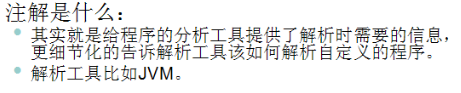
注解Annotation

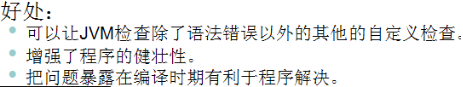
1. 注解Annotation：是给编译器看的，而注释是让程序员看的。
2. 注解是什么？



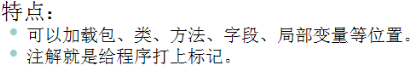
1. 有可能会出现错误，但是不让虚拟机报错，可以在开端处添加：

**@SuppressWarnings("unchecked")**

1. 注解功能的好处：将可能出现的错误提前到了编译时期。如果没有注解功能，则只有到了**运行时**才能出现错误，但是利用了**注解功能**，在**编译时期**就会出现提示。

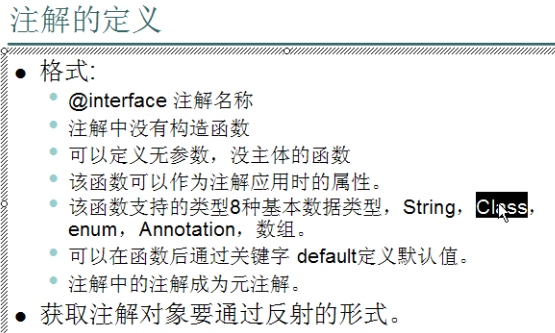


1. 注解特点：



1. 注解：（重要）

数据的数据称为是元数据，因此注解的注解称为是**元注解**。



注解中函数支持的数据类型包括：

**8中原生数据类型、String、Class、Annotation、Enum、数组**。

**示例：**

定义一个AnnotationDemo：

**package** com.java.reflect;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(value=ElementType.***METHOD***)

@Retention(value=RetentionPolicy.***RUNTIME***)

**public** **@interface** AnnotationDemo {

**int** num() **default** 90;//八种原生数据类型都支持，以int为例即可。

String show()**default** "xiaohong";//支持String类型数据

**int**[] ints()**default** {23,32,3423};//支持数组

// Dog haveDogs() default new Dog("wangzai",3);//不支持普通的一个类

Class haveDogs() **default** Dog.**class**;//支持Class

AnnotationINAnnotation getAnnotation() **default** @AnnotationINAnnotation(value="注解的注解");

EnumDemo getEnum();

}

调用注解中的函数：

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

Class clazz = Class.*forName*("com.java.reflect.Person");

Method method = clazz.getMethod("show3");

AnnotationDemo annotationdemo = method.getAnnotation(AnnotationDemo.**class**);

String str = annotationdemo.show();

System.***out***.println(str);

**int** num = annotationdemo.num();

System.***out***.println(num);

**int**[] aas = annotationdemo.ints();

System.***out***.println(Arrays.*toString*(aas));

Class clazzDog = annotationdemo.haveDogs();

System.***out***.println(clazzDog.getName());

AnnotationINAnnotation aia = annotationdemo.getAnnotation();

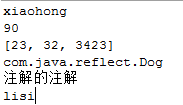
String str2 = aia.value();

System.***out***.println(str2);

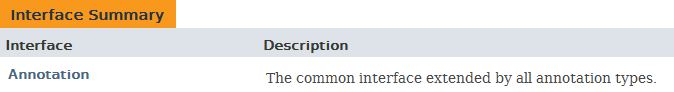
EnumDemo ed = annotationdemo.getEnum();

System.***out***.println(ed);

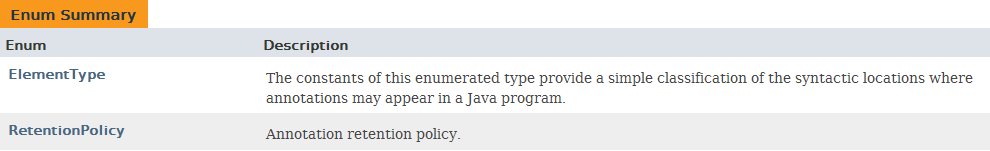
}输出结果：



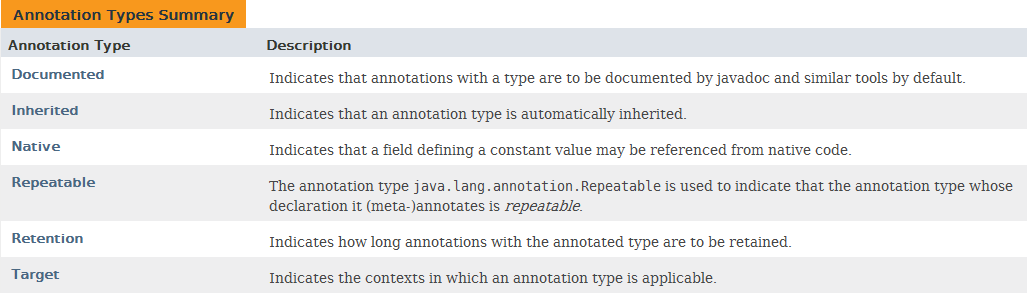
1. java.lang.annotation包：
2. 接口：



1. 枚举：



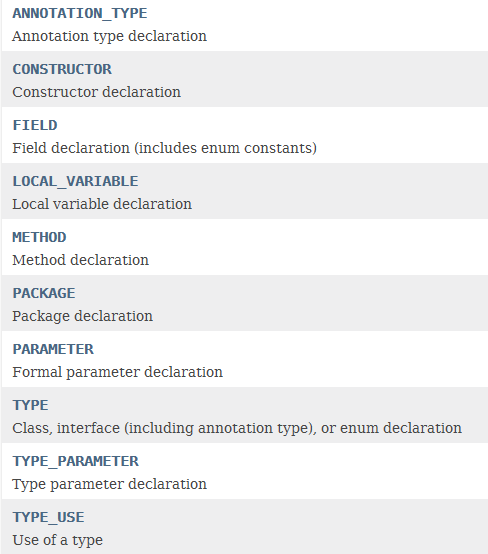
1. 注解类型：



1. 异常Exception和错误Error
2. 两个重要的枚举：
3. **ElementType**：



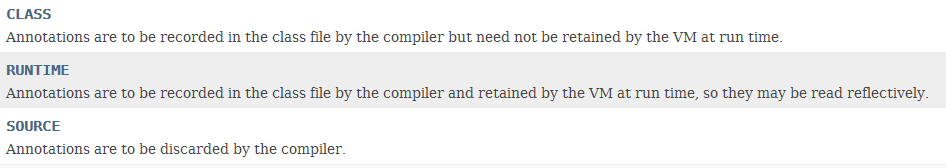
决定了此注解类型应该出现的位置。



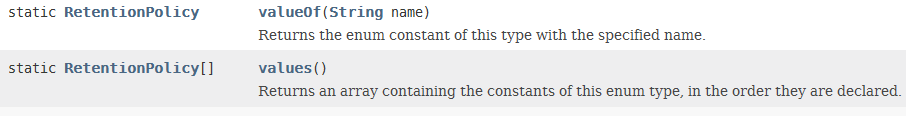
主要有FIELD、METHOD、PACKAGE、TYPE、CONSTRUCTOR、PARAMETER等.

1. RetentionPolicy：

主要是三个：CLASS、RUNTIME、SOURCE



主要是两个方法：



1. Override注解类型的源代码：

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.SOURCE)

public @interface Override {

}

1. 注解类型Retention：

只有一个元素：Element：

value



RetentionPolicy是一个枚举类型的，有三个值CLASS、RUNTIME、SOURCE、。

故在定义注解类型时，不用写value直接写上RetentionPolicy.XXX即可。如：@Retention(RetentionPolicy.SOURCE);等价于

@Retention(value=RetentionPolicy.SOURCE);

1. 获取注解对象：

获取注解对象只能是通过反射机制获取，：

1. 获取类上的注解对象：直接利用类的字节码文件对象class的getAnnotation方法；
2. 获取方法上的注解对象：首先获取此方法对象Method，然后利用 Method的getAnnotation方法获取注解对象。