注解

一、元数据：

1、定义

元数据是关于数据的数据。在编程语言上下文中，元数据是添加到程序元素如方法、字段、类和包上的额外信息。对数据进行说明描述的数据。

2、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】我们自己不能改的，比如@author

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】我们自己不能改的,比如@override

二、注解（本质：接口）

1、定义：

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

2、概念描述

- JDK1.5之后的新特性
- 说明程序的
- 使用注解：@注解名称

3、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】我们自己不能改的，比如@author

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】我们自己不能改的,比如@override

4、分类

1）jdk自带的标准注解

- @Override：检测被该注解标注的方法是否是继承自父类(接口)的
- @Deprecated：该注解标注的内容，表示已过时 (此时不实现该方法或者类不会报错)
- @SuppressWarnings：压制警告
  - 一般传递参数all

@SuppressWarnings(“all”)

2）第三方的注解 像Spring的

3）自定义注解

可以根据自己的需求定义注解

三、解析注解

获取注解中定义的属性值 替换反射中使用的配置文件，配置文件中指定类名和方法名

步骤

1. 获取注解定义的位置的对象 （Class，Method,Field）
2. 获取指定的注解 getAnnotation(Class)：其实就是在内存中生成了一个该注解接口的子类实现对象

```
public class ProImpl implements Pro{
    public String className(){
        return "com.scu.annotation.User";
    }
    public String methodName(){
        return "show";
    }
}
```

3、调用注解中的抽象方法获取配置的属性值

```
public class User {
    public void show(){
        System.out.println("hello world!");
    }
}

@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Prop {
    String className();
    String methodName();
}


/**
 * 框架类
 * 前提：不能改变该类的任何代码。可以创建任意类的对象，可以执行任意方法
 */
@Prop(className = "com.scu.annotation.User",methodName = "show")
public class PropTest {
    public static void main(String[] args) throws Exception {
        //1.解析注解
        //1.1获取该类的字节码文件对象
        Class<PropTest> clazz = PropTest.class;
        //2.获取上边的注解对象  //其实就是在内存中生成了一个该注解接口的子类实现对象
        Prop annotation = clazz.getAnnotation(Prop.class);
        //3.调用注解对象中定义的抽象方法，获取返回值
        String className = annotation.className();
        String methodName = annotation.methodName();
        //3.加载该类进内存
        Class cls = Class.forName(className);
        //4.创建对象
        Object instance = cls.newInstance();
        //5.获取方法对象
        Method method = cls.getMethod(methodName);
        //6.执行方法
        method.invoke(instance);

    }
}

//运行后输出 "hello world!"
```

四、小结

1. 以后大多数时候，我们会使用注解，而不是自定义注解
2. 注解给谁用?
    a. 编译器
    b. 给解析程序用 比如上面的CheckTest
3. 注解不是程序的一部分，可以理解为注解就是一个标签 还有就是属性不能带参数