Ordenación de objetos en JAVA.

Repaso de los videos:
[Video 18 - Colecciones ArrayList Búsqueda y Ordenación (parte 1)]
[Video 18 - Colecciones ArrayList Búsqueda y Ordenación (parte 2)]

Pueden descargar el código del [repositorio de GitHub].

Ordenar por un criterio solo (orden natural)

Se le dice orden natural dado que el criterio de ordenación queda definido dentro de la clase y solo puede tener una implementación del método compareTo.

1.- Implementar la interfaz comparable en la clase de los objetos que queremos ordenar.

```
public class Camion implements Comparable<Camion> {

    private String chapa;
    private int carga;

    public Camion(String chapa, int carga) {
        this.chapa = chapa;
        this.carga = carga;
    }
```

2.- Implementar el método compareTo.

```
@Override
public int compareTo(Camion unCamion) {
    return this.getCarga() - unCamion.getCarga();
}
```

3.- Ordenar la lista de objetos utilizando el collection, pasamos la lista al Collection.sort y ordena sobre la misma lista. Por ejemplo, Collections.sort(camiones).

```
public static void main(String[] args) {
    ArrayList<Camion> camiones = new ArrayList<>();
    camiones.add(new Camion("SAA1212", 150));
    camiones.add(new Camion("AAA1212", 350));
    camiones.add(new Camion("BAA1212", 50));
    camiones.add(new Camion("ZZZ1212", 280));

    System.out.println("camiones = " + camiones);
    Collections.sort(camiones);
    System.out.println("camiones = " + camiones);
}
```

Ordenar por varios criterios

1.- Crear una clase de comparación que implemente la interfaz comparator y sobreescriba el método compare.

```
public class CriterioCarga implements Comparator<Camion> {

    @Override
    public int compare(Camion unCamion, Camion otroCamion) {
        return unCamion.getCarga() - otroCamion.getCarga();
    }

}
```

Podemos crear tantas clases comparadoras como necesitemos, una para cada criterio de ordenación.

```
public class CriterioChapa implements Comparator<Camion> {

    @Override
    public int compare(Camion unCamion, Camion otroCamion) {
        return unCamion.getChapa().compareTo(otroCamion.getChapa());
    }

}
```

2.- Ordenar la lista de objetos utilizando el collection, pasamos la lista y una instancia del comparador al Collection.sort, ordena sobre la misma lista.
Por ejemplo, Collections.sort(camiones, new CriterioCarga()).

Para ordenar por distintos criterios solo hay que cambiar la instancia del comparador que le pasamos al collection.sort (el segundo parámetro).

```
public static void main(String[] args) {
    ArrayList<Camion> camiones = new ArrayList<>();
    camiones.add(new Camion("SAA1212", 150));
    camiones.add(new Camion("AAA1212", 350));
    camiones.add(new Camion("BAA1212", 50));
    camiones.add(new Camion("ZZZ1212", 280));

    System.out.println("camiones = " + camiones);
    System.out.println("Ordeno por Criterio Chapa");
    Collections.sort(camiones, new CriterioChapa());
    System.out.println("camiones = " + camiones);
    System.out.println("");
    System.out.println("Ordeno por Criterio Carga");
    Collections.sort(camiones, new CriterioCarga());
    System.out.println("camiones = " + camiones);
    System.out.println("");
}
```

Ejemplos de implementaciones de compareTo/compare

El método compareTo y compare son muy parecidos, tienen el mismo objetivo sirven para indicar el orden de los objetos definiendo un criterio.
La diferencia es que el método *compareTo* está dentro de la clase a ordenar, por lo que usa la instancia “this” y un parámetro, mientras que el *compare* no está dentro de la clase a ordenar, por lo que recibe las dos instancias de los objetos a ordenar por parámetro.

```
// Orden creciente por carga
public int compareTo(Camion unCamion){
	return this.getCarga() - unCamion.getCarga();
}

// Orden decreciente por carga (la resta es al revés)
public int compareTo(Camion unCamion){
	return unCamion.getCarga() - this.getCarga();
}

// Ordenar por un string, por ejemplo chapa. Se usa el compareTo de strings.
public int compareTo(Camion unCamion){
	return this.getChapa().compareTo(unCamion.getChapa());
}

// Odenar por criterios compuestos, primero por uno y si son iguales aplicamos otro. Por ejemplo, ordenar por chapa y para los que tienen la misma chapa ordenamos por carga.
public int compareTo(Camion unCamion){
	int respuesta = this.getChapa().compareTo(unCamion.getChapa());
	if ( respuesta == 0){ // Si el compareTo da cero es que tienen la misma chapa.
		respuesta = this.getCarga() - unCamion.getCarga();
	}
	return respuesta;
}
```