# 枚举的使用

## ****没有枚举之前：****

　在没有枚举之前，我们想列举一些相关的常量，我们会采用如下的方式：

|  |
| --- |
| interface ActionInterface {      public static final int RIGHT = 0;      public static final int LEFT = 1;  } |

然后在某个类似于下面的方法中，使用这些常量：

|  |
| --- |
| public void playWithInterface(int num) {      switch (num) {      case ActionInterface.RIGHT:          System.out.println("RIGHT");          break;      case ActionInterface.LEFT:          System.out.println("LEFT");          break;      default:          System.out.println("Default");      }  } |

不知道你注意了没，在这样做的时候，请记住这类常量是 Java 中 int 类型的常量，这意味着该方法可以接受任何 int 类型的值，即使它和之前的接口中定的所有常量都不对应。因此，您需要检测上界和下界，在出现无效值的时候，你只能依赖switch中的default或者其他的处理手段，而且，如果后来又添加另外一个常量的时候，那么你就必须改变很多相关联代码中的上界，下界处理程序，才能接受这个新值。

## 有了枚举之后：

　　你应该对于这种情况，总是使用枚举，比如：

|  |
| --- |
| enum ActionEnum{      RIGHT,LEFT  } |

然后在某个类似于下面的方法中，使用它们：

|  |
| --- |
| public void playWithEnum(ActionEnum actionEnum) {      switch (actionEnum) {      case RIGHT:          System.out.println("RIGHT");          break;      case LEFT:          System.out.println("LEFT");          break;      default:          System.out.println("Default");      }  } |

这样，你就不必费很大的心思去检查之前的一些问题，而且即便需求修改之后，添加了一些相关的变量，你只需要修改ActionEnum就行，其余代码都不需要修改。

　　需要注意的是，在上面的switch中，我们使用的是case RIGHT... 而不是case ActionEnum.RIGHT（这样会出错），原因是枚举重写了ToString(),也就是说枚举变量的值是不带前缀的。

上面所说的，可能并不能完全显示出枚举类型的优势，但是确实我们经常采用的形式。下面我们来尽可能的认识一下枚举：

|  |
| --- |
| package com.java;    public class EnumDemo {      /\*\*       \* 最简单的枚举示例       \* \*/      public enum SimpleEnum {          UP, DOWN      }        /\*\*       \* 带有属性和方法的枚举示例       \* 注意：枚举本身也是类，可以象一般的类一样为它添加静态或者非静态的属性或方法       \*        但是枚举列表必须写在最前面，否则编译出错。比如本例中的UP，DOWN       \* \*/      public enum EnumWithFuncAndAttr {          UP, DOWN;          private static final String description = "这个是一个有方法和属性的枚举示例";            public String getDescription() {              return description;          }      }        /\*\*       \* 带有构造函数的枚举示例       \* 注意：枚举可以有构造方法，通过括号赋值。如果不赋值，那么就不能编写构造方法，否则出错。       \* 构造方法只能是private的或者默认的。而不能是public以及protected，否则出错。这样做可以保证客户代码没有办法新建一个enum的实例       \* \*/      public enum EnumWithConstructor{          UP("hello"),DOWN("java");          private final String value;          String getValue(){              return value;          }          EnumWithConstructor(String value){              this.value=value;          }      }        public static void main(String[] args) {          System.out.println(SimpleEnum.values().length);          System.out.println(SimpleEnum.UP);          System.out.println(SimpleEnum.valueOf("UP"));          for (EnumWithConstructor item : EnumWithConstructor.values()) {              System.out.println(item+" "+item.getValue());          }          System.out.println(SimpleEnum.UP.ordinal());      }  } |

## 总结：

* 枚举本身就是一个类。
* 它不能有public的构造函数，这样做可以保证客户代码没有办法新建一个enum的实例。
* 所有枚举值都是public static final的。注意这一点只是针对于枚举值，我们可以和在普通类里面定义变量一样定义其它任何类型的非枚举变量，这些变量可以用任何你想用的修饰符。
* Enum默认实现了java.lang.Comparable接口。
* Enum覆载了了toString方法，因此我们如果调用Color.Blue.toString()默认返回字符串”Blue”.
* Enum提供了一个valueOf方法，这个方法和toString方法是相对应的。调用valueOf(“Blue”)将返回Color.Blue.因此我们在自己重写toString方法的时候就要注意到这一点，一般来说应该相对应地重写valueOf方法。
* Enum还提供了values方法，这个方法使你能够方便的遍历所有的枚举值
* Enum还有一个oridinal的方法，这个方法返回枚举值在枚举类种的顺序。