

## 1 IComparable、IComparer接口、Comparer类

2

3 -----

4

5 IComparable和IComparable 接口是.net framework 中比较对象的标准方式，这两个接口之间的区别如下：

6 1. IComparable 在要比较的对象的类中实现，可以比较该对象和另一个对象。

7 2.IComparer 在一个单独的类中实现，可以比较任意两个对象。

8 一般情况下，我们使用 IComparable 给出类的默认比较代码，使用其他类给出非默认的比较❖

❖❖码。

9

10 -----

11

12 一、IComparable提供了一个方法int CompareTo(object obj)。这个方法接受一个对象，所以可以实现这个❖

❖❖口

13 比如：以便把 Person 对象传送给它，

14 说明这个人是否比当前的人更年老或年轻。实际上，这个方法返回一个int，所以可和下面的代❖

❖❖说明第二个人更年老还是更年轻。

```
15         if(person1.CompareTo(person2) == 0)
16         {
17             Console.WriteLine("Same age");
18         }
19         else if(person1.CompareTo(person2) > 0 )
20         {
21             Console.WriteLine("person 1 is older");
22         }
23         else
24         {
25             Console.WriteLine("person1 is younger");
26         }
27
```

28 -----

29

30 二、IComparer 也提供了一个方法 Compare()。这个方法接受两个对象，返回一个整型结果，这与 Compar

eTo()相同。

31 对于支持 IComparer的对象，可以使用下面的代码：

```
32         if(personComparer.Compare(person1,person2) == 0)
33         {
34             Console.WriteLine("same age");
35         }
36         else if(personComparer.Compare(person1,person2) > 0 )
37         {
```

```

38         Console.WriteLine("person 1 is older");
39     }
40     else
41     {
42         Console.WriteLine("person1 is younger");
43     }

```

44 在这两种情况下，提供给方法的参数是system.object类型。也就是说，可以比较任意类型的两个对象。

45 所以，在返回结果之前，通常需要进行某种类型比较，

46 如果使用了错误的类型，还会抛出异常。实际上，我们是使用泛型接口IComparable<T>，可以省略❖

❖象转换。

```

47
48 -----
49

```

50 三、.net framework 在类 Comparer 上提供了 IComparer 接口的默认实现方式，  
51 类 Comparer 位于 system.collections 名称空间中，可以对简单类型以及支持

IComparable

52 接口的任意类型进行特定文化的比较。例如，可以通过下面的代码使用它：

```

53
54     string firststring = "First String";
55     string secondstring = "Second string";
56     Comparer.Default.Compare(firststring , secondstring);
57
58     int firstNumber = 35;
59     int secondNumber = 23;
60     Comparer.Default.Compare(firstNumber , secondNumber);
61

```

62 这里使用Comparer.Default静态成员获取Comparer类的一个实例，接着使用 Compare()方法来比较。

63 在使用 Comparer时，必须使用可以比较的类型。例如，试图比较 firstString 和 firstNumber 就会生成一个异常。

```

64
65 -----
66

```

67 List<T>.sort()可以实现对T的排序，比如List<int>.sort()执行后集合会按照int从小到大排序。

68 如果T是一个自定义的Object，可是我们想按照自己的方式来排序，那该怎么办呢，  
69 其实可以用过IComparable接口重写CompareTo方法来实现。流程如下：

70  
71 一.第一步我们申明一个类Person但是要继承IComparable接口：

```

72
73     using System;
74     using System.Collections.Generic;
75     using System.Linq;
76     using System.Text;

```

```

77 using System.Threading.Tasks;
78 namespace TestComparable
79 {
80     public class Person : IComparable<Person>
81     {
82         public string Name { get; set; }
83         public int Age { get; set; }
84         public int CompareTo(Person obj)
85         {
86             int result;
87             if (this.Name == obj.Name && this.Age == obj.Age)
88             {
89                 result = 0;
90             }
91             else
92             {
93                 if (this.Name.CompareTo(obj.Name) > 0)
94                 {
95                     result = 1;
96                 }
97                 else if (this.Name == obj.Name && this.Age > obj.Age)
98                 {
99                     result = 1;
100                 }
101                 else
102                 {
103                     result = -1;
104                 }
105             }
106             return result;
107         }
108         public override string ToString()
109         {
110             return this.Name + "-" + this.Age;
111         }
112     }
113 }

```

115 二. 然后在主函数里面调用sort方法即可. 类就会按照姓名从小到大, 如果姓名相同则按照  
 年龄

小到大排序了。

```

116
117 public class Program
118 {
119     public static void Main(string[] args)
120     {
121         List<Person> lstPerson = new List<Person>();
122         lstPerson.Add(new Person() { Name="Bob", Age=19});

```

```

123         lstPerson.Add(new Person(){ Name="Mary",Age=18});
124         lstPerson.Add(new Person() { Name = "Mary", Age = 17 });
125         lstPerson.Add(new Person(){ Name="Lily",Age=20});
126         lstPerson.Sort();
127         Console.ReadKey();
128     }
129 }

```

131 三，如果不继承IComparable接口，我们该如何实现排序呢。可以使用Linq来实现。  
 132 其实效果是一样的，只是如果类的集合要经常排序的话，建议使用继承接口的方法，这样  
 可🔗

🔗🔗简化sort的代码，而且更容易让人看懂。

```

133
134 public static void Main(string[] args)
135     {
136         List<Person> lstPerson = new List<Person>();
137         lstPerson.Add(new Person(){ Name="Bob",Age=19});
138         lstPerson.Add(new Person(){ Name="Mary",Age=18});
139         lstPerson.Add(new Person() { Name = "Mary", Age = 17 });
140         lstPerson.Add(new Person(){ Name="Lily",Age=20});
141         lstPerson.Sort((x,y) =>
142         {
143             int result;
144             if (x.Name == y.Name && x.Age == y.Age)
145             {
146                 result = 0;
147             }
148             else
149             {
150                 if (x.Name.CompareTo(y.Name) > 0)
151                 {
152                     result = 1;
153                 }
154                 else if (x.Name == y.Name && x.Age > y.Age)
155                 {
156                     result = 1;
157                 }
158                 else
159                 {
160                     result = -1;
161                 }
162             }
163             return result;
164         });
165         Console.ReadKey();
166     }
167
168

```