Java 比较器 Comparable 与 Comparator

Comparable接口

Comparable可以认为是一个**内比较器**，实现了Comparable接口的类有一个特点，就是这些类是可以和自己比较的，至于具体和另一个实现了Comparable接口的类如何比较，则依赖compareTo方法的实现，compareTo方法也被称为**自然比较方法**。如果开发者add进入一个Collection的对象想要通过Collections.sort（或 Arrays.sort）进行排序的话，那么这个对象必须实现Comparable接口。compareTo方法的返回值是int，有三种情况：

1、比较者大于被比较者（也就是compareTo方法里面的对象），那么返回正整数

2、比较者等于被比较者，那么返回0

3、比较者小于被比较者，那么返回负整数

Comparable 定义

```
package java.lang;
import java.util.*;
public interface Comparable<T> {
    public int compareTo(T o);
}
```

实现`Comparable`接口的类必须实现`compareTo()`方法，对象就可以比较大小。假设我们通过`x.compareTo(y)`来“比较 x 和 y 的大小”。若**返回 负数，意味着 “x 比 y 小”；返回 “零”，意味着 “x 等于 y”；返回 “正数”，意味着 “x 大于 y”。**

```
public class Student implements Comparable {
     String name;
     public int compareTo(Student another) {
         return name.compareTo(another.name);
     }
}

// 比较两个对象
student1.compareTo(student2);

// 排序数组或集合
Arrays.sort(students);
Collections.sort(collection);
```

Java 的一些常用类已经实现了 Comparable 接口，并提供了比较大小的标准。比如包装类按照它们对应的数值大小进行比较。String，Date，Time 等也都实现了 Comparable 接口。一个类如果实现 Comparable 接口，那么它就具有了可比较性。

demo

```
public class Domain implements Comparable<Domain>
{
   private String str;

   public Domain(String str)
   {
       this.str = str;
   }

   public int compareTo(Domain domain)
   {
       if (this.str.compareTo(domain.str) > 0)
           return 1;
       else if (this.str.compareTo(domain.str) == 0)
           return 0;
       else
           return -1;
   }

   public String getStr()
   {
       return str;
   }
}
public static void main(String[] args)
   {
       Domain d1 = new Domain("c");
       Domain d2 = new Domain("c");
       Domain d3 = new Domain("b");
       Domain d4 = new Domain("d");
       System.out.println(d1.compareTo(d2));
       System.out.println(d1.compareTo(d3));
       System.out.println(d1.compareTo(d4));
   }
```

输出结果为

```
0
1
-1
```

分析比较器的排序原理

实际上比较器的操作，就是经常听到的二叉树的排序算法。

排序的基本原理：使用第一个元素作为根节点，之后如果后面的内容比根节点小，则放在左子树，如果内容比根节点的内容要大，则放在右子树。

Comparator

Comparator可以认为是是一个**外比较器**，个人认为有两种情况可以使用实现Comparator接口的方式：

1、一个对象**不支持自己和自己比较**（没有实现`Comparable`接口），但是又想对两个对象进行比较

2、一个对象实现了`Comparable`接口，但是开发者认为`compareTo`方法中的比较方式并不是自己想要的那种比较方式

Comparator 定义

```
package java.util;
public interface Comparator<T> {
    int compare(T o1, T o2);
    boolean equals(Object obj);
}
```

若一个类要实现`Comparator`接口，它一定要实现`compareTo(T o1, T o2)`方法，但可以不实现`equals(Object obj)`方法。Object 类是所有类的父类，也就是说实现接口的子类已经重写了 equals 方法。

int compare(T o1, T o2) 是 “比较 o1 和 o2 的大小”。返回 “负数”，意味着 “o1 比 o2 小”；返回 “零”，意味着 “o1 等于 o2”；返回 “正数”，意味着 “o1 大于 o2”。

在不希望修改一个原有的类，或提供的比较器不适用时，就需要使用外部比较器，比如 String 类实现 Comparable，而且 String.compareTo() 是按字典序比较字符串的，这时如果需要按长度对字符串进行排序，就不能让 String 类用两种不同的方法实现 compareTo 方法了，更何况，String 类也不应该由我们来修改，这时就需要使用外部比较器：

```
class LengthComparator implements Comparator<String> {
    public int compare(String f, String s) {
        return f.length() - s.length();
    }
}

// 比较两个对象
LengthComparator comparator = new LengthComparator();
comparator.compare(person1,person2);

// 排序数组或集合
Arrays.sort(arr, new LengthComparator());
Collections.sort(collection, new LengthComparator());
```

Comparator 体现了设计模式中的策略模式，就是不改变对象自身，而用**一个策略对象来改变它的行为。**

Comparator 和 Comparable 区别

内部比较器 Comparable 是排序接口，只包含一个函数 compareTo()；若一个类实现了 Comparable 接口，就意味着 “该类支持排序”，它可以直接通过 Arrays.sort() 或 Collections.sort() 进行排序。

外部比较器 Comparator 是比较器接口，单独实现第一个比较器，不需要对原来的类进行结构上的变化，属于无侵入式的；一个类实现了 Comparator 接口，那么它就是一个 “比较器”。其它的类，可以根据该比较器去排序。

一个类本身实现了 Comparable 接口，就意味着它本身支持排序；若它本身没实现 Comparable，也可以通过外部比较器 Comparator 进行排序。

- 如果比较的方法只要用在一个类中，用该类实现 Comparable 接口就可以。

- 如果比较的方法在很多类中需要用到，就自己写个类实现 Comparator 接口，这样当要比较的时候把实现了 Comparator 接口的类传过去就可以，省得重复造轮子。这也是为什么 Comparator 会在 java.util 包下的原因。

使用 Comparator 的优点：

1. 与实体类分离
2. 方便应对多变的排序规则，可以同时使用多种排序标准