# 枚举类

## 原理

枚举类隐式继承java.lang.Enum<E>。在编译的时候，编译器将enum类单独编译成一个字节码文件。

例如

|  |
| --- |
| public enum Color{  RED,BLUE,BLACK,YELLOW,GREEN  } |

编译后的类

|  |
| --- |
| final enum hr.test.Color {    // 所有的枚举值都是类静态常量  public static final enum hr.test.Color RED;  public static final enum hr.test.Color BLUE;  public static final enum hr.test.Color BLACK;  public static final enum hr.test.Color YELLOW;  public static final enum hr.test.Color GREEN;    private static final synthetic hr.test.Color[] ENUM$VALUES;    // 初始化过程，对枚举类的所有枚举值对象进行第一次初始化  static {  0 new hr.test.Color [1]  3 dup  4 ldc <String "RED"> [16] //把枚举值字符串"RED"压入操作数栈  6 iconst\_0 // 把整型值0压入操作数栈  7 invokespecial hr.test.Color(java.lang.String, int) [17] //调用Color类的私有构造器创建Color对象RED  10 putstatic hr.test.Color.RED : hr.test.Color [21] //将枚举对象赋给Color的静态常量RED。  ......... 枚举对象BLUE等与上同  102 return  };    // 私有构造器，外部不可能动态创建一个枚举类对象(也就是不可能动态创建一个枚举值)。  private Color(java.lang.String arg0, int arg1){  // 调用父类Enum的受保护构造器创建一个枚举对象  3 invokespecial java.lang.Enum(java.lang.String, int) [38]  };    public static hr.test.Color[] values();    // 实现Enum类的抽象方法  public static hr.test.Color valueOf(java.lang.String arg0);  } |

注：枚举类是单利模式的变形。但是所有属性和类本身都是final的。

## 枚举类方法

1. 原有类和方法
   1. compareTo（E e）比较顺序
   2. getDeclaringClass（）返回对应的class对象
   3. name（）返回此枚举类的常量名称
   4. ordinal（）返回枚举常量的序数
   5. toString（）返回此枚举类的常量名称
   6. static valueOf（enumType，name）返回带指定名称的指定枚举类型的枚举常量
2. 在枚举常量上自定义方法和属性

在枚举常量上可以重写枚举方法，定义属性。

1. 在枚举类中自定义类和方法

在声明枚举常量后面，且最后一个枚举类以分号“；”结尾，可以定义枚举属性和方法。

|  |
| --- |
| package com.hmw.test;    /\*\*  \* 枚举测试类  \*  \* @author <a href="mailto:hemingwang0902@126.com">何明旺</a>  \*/  public enum EnumTest {  MON(1), TUE(2), WED(3), THU(4), FRI(5), SAT(6) {  @Override  public boolean isRest() {  return true;  }  },  SUN(0) {  @Override  public boolean isRest() {  return true;  }  };    private int value;    private EnumTest(int value) {  this.value = value;  }    public int getValue() {  return value;  }    public boolean isRest() {  return false;  }  } |

## EnumSet和EnumMap

|  |
| --- |
| public class Test {      public static void main(String[] args) {          // EnumSet的使用          EnumSet<EnumTest> weekSet = EnumSet.allOf(EnumTest.class);          for (EnumTest day : weekSet) {              System.out.println(day);          }            // EnumMap的使用          EnumMap<EnumTest, String> weekMap = new EnumMap(EnumTest.class);          weekMap.put(EnumTest.MON, "星期一");          weekMap.put(EnumTest.TUE, "星期二");          // ... ...          for (Iterator<Entry<EnumTest, String>> iter = weekMap.entrySet().iterator(); iter.hasNext();) {              Entry<EnumTest, String> entry = iter.next();              System.out.println(entry.getKey().name() + ":" + entry.getValue());          }      }  } |

## 用法

### 常量

### Switch

### 重写枚举方法

### 添加新的方法

### 使用接口组织枚举

### 枚举集合