1. Comparator 和 Comparable 相同的地方  
  
他们都是java的一个接口, 并且是用来对自定义的class比较大小的,  
  
什么是自定义class: 如 public class Person{ String name; int age }.  
  
当我们有这么一个personList,里面包含了person1, person2, persion3....., 我们用Collections.sort( personList ),   
是得不到预期的结果的. 这时肯定有人要问, 那为什么可以排序一个字符串list呢:  
  
如 StringList{"hello1" , "hello3" , "hello2"}, Collections.sort( stringList ) 能够得到正确的排序, 那是因为   
String 这个对象已经帮我们实现了 Comparable接口 , 所以我们的 Person 如果想排序, 也要实现一个比较器。  
   
2. Comparator 和 Comparable 的区别  
  
Comparable  
  
Comparable 定义在 Person类的内部:  
  
public class Persion implements Comparable {..比较Person的大小..},  
  
 因为已经实现了比较器,那么我们的Person现在是一个可以比较大小的对象了,它的比较功能和String完全一样,可以随时随地的拿来  
比较大小,因为Person现在自身就是有大小之分的。Collections.sort(personList)可以得到正确的结果。  
  
Comparator  
  
Comparator 是定义在Person的外部的, 此时我们的Person类的结构不需要有任何变化,如  
  
public class Person{ String name; int age },  
  
然后我们另外定义一个比较器:  
  
public PersonComparator implements Comparator() {..比较Person的大小..},  
  
在PersonComparator里面实现了怎么比较两个Person的大小. 所以,用这种方法,当我们要对一个 personList进行排序的时候,   
我们除了了要传递personList过去, 还需要把PersonComparator传递过去,因为怎么比较Person的大小是在PersonComparator  
里面实现的, 如:  
  
Collections.sort( personList , new PersonComparator() ).  
  
3. Comparator 和 Comparable 的实例  
  
Comparable:  
  
实现Comparable接口要覆盖compareTo方法, 在compareTo方法里面实现比较：  
public class Person implements Comparable {  
 String name;  
 int age  
 public int compareTo(Person another) {  
 int i = 0;  
 i = name.compareTo(another.name); // 使用字符串的比较  
 if(i == 0) { // 如果名字一样,比较年龄, 返回比较年龄结果  
 return age - another.age;  
 } else {  
 return i; // 名字不一样, 返回比较名字的结果.  
 }  
 }  
}  
 这时我们可以直接用 Collections.sort( personList ) 对其排序了.  
  
Comparator:  
  
实现Comparator需要覆盖 compare 方法：  
public class Person{  
 String name;  
 int age  
}  
  
class PersonComparator implements Comparator {   
 public int compare(Person one, Person another) {  
 int i = 0;  
 i = one.name.compareTo(another.name); // 使用字符串的比较  
 if(i == 0) { // 如果名字一样,比较年龄,返回比较年龄结果  
 return one.age - another.age;  
 } else {  
 return i; // 名字不一样, 返回比较名字的结果.  
 }  
 }  
}  
 Collections.sort( personList , new PersonComparator()) 可以对其排序  
  
 4:总结  
  
两种方法各有优劣, 用Comparable 简单, 只要实现Comparable 接口的对象直接就成为一个可以比较的对象,  
但是需要修改源代码, 用Comparator 的好处是不需要修改源代码, 而是另外实现一个比较器, 当某个自定义  
的对象需要作比较的时候,把比较器和对象一起传递过去就可以比大小了, 并且在Comparator 里面用户可以自  
己实现复杂的可以通用的逻辑,使其可以匹配一些比较简单的对象,那样就可以节省很多重复劳动了。