```
1   class FindMaxDemo
2   {
3       /**
4        * Return max item in arr.
5        * Precondition: arr.length > 0
6        */
7       public static Comparable findMax( Comparable [ ] arr )
8       {
9           int maxIndex = 0;
10
11          for( int i = 1; i < arr.length; i++ )
12              if( arr[ i ].compareTo( arr[ maxIndex ] ) > 0 )
13                  maxIndex = i;
14
15          return arr[ maxIndex ];
16      }
17
18      /**
19       * Test findMax on Shape and String objects.
20       */
21      public static void main( String [ ] args )
22      {
23          Shape [ ] sh1 = { new Circle( 2.0 ),
24                            new Square( 3.0 ),
25                            new Rectangle( 3.0, 4.0 ) };
26
27          String [ ] st1 = { "Joe", "Bob", "Bill", "Zeke" };
28
29          System.out.println( findMax( sh1 ) );
30          System.out.println( findMax( st1 ) );
31      }
32  }
```

图 1-8　一般的 findMax 例程，使用 Shape 和 String 演示(pre-Java 5)

乍一看，该问题不值得一问，似乎 Employee[]就应该是和 Person[]类型兼容的。然而，这个问题却要比想象的复杂。假设除 Employee 外，我们还有 Student *IS-A* Person，并设 Employee[]是和 Person[]类型兼容的。此时考虑下面两条赋值语句：

```
Person[] arr = new Employee[ 5 ]; // 编译: arrays are compatible
arr[ 0 ] = new Student( ... );    // 编译: Student IS-A Person
```

两句都编译，而 arr[0]实际上是引用一个 Employee，可是 Student *IS-NOT-A* Employee。这样就产生了类型混乱。运行时系统(runtime system)(Java 虚拟机—译者注)不能抛出 ClassCastException 异常，因为不存在类型转换。

避免这种问题的最容易的方法是指定这些数组不是类型兼容的。可是，在 Java 中数组却是类型兼容的。这叫做**协变数组类型**(covariant araay type)。每个数组都明了它所允许存储的对象的类型。如果将一个不兼容的类型插入到数组中，那么虚拟机将抛出一个 ArrayStoreException 异常。

在较早版本的 Java 中是需要数组的协变性的，否则在图 1-8 的第 29 行和第 30 行的调用将编译不了。