



枚举

李玮玮

讲授思路

- 枚举定义
- 基本用法

枚举的引入

- 功能类似C/C++中的枚举.
- 一些方法在运行时，它需要的数据不能是任意的，而必须是一定范围内的值.
- 如：设计一个Student类，要求：学生的grade只能是a，b，c，d，e五个级别。可以给学生设置成绩，并可以输出学生成绩情况（优，良，中，可，差）
 - class Student {
 - private String name;
 - private Grade grade;
 -
 - }

枚举的定义

- 定义
 - 权限修饰符 enum enum_name{
 - 元素1 ,
 - 元素2 ,
 -
 - }

枚举使用举例

- 定义星期的枚举
- 根据日期的不同输出相应的日期信息

```
public enum WeekDay{  
    MONDAY, TUESDAY, WENSDAY, THURSDAY, FRIDAY;  
    //最后这个 “;” 可写可不写。  
}
```

枚举使用举例

```
public void printWeekDay(WeekDay weekDay){  
    switch(weekDay){  
        case MONDAY:  
            System.out.println( "Today is Monday!" );  
            break;  
        case TUESDAY:  
            System.out.println( "Today is Tuesday!" );  
            break;  
        ...  
        default:  
            throw new AssertionError("Unexpected value: " + weekDay);  
    }  
}
```

枚举的本质

- Java中的enum本质就是一个class
 - 如下：枚举类型 Color，编译之后是Color.class

```
public enum Color {  
    RED, GREEN, BLUE, YELLOW;  
}
```

- 上述枚举类型Color也可改写成下面的形式
 - 枚举的成员就是枚举对象，只不过他们是静态常量而已

```
public enum Color {  
    RED(), GREEN(), BLUE(), YELLOW();  
}
```

枚举的本质


- 枚举可以添加构造方法

```
public class Color {  
    private static final Color RED = new Color();  
    private static final Color GREEN = new Color();  
    private static final Color BLUE = new Color();  
    private static final Color YELLOW = new Color();  
}
```


```
    Color(String name, int id) {  
        _name = name;  
        _id = id;  
    }  
    String _name;  
    int _id;
```


枚举的本质

```
public enum Color {  
    RED("red color", 0), GREEN("green color", 1),  
    BLUE("blue color", 2), YELLOW("yellow color", 3);  
    Color(String name, int id) {  
        _name = name;  
        _id = id;  
    }  
    String _name;  
    int _id;  
}
```



```
public enum Color {  
    RED("red color", 0), GREEN("green color", 1),  
    BLUE("blue color", 2), YELLOW("yellow color", 3);  
}
```



枚举与类的区别

- 枚举不可以实例化
 - 编译器会自动为其构造方法加上了 `private`

```
Color(String name, int id) {  
    _name = name;  
    _id = id;  
}  
String _name;  
int _id;
```

- 通常只为枚举成员变量提供get方法，而不提供set方法

枚举与类的区别

```
public enum Color {  
    RED("red color", 0), GREEN("green color", 1),  
    BLUE("blue color", 2), YELLOW("yellow color", 3);  
    Color(String name, int id) {  
        _name = name;  
        _id = id;  
    }  
    private String _name;  
    private int _id;  
    public String getName() {  
        return _name;  
    }  
    public int getId() {  
        return _id;  
    }  
}
```

```
public void setName(String name) {  
    _name = name;  
}  
public void setId(int id) {  
    _id = id;  
}
```

不推荐

?

枚举的特点

- 数据集
 - 他们的数值在程序中是稳定的
 - 元素个数有限
- 所有枚举类都继承了 Enum 类的方法
 - toString
 - Equals
 - Hashcode
 - ...
 - 注：equals、hashcode 方法是 final 的，所以不可以被枚举重写（只可以继承）。但是，可以重写 toString 方法
 - Java 不允许使用 = 为枚举常量赋值

枚举成员

- 枚举成员也是变量，变量名当然不能以数字开头的
- Java 不允许使用 = 为枚举常量赋值

```
public enum Num{  
    1, 2, 3 ;  
}
```



```
enum Grade{  
    A = 1, B = 2, C = 3;  
}
```



枚举补充

- 枚举值默认为从0开始的有序数值
- 枚举的典型应用场景：错误码、状态机等
- Enum不能继承类，它是继承自`java.lang.Enum`的特殊的类
- enum可以实现接口
- 在enum中，提供了
 - `values()`：返回enum中声明时的所有实例。
 - `name()`：返回实例的字符串名称。
 - `ordinal()`：返回实例声明时的顺序，从0开始。
 - `getDeclaringClass()`：返回实例所在的enum类型。
 - `equals()`：判断是否为同一个对象。
 - 可以使用 `==` 来比较enum实例。

除了不能继承，基本上可以将 enum 看做一个常规的类

枚举补充

```
public class EnumMethodDemo {  
    enum Color {RED, GREEN, BLUE;}  
    enum Size {BIG, MIDDLE, SMALL;}  
    public static void main(String args[]) {  
        System.out.println("=== Print all Color ===");  
        for (Color c : Color.values()) {  
            System.out.println(c + " ordinal: " +c.ordinal());  
        }  
        System.out.println("===Print all Size ===");  
        for (Size s : Size.values()) {  
            System.out.println(s + " ordinal: " +s.ordinal());  
        }  
        Color green = Color.GREEN;  
    }  
}
```


枚举补充

```
System.out.println("green name(): " + green.name());
System.out.println("green getDeclaringClass(): "
    + green.getDeclaringClass());
System.out.println("green hashCode(): "
    + green.hashCode());
System.out.println("green compareTo Color.GREEN: "
    + green.compareTo(Color.GREEN));
System.out.println("green equals Color.GREEN: "
    + green.equals(Color.GREEN));
System.out.println("green equals Size.MIDDLE: "
    + green.equals(Size.MIDDLE));
System.out.println("green equals 1: " + green.equals(1));
System.out.format("green == Color.BLUE: %b\n"
    , green == Color.BLUE); } }
```


总结

- 枚举的使用方法



Thank You