**枚举特点**

1.用enum定义枚举类默认继承了java.lang.Enum类而不是继承了Object类。其中java.lang.Enum类实现了java.lang.Serializable和java.lang.Comparable两个接口

1.我们所定义的枚举类型，都继承于java.lang.enum类，枚举类型的成员变量默认为public static fianl。

2. 枚举的一个成员变量是你所定义枚举类型的一个实例。一个成员对应一个枚举的一个实例

3.枚举是一种类别，跟class，interface 相似

枚举类的构造函数只能使用private访问修饰符，如果省略了其构造器的访问控制符，则默认使用private修饰

实例1：

**package** cn.com.enums;

**public** **enum** Color {

*RED*,*BLACK*,*BLUE*;

**public** **static** **void** main(String args []){

Color c = Color.*RED*;

System.*out*.println(c);

//获得枚举类型所有的枚举常量

**for**(Color c1: Color.*values*()){

System.*out*.println(c1);

}

}

}

练习2：

**package** cn.com.enums;

**public** **enum** Coin {

*BIG*(12) ,*SMALL*(6), *med*(9);

**private** **int** values;

Coin(**int** values){

**this**.values = values;

}

**int** getSize(){

**return** values;

}

**public** **static** **void** main(String[] args) {

Coin c = Coin.*BIG*;

System.*out*.println(c.getSize());

}

}

练习3：tocompare（用法）

**package** cn.com.enums;

**import** java.util.EnumSet;

**public** **class** CompareTest {

/\*\*

\* **@param** args

\*/

//比较的时枚举常量的下标（序号）

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

*myCompare*(Driect.*valueOf*("LEFT"));

System.*out*.println("======");

**for**(Driect d:Driect.*values*()){

System.*out*.println(d);

System.*out*.println(d.ordinal());//ordinal方法获得枚举常量的序号

System.*out*.println(d.toString());//获取枚举长量的名称

System.*out*.println(d.name());//获取枚举长量的名称

}

}

**public** **static** **void** myCompare(Driect d){

System.*out*.println(d);

**for**(Driect d1:Driect.*values*()){

System.*out*.println(d.compareTo(d1));

}

}

}

练习四：Swtich()用法

**package** cn.com.enums;

**public** **enum** Driect {

*UP*,*DOWN*,*RIGHT*,*LEFT*;

**public** **static** **void** getDriect(Driect d){

**switch** (d) {

**case** *UP*:

System.*out*.println("向上");

**break**;

**case** *DOWN*:

System.*out*.println("向下");

**break**;

**case** *RIGHT*:

System.*out*.println("向右");

**break**;

**case** *LEFT*:

System.*out*.println("向左");

**break**;

**default** : System.*out*.println("没有该项");

}

}

**public** **static** **void** main(String[] args) {

Driect.*getDriect*(*LEFT*);

}

}

练习5：集合的练习EnumSet

import java.util.EnumSet;

import java.util.Iterator;

import cn.com.enums.Driect;

public class EnumSetTest {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

EnumSet<Driect> set = EnumSet.of(Driect.LEFT);//创建一个有元素的enumset对象

EnumSet<Driect> set1 = EnumSet.noneOf(Driect.class);//创建一个无元素的enumset对象

set1.add(Driect.LEFT);

set1.add(Driect.RIGHT);

showEnumSet(set);

System.out.println(EnumSet.complementOf(set));

for(Driect d: EnumSet.complementOf(set)){

System.out.println(d);

}

for(Driect d1:set){

System.out.println(d1);

}

for(Driect d: set1){

System.out.println(d);

}

}

public static void showEnumSet(EnumSet<Driect> d){

for(Iterator<Driect> iterator= d.iterator();iterator.hasNext();){

System.out.println(iterator.next());

}

}

}

1．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。   
  
2．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。   
  
3．Enum默认实现了java.lang.Comparable接口。   
  
4．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”.   
  
5．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，一把来说应该相对应地重写valueOf方法。   
  
6．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。   
  
7．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，这个顺序根据枚举值声明的顺序而定，这里Color.Red.ordinal()返回0。   
  
了解了这些基本特性，我们来看看如何使用它们。   
  
1．遍历所有有枚举值. 知道了有values方法，我们可以轻车熟路地用ForEach循环来遍历了枚举值了。   
  
for (Color c: Color.values())   
System.out.println(“find value:” + c);   
  
2．在enum中定义方法和变量，比如我们可以为Color增加一个方法随机返回一个颜色。   
  
public enum Color {   
Red,   
Green,   
Blue;   
  
private static int number = Color.values().length ;   
  
public static Color getRandomColor(){   
long random = System.currentTimeMillis() % number;   
switch ((int) random){   
case 0:   
return Color.Red;   
case 1:   
return Color.Green;   
case 2:   
return Color.Blue;   
default : return Color.Red;   
}   
}   
}   
  
可以看出这在枚举类型里定义变量和方法和在普通类里面定义方法和变量没有什么区别。唯一要注意的只是变量和方法定义必须放在所有枚举值定义的后面，否则编译器会给出一个错误。   
  
3．覆载(Override)toString, valueOf方法   
  
前面我们已经知道enum提供了toString,valueOf等方法，很多时候我们都需要覆载默认的toString方法，那么对于enum我们怎么做呢。其实这和覆载一个普通class的toString方法没有什么区别。   
  
….   
public String toString(){   
switch (this){   
case Red:   
return "Color.Red";   
case Green:   
return "Color.Green";   
case Blue:   
return "Color.Blue";   
default:   
return "Unknow Color";   
}   
}   
….   
  
这时我们可以看到，此时再用前面的遍历代码打印出来的是   
  
Color.Red   
Color.Green   
Color.Blue   
  
而不是   
  
Red   
Green   
Blue.   
  
可以看到toString确实是被覆载了。一般来说在覆载toString的时候我们同时也应该覆载valueOf方法，以保持它们相互的一致性。   
  
4．使用构造函数   
  
虽然enum不可以有public的构造函数，但是我们还是可以定义private的构造函数，在enum内部使用。还是用Color这个例子。   
  
public enum Color {   
Red("This is Red"),   
Green("This is Green"),   
Blue("This is Blue");   
  
private String desc;   
  
Color(String desc){   
this.desc = desc;   
}   
  
public String getDesc(){   
return this.desc;   
}   
  
}   
  
这里我们为每一个颜色提供了一个说明信息, 然后定义了一个构造函数接受这个说明信息。   
  
要注意这里构造函数不能为public或者protected, 从而保证构造函数只能在内部使用，客户代码不能new一个枚举值的实例出来。这也是完全符合情理的，因为我们知道枚举值是public static final的常量而已。   
  
5．实现特定的接口   
  
我们已经知道enum可以定义变量和方法，它要实现一个接口也和普通class实现一个接口一样，这里就不作示例了。   
  
6．定义枚举值自己的方法。   
  
前面我们看到可以为enum定义一些方法，其实我们甚至可以为每一个枚举值定义方法。这样，我们前面覆载 toString的例子可以被改写成这样。   
  
public enum Color {   
Red {   
public String toString(){   
return "Color.Red";   
}   
},   
Green {   
public String toString(){   
return "Color.Green";   
}   
},   
Blue{   
public String toString(){   
return "Color.Blue";   
}   
};   
}   
  
从逻辑上来说这样比原先提供一个“全局“的toString方法要清晰一些。   
  
总的来说，enum作为一个全新定义的类型，是希望能够帮助程序员写出的代码更加简单易懂  
在Java中，枚举类型本质上其实就是一个类，枚举中的常量都是该枚举类型的实例。虽然枚举类型有一些限制，比如不能再派生出子枚举类型，不能调用构造函数，不过我们仍然可以在枚举类型中定义构造函数、字段和方法