Java枚举解析

Java SE5新增了枚举类型，为我们编写程序提供了很大的方便。在jdk 1.5之前，程序员定义类似星期这一类具有枚举特点的数据类型时会采用一些自定义的方式，比如下面的代码：

代码段一：

**final** **class** WeekDay{

**public** **static** **final** **int** ***SUN*** = 0;

**public** **static** **final** **int** ***TUE*** = 1;

**public** **static** **final** **int** ***WED*** = 2;

//others...

}

代码段二：

**final** **class** WeekDay{

**public** **static** **final** WeekDay ***SUN*** = **new** WeekDay();

**public** **static** **final** WeekDay ***TUE*** = **new** WeekDay();

**public** **static** **final** WeekDay ***WEN*** = **new** WeekDay();

// others...

**private** WeekDay(){};

}

上面两段代码能在一定程度上解决问题。第一段代码可以解决纯数字带来的歧义，但是有安全隐患，因为枚举值很容易被改动，比如SUN改为1，TUE改为0，而编译器检测不出这一点。第二段代码看似是一个不错的解决方案，但是最大的缺陷是不能支持switch语句。可见自定义的”枚举“类型并不能完全地解决问题，而且不同的编程人员之间必须达成编码协议。所以jdk1.5支持了枚举类型，使编程人员可以以一致的方式定义枚举类型。下面就总结一下java的枚举类型。

一 枚举的实现方式

用enum关键字定义枚举类型非常简洁：

**enum** WeekDay{

***SUN***, ***TUE***, ***MON***, ***WEN***, ***THU***, ***FRI***, ***SAT***;

}

可以用javap工具查看编译后的字节码，上面的代码基本等价于下面的代码：

**class** WeekDay **extends** Enum<WeekDay>{

**public** **static** **final** WeekDay ***SUN*** = **new** WeekDay("SUN", 0);

**public** **static** **final** WeekDay ***TUE*** = **new** WeekDay("TUE", 1);

**public** **static** **final** WeekDay ***MON*** = **new** WeekDay("MON", 2);

// others...

**private** **static** **final** WeekDay[] ***ENUM$VALUES***;

**static**{

***ENUM$VALUES*** = **new** WeekDay[7];

***ENUM$VALUES***[0] = ***SUN***;

***ENUM$VALUES***[1] = ***TUE***;

***ENUM$VALUES***[2] = ***MON***;

// others...

}

**private** WeekDay(String name, **int** ordinal){

**super**(name, ordinal);

}

**public** **static** WeekDay[] values() {

WeekDay[] values = **new** WeekDay[7];

System.*arraycopy*(***ENUM$VALUES***, 0, values, 0, 7);

**return** values;

}

**public** **static** WeekDay valueOf(String name) {

**return** Enum.*valueOf*(WeekDay.**class**, name);

}

}

上面的代码不能通过编译，这只是编译器对枚举类型编译后的字节码的还原，方便理解。从上面的还源代码可以总结出一下几点：

1. 用enum定义的枚举类型其实是一个普通的类，并且继承自Enum<E>；
2. 编译后的枚举类的构造函数为私有，这样外界就不能随意构造枚举对象；
3. 每一个枚举值其实是一个枚举类的对象，并且声明为静态常量；
4. 枚举类的基类维护两个字段：name和ordinal，前者为枚举值对应名称的字符串，后者为每一个枚举值对应的序号。这两个值在构造每一个枚举值对象的时候传进构造函数；
5. 枚举类型多了两个静态方法values和valueOf，前者以数组的方式返回所有枚举值，后者根据字符串返回对应的枚举值；
6. 当只有一个枚举值的时候，枚举类其实就是一个实现单例模式的类，所以为了方便可以以枚举的方式定义一个单例类（不建议哦）；

二 在枚举中定义其他成员

1. 定义自定义构造函数

枚举可以定义私有构造函数，并在定义枚举值的时候指定调用某个构造函数：

**enum** WeekDay{

***SUN***("Sunday"), ***TUE***, ***MON***, ***WEN***, ***THU***, ***FRI***, ***SAT***;

**private** String fullName;

**private** WeekDay(){}

**private** WeekDay(String fullName){

**this**.fullName = fullName;

}

}

其实编译器会在每个构造函数的参数列表前面加上一个String和int类型的参数，用来传递name和ordinal两个值。

1. 枚举类型可以定义方法

在枚举类型中可以定义任何方法：

**enum** WeekDay{

***SUN***(){

**public** WeekDay nextDay() {

**return** ***TUE***;

}

},

***TUE***(){

**public** WeekDay nextDay() {

**return** MON;

}

}

//others...

;

**public** **abstract** WeekDay nextDay();

}

这种方式虽然很方便很灵活，但是有个缺点就是会产生很多class文件，比如上面的一个类就要产生8个class文件。

三 枚举的优点

1. 语法简洁；
2. 类型安全；
3. 支持switch语句；