# 注解

注释：用文字描述程序的。给程序员看的

注解概念：说明程序的，给计算机看的

注解定义：注解（Annotation）,也叫元数据。一种代码级别说明。他是JDK1.5及以后版本引入的一个特性，与类、接口、枚举是在同一个程层次。他可以声明在包、类、字段、方法、局部变量、方法参数等前面，用来对这些元素进行说明，注释。

概念描述：

1. JDK1.5之后的新特性
2. 说明程序的
3. 使用注解：@注解名

作用分类：

1. 编写文档：通过代码里标识的注解生成文档（生成文档doc文档）
2. 代码分析：通过代码里标识的注解对代码进行分析（使用反射）
3. 编译检查：通过代码里标识的注解让编译器能够实现基本的编译检查（Override）

学习要点：

1. JDK中的预定义的一些注解
   1. @override：检测被该注解标注的方法是否继承自（接口）的
   2. @Deprecated:该注解标注的内容，表示已过时
   3. @SuppressWarnings(传递参数):压制警告
      1. 一般传递参数all
2. 自定义注解
   1. 格式：
      1. 元注解
      2. Public @interface 注解名称{
         * 1. 属性列表
           2. }
   2. 本质：注解本质上就是一个接口，该接口默认继承Annotation接口
      1. Public interface 接口名 extends java.lang.ammotation.Annotation{}
   3. 属性：接口中的抽象方法（也叫属性）
      1. 属性要求：
         1. 属性的返回值类型有下列取值（其他不可以）
            1. 基本数据类型
            2. String
            3. 枚举
            4. 其他注解
            5. 以上类型的数组
         2. 定义了属性，在使用时需要给属性赋值
            1. 如果定义属性时，使用default关键字给属性初始化值，则使用注解时，可以不进行属性的赋值。
            2. 如果只有一个属性需要赋值，并且属性的名称是value，则value可以省略，直接定义值即可。
            3. 数组赋值时，值使用{}包裹。如果数组只有一个值，则{}可以省
         3. 元注解：用于描述注解的注解
            1. @Target:描述注解能够作用的位置

ElementType取值：

TYPT：可以作用于类上

METHOD：可以作用于方法上

FIELD：可以作用于成员变量上

1. @R etention:描述注解被保留的阶段
   1. @Retention(RetentionPolicy.RUNTIME):当前被描述的注解，会保留到class字节码文件中，并被JVM读取到
2. @Documented:描述注解是否被抽取到API文档中
3. @Inherited：描述注解是否被子类继承
4. 在程序使用（解析）注解：获取注解中定义的属性值
   1. 获取注解定义的位置的对象（Class，Method，Field）
   2. 获取指定的注解
      1. getAnnotation（class）其实就是在内存中生成了一个该注解的接口的子类的实现对象
5. 调用注解中的抽象方法配置的属性值

小结：

以后大多数时候，我们会使用注解，而不是自定义注解

注解给谁用？

编译器

给解析程序看

注解不是程序的一部分，可以理解为注解就是一个标签