因此，在图1-18中的第9行对compare的调用可以用来比较数组的项。第4行的带有限制的通配符用来表示如果查找数组中的最大的项，那么该comparator必须知道如何比较这些项，或者这些项的超类型的那些对象。我们可以在第26行看到，为了使用这种版本的findMax，findMax通过传递一个String数组以及一个实现comparator<String>的对象而被调用。这个对象属于CaseInsensitiveCompare类型，它是我们编写的一个类。

```
1       // Generic findMax, with a function object.
2       // Precondition: a.size( ) > 0.
3       public static <AnyType>
4       AnyType findMax( AnyType [ ] arr, Comparator<? super AnyType> cmp )
5       {
6           int maxIndex = 0;
7
8           for( int i = 1; i < arr.size( ); i++ )
9               if( cmp.compare( arr[ i ], arr[ maxIndex ] ) > 0 )
10                  maxIndex = i;
11
12          return arr[ maxIndex ];
13      }
14
15  class CaseInsensitiveCompare implements Comparator<String>
16  {
17      public int compare( String lhs, String rhs )
18        { return lhs.compareToIgnoreCase( rhs ); }
19  }
20
21  class TestProgram
22  {
23      public static void main( String [ ] args )
24      {
25          String [ ] arr = { "ZEBRA", "alligator", "crocodile" };
26          System.out.println( findMax( arr, new CaseInsensitiveCompare( ) ) )
27      }
28  }
```

图1-18 利用一个函数对象作为第2个参数传递给findMax；输出ZEBRA

```
1   package java.util;
2
3   public interface Comparator<AnyType>
4   {
5       int compare( AnyType lhs, AnyType rhs );
6   }
```

图1-19 Comparator接口

在第4章我们将给出关于一个类的例子，这个类需要将它存储的项排序。我们将利用Comparable编写大部分的代码，并指出其中需要使用函数对象的改动部分。在本书的其他地方，我们将避免函数对象的细节以使得代码尽可能地简单，我们知道以后将函数对象添加进去并不困难。