# 注解

注解(Annotation)是 Java1.5 中引入的一个重大修改之一，为我们在代码中添加信息提供了一种形式化的方法，使我们可以在稍后某个时刻非常方便的使用这些数据。注解在一定程度上是把元数据与源代码结合在一起，而不是保存在外部文档中。注解的含义可以理解为 java 中的元数据。元数据是描述数据的数据。

注解是一个继承自java.lang.annotation.Annotation的接口

## 可见性

根据注解在程序不同时期的可见性，可以把注解区分为： - source：注解会在编译期间被丢弃，不会编译到 class 文件 - class：注解会被编译到 class 文件中，但是在运行时不能获取 - runtime：注解会被编译到 class 文件中，并且能够在运行时通过反射获取

## 继承

|  |  |  |
| --- | --- | --- |
|  | 有@Inherited | 没有@Inherited |
| 子类的类上能否继承到父类的类上的注解？ | 否 | 能 |
| 子类实现了父类上的抽象方法 | 否 | 否 |
| 子类继承了父类上的方法 | 能 | 能 |
| 子类覆盖了父类上的方法 | 否 | 否 |

@Inherited 只是可控制对类名上注解是否可以被继承。不能控制方法上的注解是否可以被继承。

## 注解的实现机制

1. 注解是继承自：java.lang.annotation.Annotation 的接口

...
 Compiled from "TestAnnotation.java"
public interface TestAnnotation extends java.lang.annotation.Annotation
...

1. 注解内部的属性是在编译期间确定的

...
SourceFile: "SimpleTest.java"
RuntimeVisibleAnnotations:
 0: #43(#44=s#45)
...

1. 注解在运行时会生成 Proxy 代理类，并使用 AnnotationInvocationHandler.memberValues 来进行数据读取

...
default:
 //从 Map 中获取数据
 Object var6 = this.memberValues.get(var4);
 if (var6 == null) {
 throw new IncompleteAnnotationException(this.type, var4);
 } else if (var6 instanceof ExceptionProxy) {
 throw ((ExceptionProxy)var6).generateException();
 } else {
 if (var6.getClass().isArray() && Array.getLength(var6) != 0) {
 var6 = this.cloneArray(var6);
 }

 return var6;
 }
}
...

## 参考链接

* [java注解是怎么实现的？](https://www.zhihu.com/question/24401191)