kdk内置的基本注解类型（3个）

自定义注解类型

对注解进行注解

利用反射获取注解信息（在反射部分涉及）

1，从jdk5.0开始，java增加了对元数据metadata的支持，也就是annotation（注解）

2，annotation其实就是代码里的特殊标记，这些标记可以在编译，类加载，运行时被读取，并执行相应的处理，通过使用annotation。程序员可以在不改变原有逻辑的情况下，在源文件中嵌入一些补充信息

3，annotation可以像修饰符一样被使用，可用于修饰包，构造器，方法，成员变量，参数，局部变量的声明，这些信息被保存在annotation的“name=value”对中

4，annotation能被用来为程序元素设置元数据

一，注解：

使用annotation时要在其前面增加@符号，并把该annotation当成一个修饰符使用，用于修饰它支持的程序元素

三个基本的annotation：①@override：限定重写父类方法，该注释只能用于方法

②@deprecated：用于表示某个程序元素（类，方法等）已过时

③@suppresswarning：抑制编译器警告（集合未使用泛型以及定义了以后变量没有使用）

二，自定义一个注解：

```
public @interface may {
String valueString()default "hello";
}
```

```
@may(valueString="anqili")
public student(int id, String nameString) {
```

三，元注解：

jdk的元annotation用于修饰其他的annotation定义

jdk5.0提供了专门在注解上的注解类型，分别是：retention，target，documented，inherited

①retention：只能用于修饰一个annotation定义，用于指定该annotation可以保留多长时间

@retention包含一个retentionpolicy类型的成员变量，使用@retention时必须为该value成员变量指

定值： ①retentionPolicy.source：编译器直接丢弃这种策略的注释

②retentionPolicy.class：编译器将把注释记录在class文件中，当运行java程序的时候，

jvm不会保留注释，这是默认值

③retentionPolicy.runtime：编译器将把注释记录在从class文件中，当运行java程序的时

候，jvm会保留注释，可以通过反射机制获取注解的信息

②target：用于修饰annotation定义，用于指定被修饰的annotation能用于修饰哪些程序元素，

@target也包含一个名为value的成员变量

③@documented：用于指定被该元素annotation修饰的annotation类将被javadoc工具提取成为文档

（定义为documented的注解必须设置retention值为runtime）

④@inherited：被它修饰的annotation将具有继承性，如果某个类使用了被@inherited修饰的

annotation，则其子类将自动具有该注解