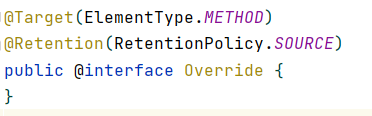
**一，注解的作用**

**对程序作出解释，告诉编译器一些信息**

**二，注解的本质**

**注解的本质接口，默认的继承自 Annotation接口**

**三，注解的定义**



**@Target*({TYPE*, *FIELD*, *METHOD*, *PARAMETER*, *CONSTRUCTOR*, *LOCAL\_VARIABLE})*@Retention*(*RetentionPolicy.*SOURCE)*public @interface SuppressWarnings *{***

**String*[]* value*()*;  
*}***

**public @interface 注解名{**

**定义属性**

**}**

属性成员只能用基本类型**byte，short，char，int，long，float，double，boolean**八种基本数据类型和**String**、**Enum**、**Class**、**annotations**等数据类型，以及这一些类型的**数组。**

**①注解的定义可以添加元注解**

**例如 元注解@Target，就是告诉编译器，这个自定义的注解可以用在哪里**

**②注解中可以定义属性，就是这里的属性名后面要跟上( )**

**例如 String[]value()；定义了一个String数组value**

**另外 定义属性还可以给属性添加默认值**

**例如 int age（） default 10;在注解里面定义了int属性age，如果使用注解时，没有给age赋值，那么age默认为10；**

**四，注解的使用**

**！！！使用注解时——保证注解中的每个属性都是有值得**

**1,注解无属性——@注解名**

**2，注解有属性——@注解名(属性名=值，属性名=值……)**

**对于非数组属性——属性名=属性值**

**对于数组属性——①属性名=属性值（数组只有一个元素）**

**②属性名={属性值1，属性值2……}(数组有多个元素)**

**！！没有设置默认值的属性，在使用注解时一定要赋值**

**！！设置了默认值的属性，在使用注解时可以不赋值，那么将默认采用默认值**

**五，内置注解——Java自带的注解**

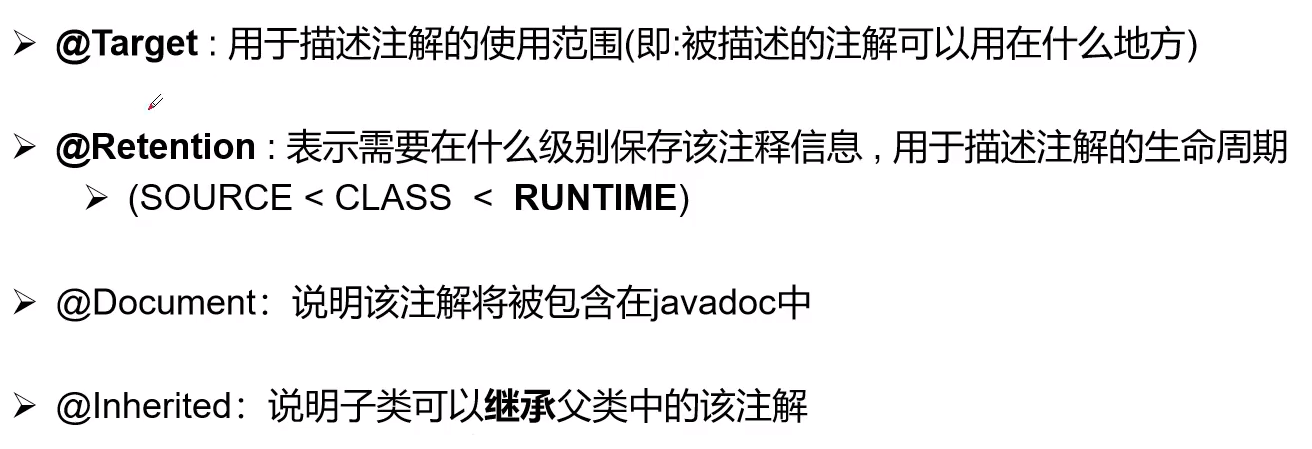
@Override

@SuppressWarnings

@Deprecated

**六，元注解**

可以用来注解其他注解的注解——定义一个注解时，可以添加元注解



@Retention

SOURCE——注解仅在源码(.java文件中)的时有效，编译时丢弃

CLASS——注解在源码，编译为.class文件时有效，被JVM丢弃，不会被加载

RUNTIME——注解在运行时仍有效，可以被JVM加载，可以通过反射获取

**七，通过反射获取注解**

**只有RUNTIME的注解才可以被加载到JVM中**

**注解载入JVM内存后，会形成一个Annotation对象**

**Annotion对象可以存在于Class对象，Field对象，Method对象，Constructor对象中**

**1，Annotation提供的方法**

**①String annotationType( )；返回 注解的定义语句**

**②String toString(); 返回 注解的使用语句**



**Class对象中提供的获取注解方法**

**①Annotation[] getAnnotations()**

返回注解对象数组，但只含有注解这个类的注解

**②A getAnnotation(Class <A> annotationClass)**

根据注解的Class对象，返回注解对象（只能返回注解这个类的注解）

**Method对象，Constructor对象，Field对象也提供了上述的方法**

**同样的Method对象只能获得 注解了这个方法的注解**

**Field对象只能获得 注解了这个属性的注解**

**Constructor对象只能获得 注解了这个构造方法的注解**