

```
public int compareTo(Object o)
```

- Este método permite comparar instancias de dicha clase entre sí. Por ejemplo, si mi clase Persona implementa Comparable, puedo hacer

```
Persona p1 = new Persona(...)  
Persona p2 = new Persona(...)  
int comparación = p1.compareTo(p2);
```

comparación {	< 0	si	p1 < p2
	= 0	si	p1 = p2
	> 0	si	p1 > p2

Comparable

- Si una clase implemente **Comparable** se habilita la posibilidad de usar el método **Collections.sort(List)** para ordenar listas de ese tipo de objetos.

```
ArrayList< Persona > personas = new ArrayList<>();  
personas.add(p1);  
personas.add(p2);  
Collections.sort(personas);
```

- Los elementos se ordenan según la lógica del **compareTo**.
- Si los objetos almacenados NO son **Comparable**, entonces NO puedo utilizar **Collections.sort(List)**.

Comparable

- **Comparable** solo me permite definir **UNA** forma de comparar los elementos (la que establezca en el **compareTo**), y esa forma no puede cambiar.
- Si, por ejemplo, quiero modificar la forma en la que mi lista de Personas se ordena, no puedo hacerlo. Tengo que cambiar el **compareTo** y recompilar todo mi código.
- ¿Y si quiero cambiar la forma en la que se ordenan as personas? ¿Que hago cuando quiero que algo pueda cambiar dinámicamente?

Encapsulo esa porción de lógica en un objeto y la saco afuera → **Comparator**

— — —

Comparator



- Cuando una clase implementa la interfaz **Comparator**, debe implementar el método

```
public int compare(Object o1, Object o2)
```

- En este método, se comparan los dos objetos (o1 y o2) de la forma que se necesite. Por ejemplo, puedo tener una clase:

```
ComparadorProcesosMemoria implements Comparator { ...
```

que compare dos procesos según su consumo de memoria. Y otra clase:

```
ComparadoProcesosCPU implements Comparator { ...
```

que compare mis procesos según su consumo de CPU.

Comparator

- Definir un Comparator me permite utilizar el método `Collections.sort(List,Comparator)`
- Este método ordena los elementos de una lista (primer parámetro), según la manera de comparar los elementos de un comparator (segundo parámetro).
- En este punto, al especificar un Comparator, NO es necesario que la clase de los objetos que se van a ordenar (por ejemplo, Proceso) sea Comparable.

Comparator

Por ejemplo, con Comparator puedo hacer:

```
ArrayList<Proceso> procesos = new ArrayList<>();  
procesos.add(p1);  
procesos.add(p2);  
Comparator comparadorMem = new ComparadorProcesosMemoria();  
Collections.sort(procesos, comparadorMem);  
Comparator comparadorCPU = new ComparadorProcesosCPU();  
Collections.sort(procesos, comparadorCPU);
```

Comparator

- Con este mismo método sort,
`Collections.sort(List,Comparator)`
- puedo ordenar por orden inverso del orden dado por Comparable:
`Collections.sort(list, Collections.reverseOrder())`
- Y el orden inverso dado por un comparador dado:
`ComparadorMemoria comp = new ComparadorMemoria()
Collections.sort(list, Collections.reverseOrder(comp));
Collections.sort(list, comp.reversed())`

Evitar usar casting

- Para evitar hacer casting de Object a la clase que queremos comparar, tanto en **Comparable** como en **Comparator** podemos indicar el tipo de clase que vamos a comprar.

- De la siguiente manera en Comparable:

```
public class Proceso implements Comparable<Proceso> {  
    public int compareTo(Proceso otroProceso) { ... }  
}
```

- O, en Comparator:

```
public class ComparadorProcesoCPU implements Comparator<Proceso> {  
    public int compare(Proceso p1, Proceso p2) { ... }  
}
```

Comparable vs Comparator

— — —

Comparable	Comparator
Se utiliza para ordenar objetos con su orden “natural”	Se usa para ordenar objetos de acuerdo a los valores de diferentes atributos
Utiliza el método compareTo(Object otroObjeto)	Utiliza el método compara(Object o1, Object o2)
Compara “this” con respecto a otro objeto	Compara dos objetos recibidos como parámetro
Está en el paquete java.Lang, junto a la clase Object	Está en el paquete java.util
Debe ser implementado por la clase original, aquella que quiero que sus instancias se puedan comparar (por ejemplo, Persona, Computadora, Proceso)	No afecta a la clase original. Se implementan clases independientes de la misma.