1. Java比較器的使用背景：

*Java中的对象，正常情况下，只能进行比较：== 或 != 。不能使用 > 或 < 的  
但是在开发场景中，我们需要对多个对象进行排序，言外之意，就需要比较对象的大小。如何实现？使用两个接口中的任何一个：Comparable 或 Comparator*

1. 自然排序：使用comparaable接口
   1. 說明

*1.像String、包装类等实现了Comparable接口，重写了compareTo(obj)方法，给出了比较两个对象大小的方式。  
2.像String、包装类重写compareTo()方法以后，进行了从小到大的排列  
3. 重写compareTo(obj)的规则：  
 如果当前对象this大于形参对象obj，则返回正整数，  
 如果当前对象this小于形参对象obj，则返回负整数，  
 如果当前对象this等于形参对象obj，则返回零。  
4. 对于自定义类来说，如果需要排序，我们可以让自定义类实现Comparable接口，*

*重写compareTo(obj)方法。 在compareTo(obj)方法中指明如何排序*

* 1. 自定義類代碼舉例：

public class Goods implements Comparable{  
 private String name;  
 private double price;  
 *//指明商品比较大小的方式:按照价格从低到高排序,再按照产品名称从高到低排序* @Override  
 public int compareTo(Object o) {  
*// System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*");* if(o instanceof Goods){  
 Goods goods = (Goods)o;  
 *//方式一：* if(this.price > goods.price){  
 return 1;  
 }else if(this.price < goods.price){  
 return -1;  
 }else{  
*// return 0;* return -this.name.compareTo(goods.name);  
 }  
 *//方式二：  
// return Double.compare(this.price,goods.price);* }  
*// return 0;* throw new RuntimeException("传入的数据类型不一致！");  
 }  
}

*小結論:自定義類要比較大小來排序要先繼承Comparable,再重寫compareTo(obj),重寫方法要先把obj轉換成當前自定義類型,再去比較要比較的條件,返回值可以用包裝類的compare(obj1,obj2)來返回(由低到高),如要由高到低加"-": -compare(obj1,obj2)*

1. 定制排序：使用comparator 接口
   1. 說明
2. *背景：  
   当元素的类型没有实现java.lang.Comparable接口而又不方便修改代码，  
   或者实现了java.lang.Comparable接口的排序规则不适合当前的操作，  
   那么可以考虑使用 Comparator 的对象来排序  
   2.重写compare(Object o1,Object o2)方法，比较o1和o2的大小：  
   如果方法返回正整数，则表示o1大于o2；  
   如果返回0，表示相等；  
   返回负整数，表示o1小于o2。*

*3.使用comparator接口,不會在要排序的自定義當中去實現,而是在方法中  
 去造一個內部類重寫compare(Object o1, Object o2),給函數去調用 例:*

*Array.sort(goods,com); / Collections(coll,com); / new TreeSet(com);*

* 1. 代碼舉例：

Comparator com = new Comparator() {

*//指明商品比较大小的方式:按照产品名称从低到高排序,再按照价格从高到低排序* @Override  
 public int compare(Object o1, Object o2) {  
 if(o1 instanceof Goods && o2 instanceof Goods){  
 Goods g1 = (Goods)o1;  
 Goods g2 = (Goods)o2;  
 if(g1.getName().equals(g2.getName())){  
 return -Double.*compare*(g1.getPrice(),g2.getPrice());  
 }else{  
 return g1.getName().compareTo(g2.getName());  
 }  
 }  
 throw new RuntimeException("输入的数据类型不一致");  
 }  
}

使用：

Array.sort(goods,com); / Collections(coll,com); / new TreeSet(com);

1. 兩種排序方式對比

*Comparable接口的方式一旦一定，保证Comparable接口实现类的对象在任何位置都可以比较大小。  
Comparator接口属于临时性的比较。*