

# 反射之前



# 反射之后





疑问：反射机制与面向对象中的封装机制（封装机制的目的：private是内部用，告诉调用者你用不着，你用public就可以了）是不是有矛盾？（不矛盾）如何看待这两种技术。

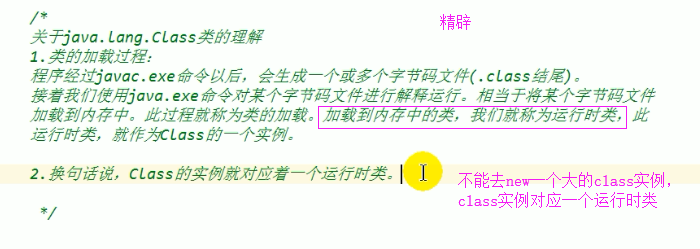
疑问：通过直接new的方式或反射的方式都可以直接调用公共的结构，开发中到底用那个？

建议用new的方式，什么时候用反射：反射的动态性，反射的特征：动态性（比如一个网站部署了，/login就会跳到login界面（服务器运行起来，看到你传过来的是login，就会造login的对象）。

# Class类

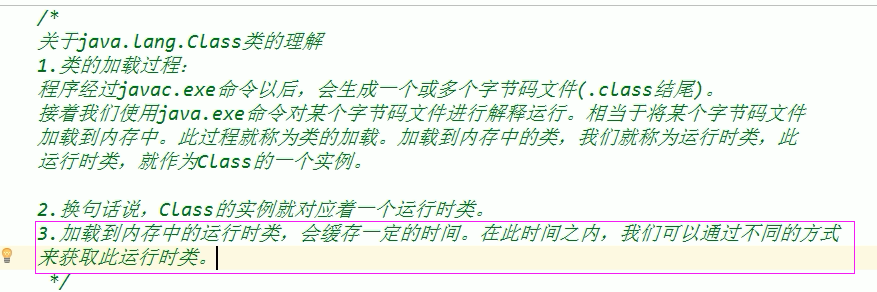
## 谈谈你对Class类的理解：

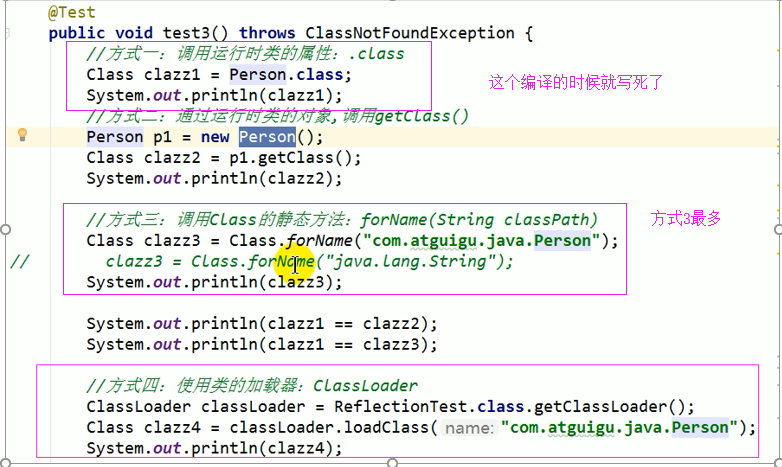
Class实例对应着加载到内存中的一个运行时类。



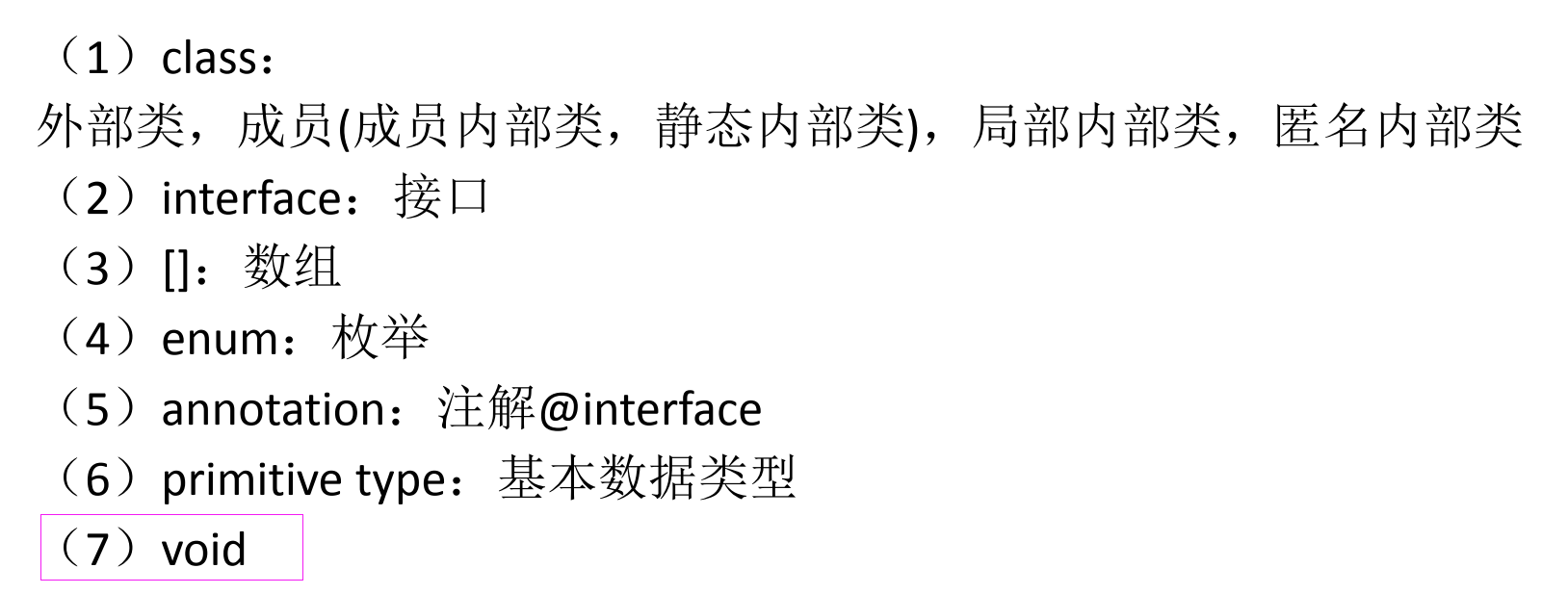
## 获取Class类实例的四种方法



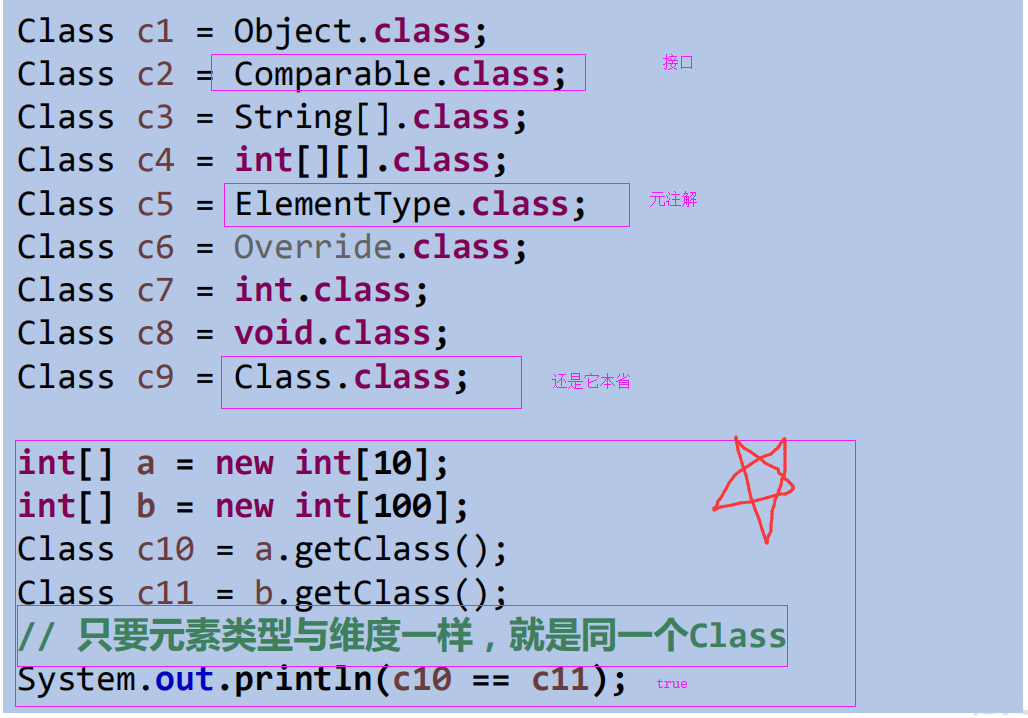




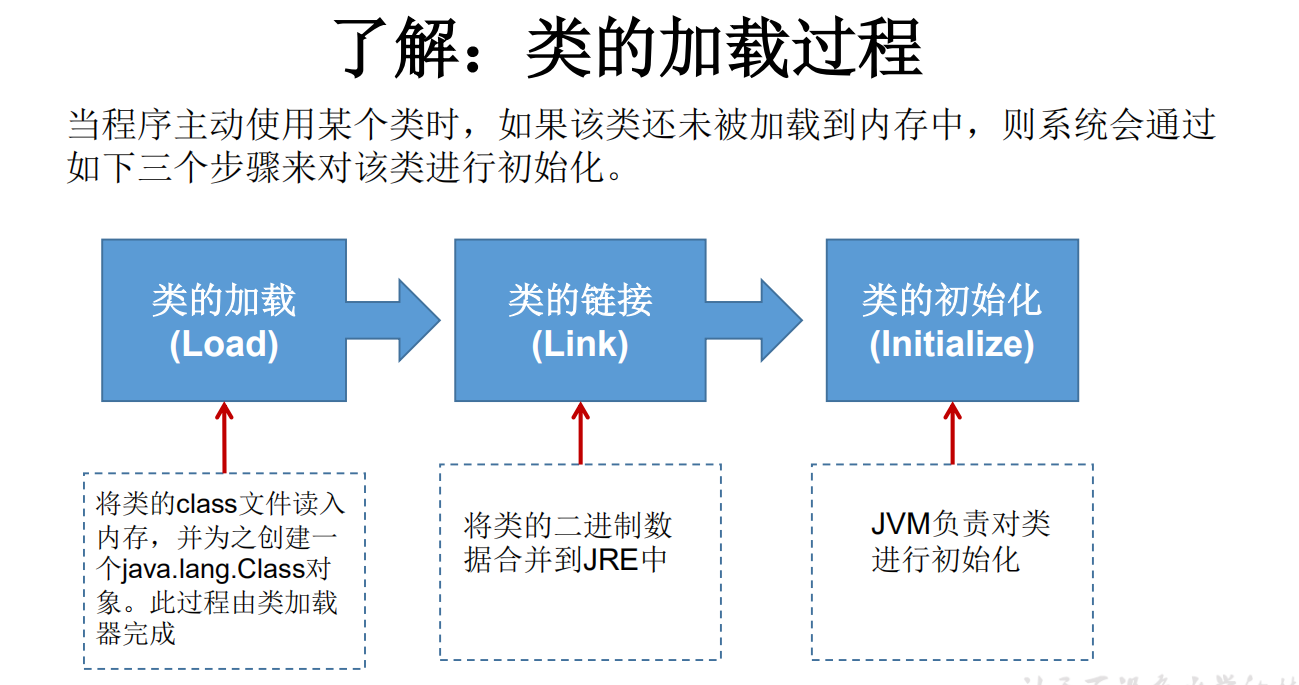
# 哪些类型可以有Class对象？

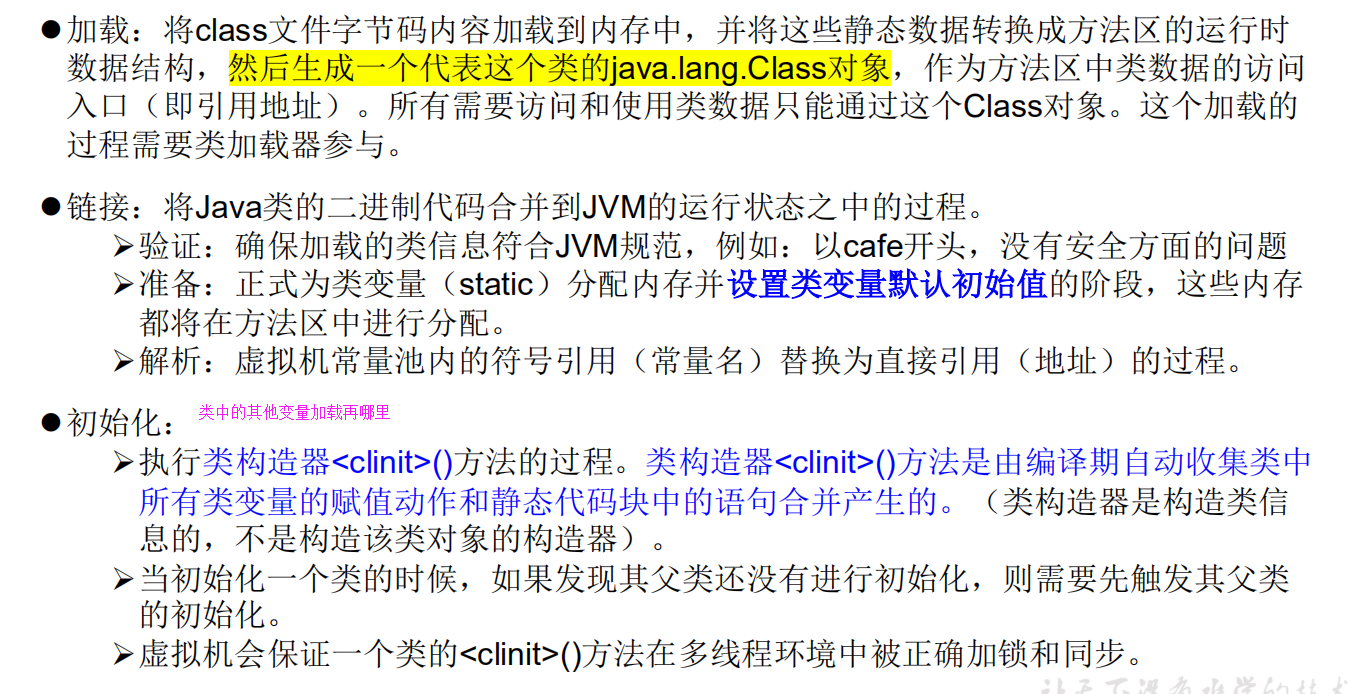




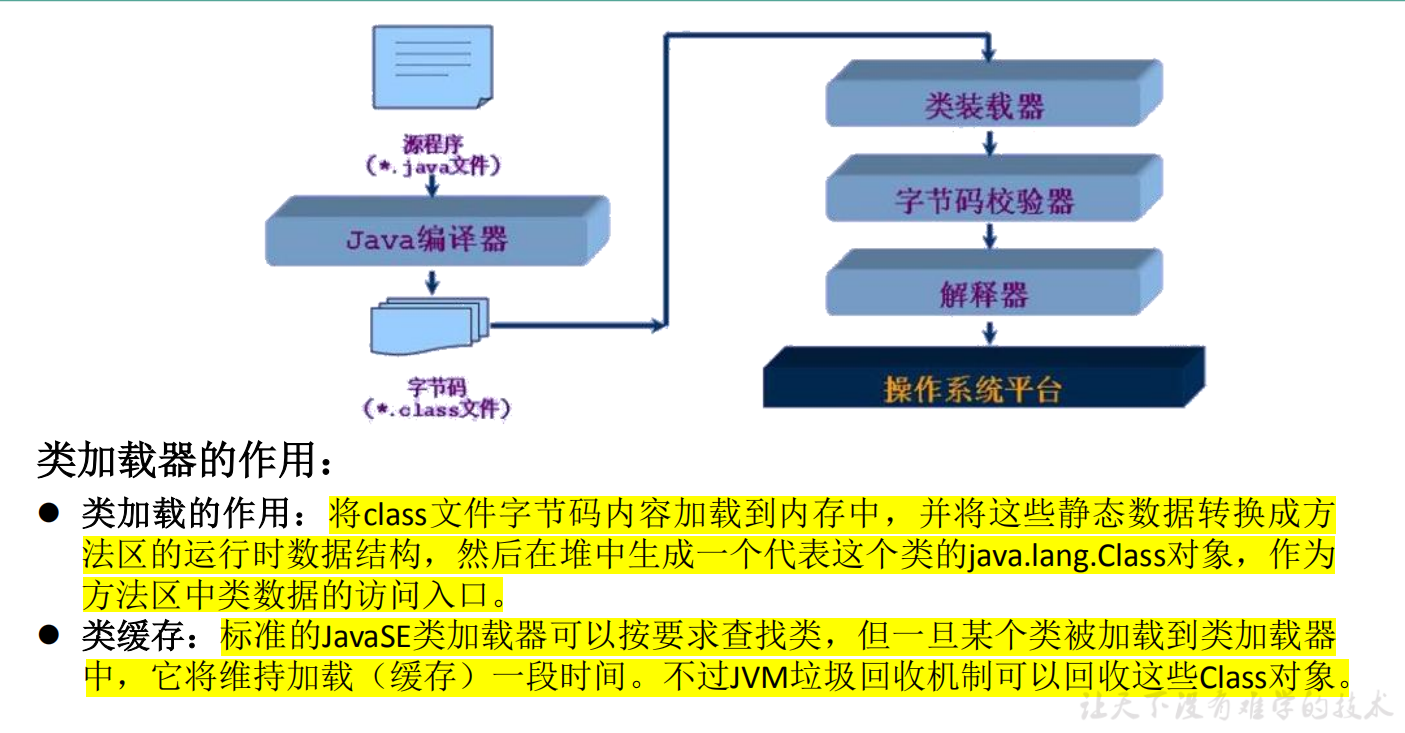


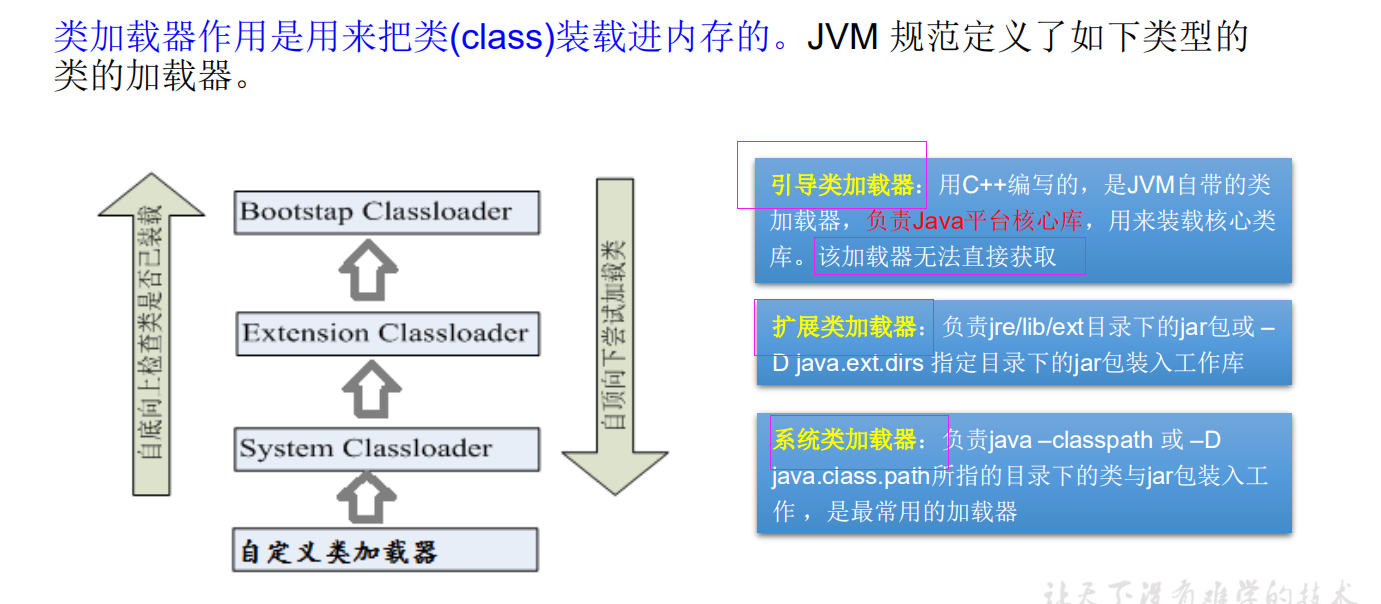
# 类的加载过程

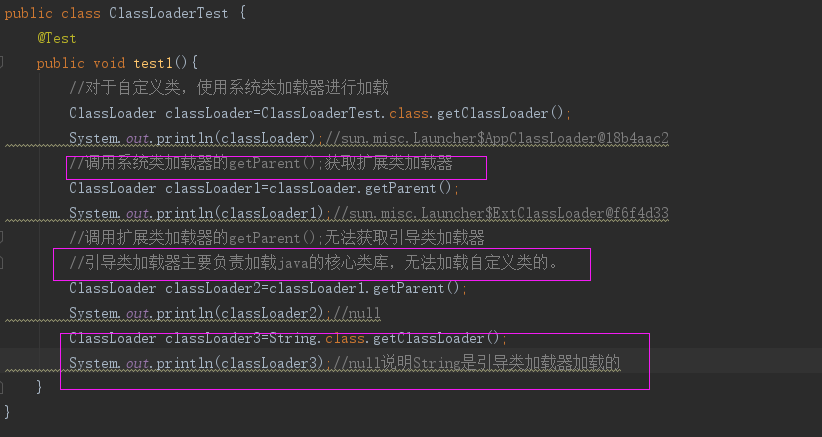




# 类加载器

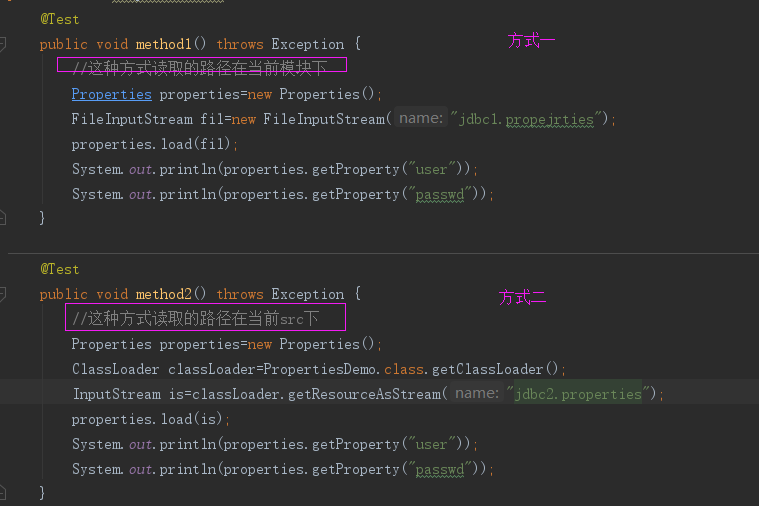




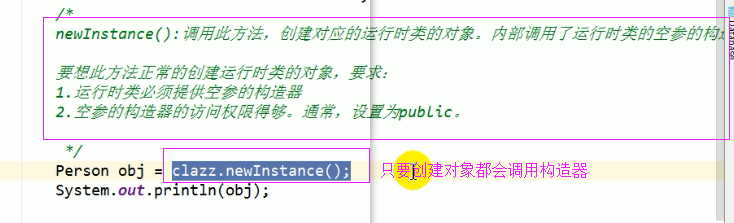


# Properties(用来读取配置文件)

当前模块指的是项目名



# 通过反射创建运行时类对象



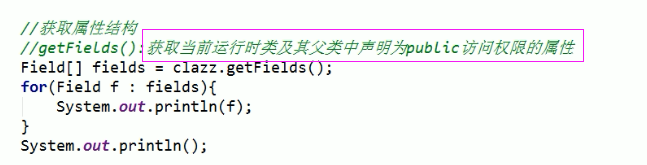
所以直接跳过获得构造器的语句来newInstance的话，需要有空的构造器。推荐，使用反射创建对象的时候，一般都是跳过获得构造器，直接用newInstacne创建对象，然后使用调用属性来给属性赋值。（原因时确保通用，因为每个类的含有参数的构造器的参数个数不一致）

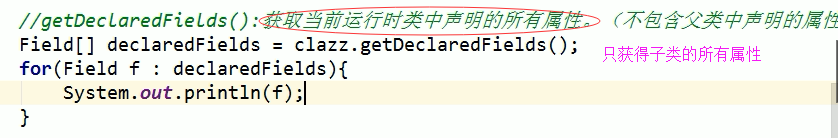
# \*\*\*反射的动态性



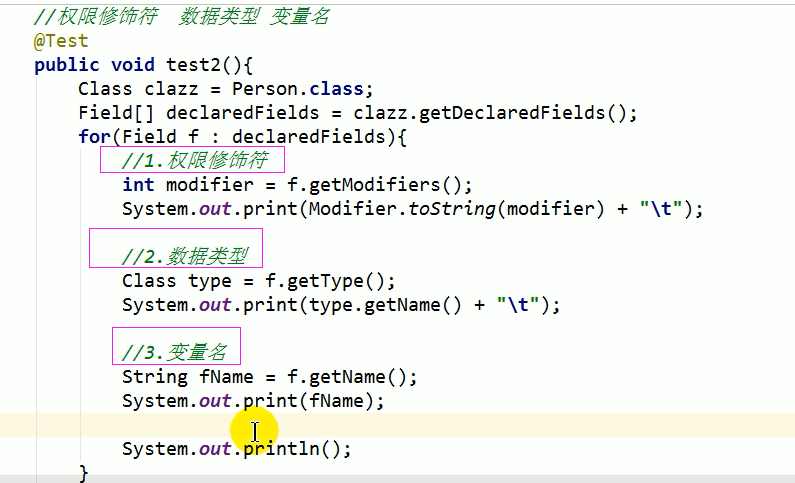
# 通过反射获取运行时类的完成结构

## 属性

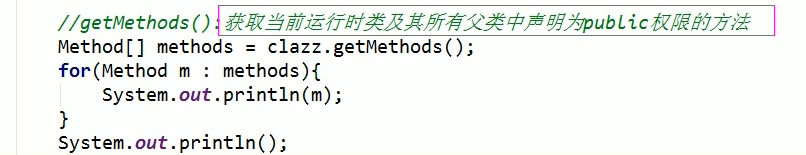




## 属性具体的结构

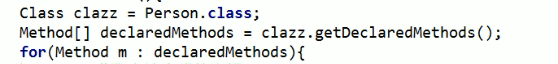


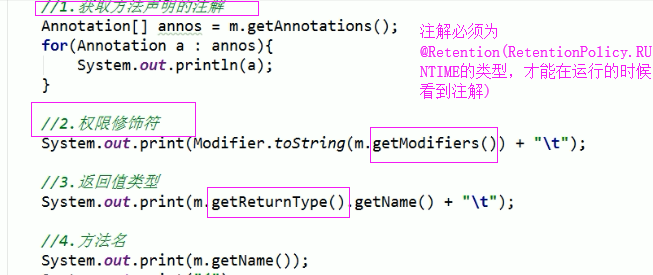
## 方法

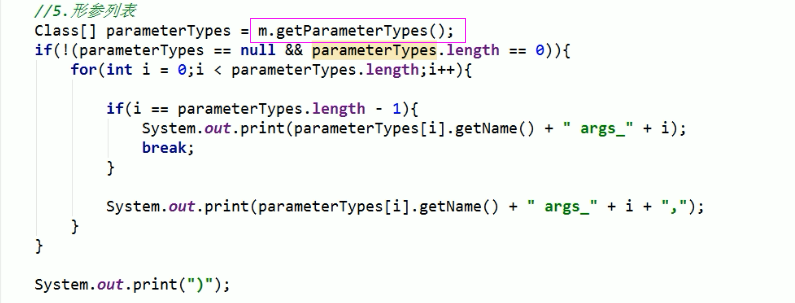


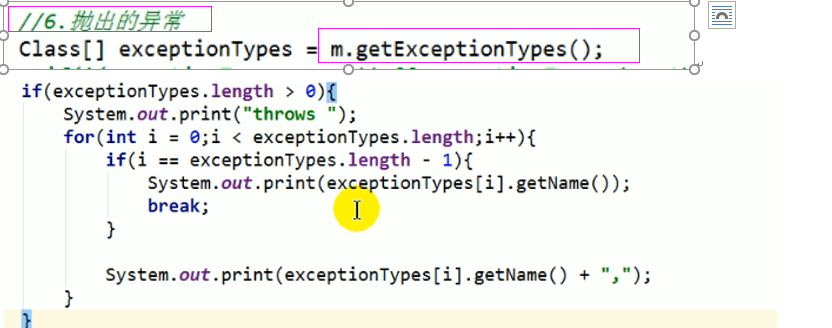


## 方法的具体结构



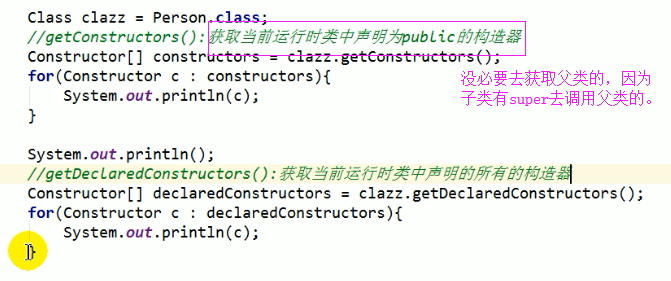




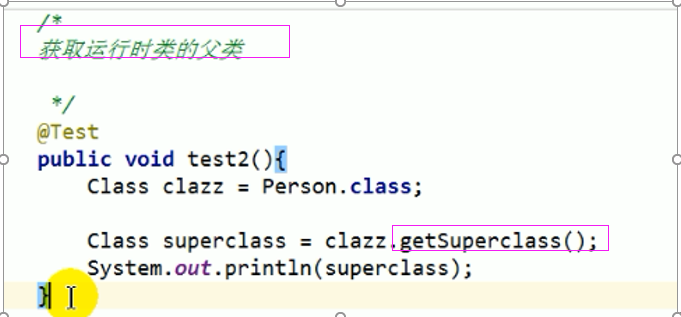


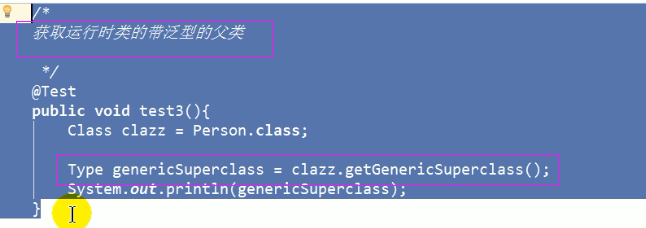
## \*\*\*\*框架=注解+反射+设计模式

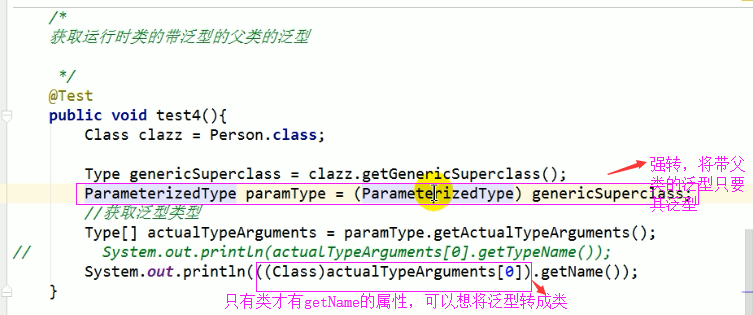
## 获取运行时类的构造器结构



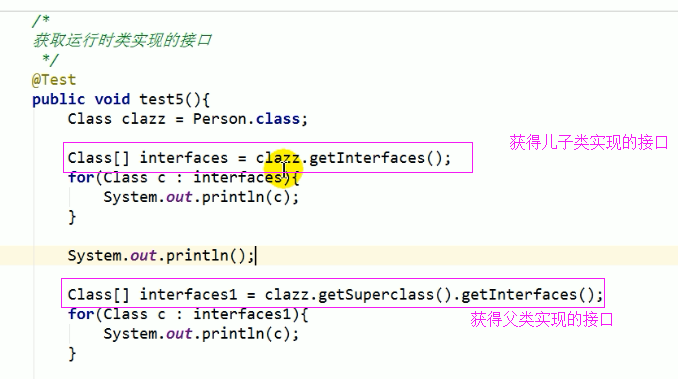
## 获取运行时类的父类（或者是运行时类带父类的泛型（泛型和父类一起输出））



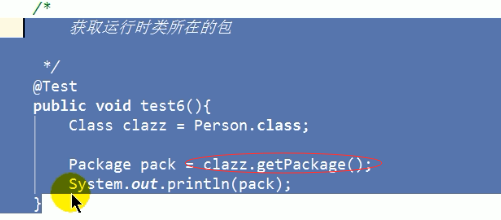




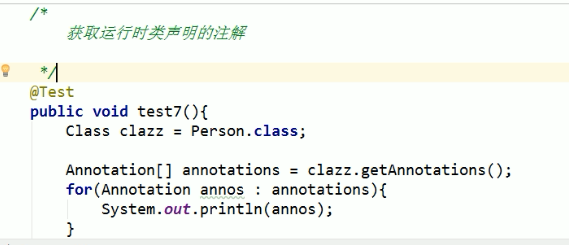
## 获取运行时类实现的接口（动态代理会用）



## 获取运行时类所在的包

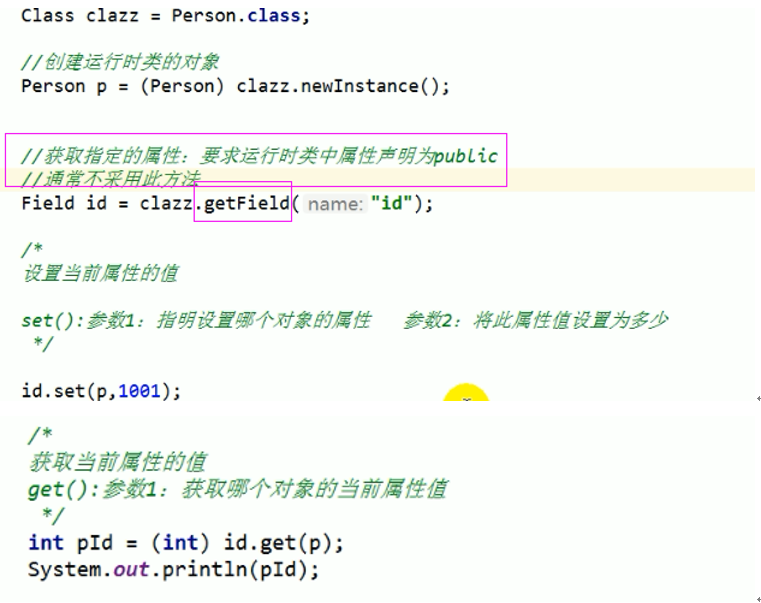


## 获取运行时类声明的注解



# 调用运行时类的指定结构（结构是属性，方法（重点），构造器）

## 属性



一般属性是private，所以我们用下面这个



## 方法

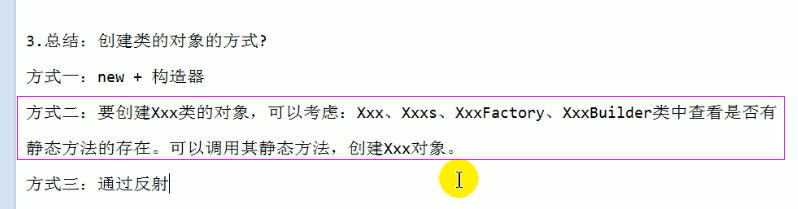




## 调用运行时类中的指定构造器（不常用，调构造器是造对象，我们一般调用空构造器）

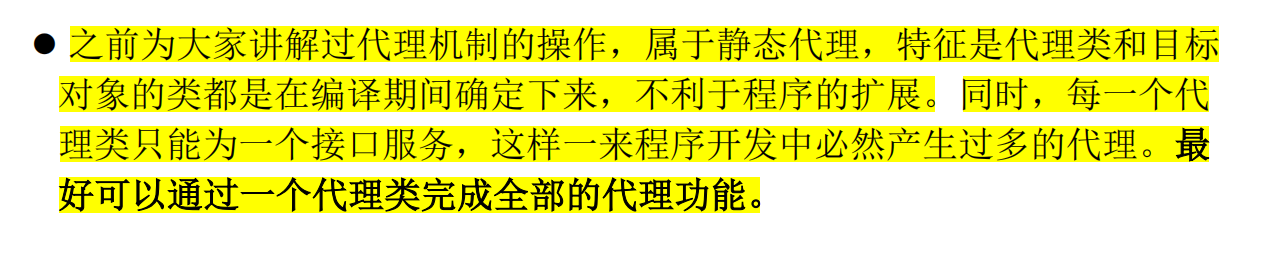


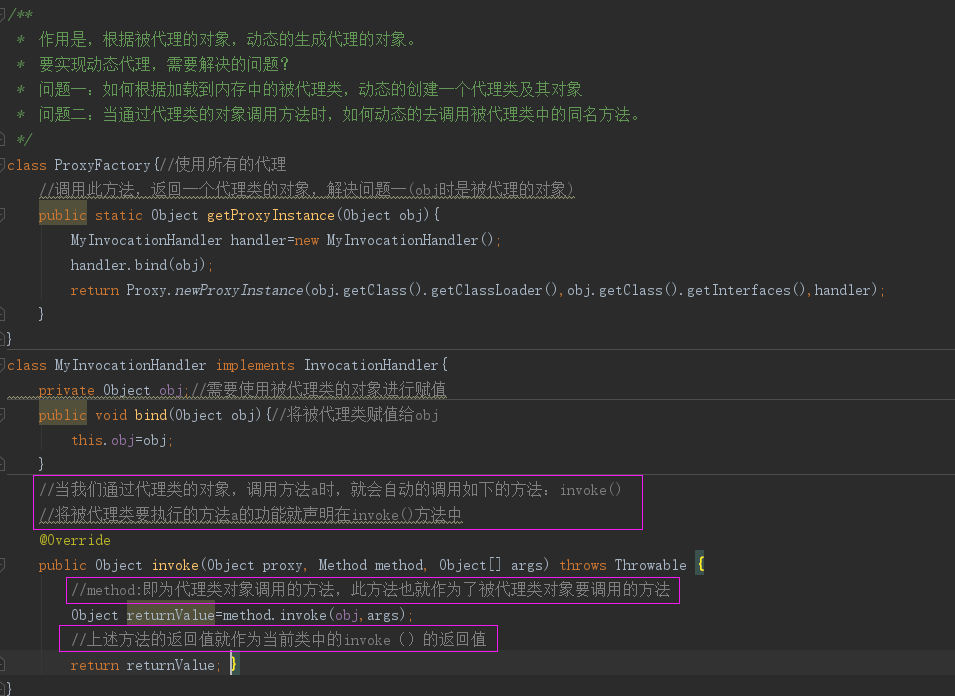
# 创建类的对象方式



# 反射的应用：动态代理

IOC容器的常见方式是依赖注入，AOP是动态代理







# 动态代理与AOP(Aspect Orient Programming)

