日期: /

枚举类

1、使用场景

① 类的对象只有有限个，确定的，此类为枚举类

② 定义一组常量时，推荐使用

③ 若枚举类中只有一个对象，可作为单例模式的实现方式

2、定义方法

(1) 自定义 JDK5之前

```
}
//自定义枚举类
class Season{
    //1.声明Season对象的属性:private final修饰
    private final String seasonName;
    private final String seasonDesc;

    //2.私有化类的构造器,并给对象属性赋值
    private Season(String seasonName,String seasonDesc){
        this.seasonName = seasonName;
        this.seasonDesc = seasonDesc;
    }

    //3.提供当前枚举类的多个对象：public static final的
    public static final Season SPRING = new Season( seasonName: "春天", seasonDesc:
    public static final Season SUMMER = new Season( seasonName: "夏天", seasonDesc:
    public static final Season AUTUMN = new Season( seasonName: "秋天", seasonDesc:
    public static final Season WINTER = new Season( seasonName: "冬天", seasonDesc:

    //4.其他诉求：获取枚举类对象的属性
```

(2) enum关键字

```
//使用enum关键字枚举类
enum Season1{
    //1.提供当前枚举类的对象，多个对象之间用","隔开，末尾对象";"结束
    SPRING("春天","春暖花开"),
    SUMMER("夏天","夏日炎炎"),
    AUTUMN("秋天","秋高气爽"),
    WINTER("冬天","冰天雪地");
```

继承于java.lang.Enum.

日期: /

(3) Enum类的常用方法

① toString() 输出对象名

② values() 返回对象数组

③ valueOf(String objName) 返回对应的对象

(4) 枚举类实现接口

1、正常实现

2、若想每个对象有不同的实现方法，可写成

```
SPRING("春天","春暖花开"){
    @Override
    public void show() {
        System.out.println("春天在哪里？");
    }
},
```

注解 (Annotation)

框架 = 注解 + 反射 + 设计模式

1、使用示例

① 生成文档相关

② 编译时格式检查

@Override @Deprecated @SuppressWarnings：抑制编译器警告

③ 跟踪代码依赖性，替代配置文件

2、自定义注解

配上注解的信息处理流程(反射)才有意义

```
3. 如何自定义注解：参照@SuppressWarnings定义
 * ① 注解声明为：@interface
 * ② 内部定义成员，通常使用value表示
 * ③ 可以指定成员的默认值，使用default定义
 * ④ 如果自定义注解没有成员，表明是一个标识作用。

如果注解有成员，在使用注解时，需要指明成员的值。
```

日期: /

3、元注解

(1) 对现有的注解进行注解

(2) 4种

用在定注解种

① Retention：指定Annotation的生命周期 反射用SOURCE

② Target ：指定Annotation能修饰什么结构

③ Documented：将指定的Annotation提取到Javadoc中

④ Inherited：被修饰的An'具有继承性

4、JDK8新特性：可重复注解、类型注解

```
6. jdk 8 中注解的新特性：可重复注解、类型注解

6.1 可重复注解：① 在MyAnnotation上声明@Repeatable，成员值为MyAnnotations.class
                ② MyAnnotation的Target和Retention和MyAnnotations相同。

6.2 类型注解：
ElementType.TYPE_PARAMETER 表示该注解能写在类型变量的声明语句中（如：泛型声明）。
ElementType.TYPE_USE 表示该注解能写在使用类型的任何语句中。
```

等

```
class Generic<@MyAnnotation T>{

    public void show() throws @MyAnnotation RuntimeException{

        ArrayList<@MyAnnotation String> list = new ArrayList<>();

        int num = (@MyAnnotation int) 10L;
    }

}
```

日期: /

日期: /