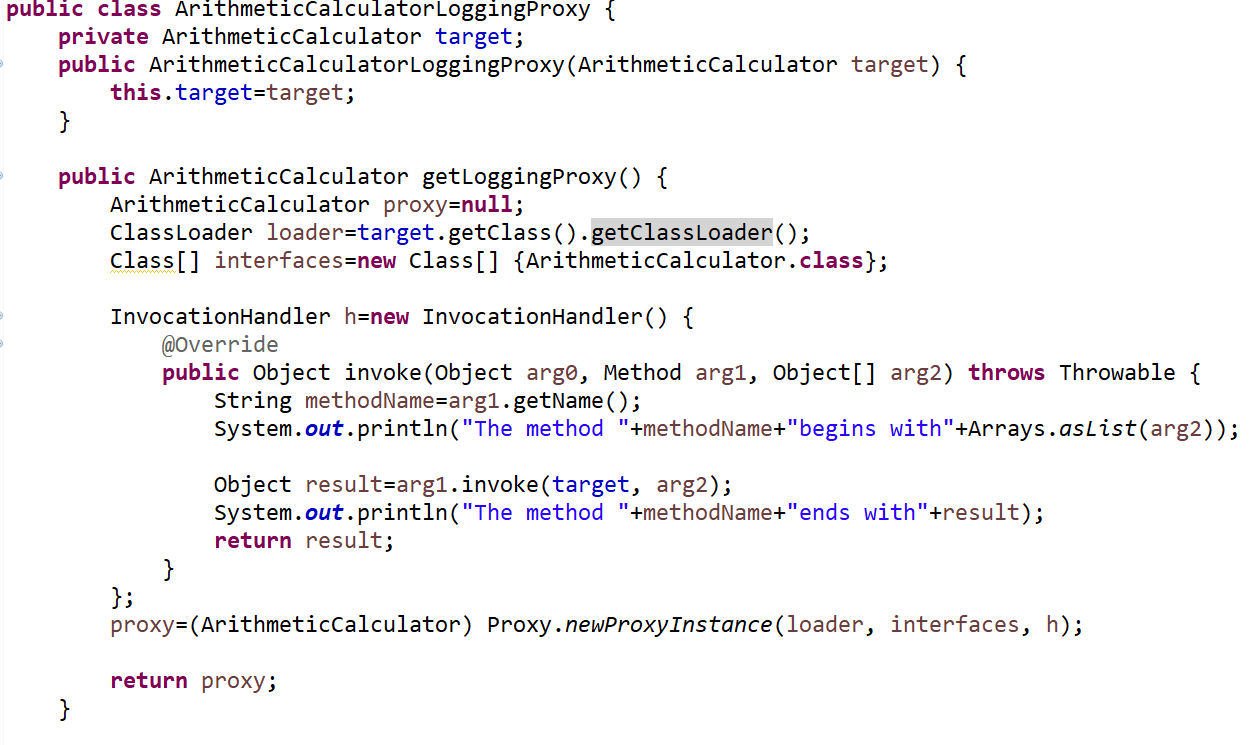
1. 用动态代理为方法添加日志



1. AOP简介
   * 1. .AOP(Aspect-Oriented Programming, **面向切面编程**): 是一种新的方法论, 是对传统 OOP(Object-Oriented Programming, 面向对象编程) 的补充.

AOP 的主要编程对象是**切面**(aspect), 而**切面模块化横切关注点**.

②.在应用 AOP 编程时, 仍然需要**定义公共功能**, 但可以明确的定义这个功能在哪里, 以什么方式应用, **并且不必修改受影响的类**. 这样一来**横切关注点就被模块化到特殊的对象(切面)**里.

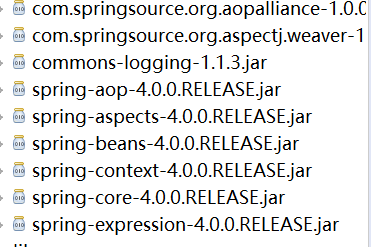
1. .AOP 的好处:
   1. 每个事物逻辑位于一个位置, 代码不分散, 便于维护和升级
   2. 业务模块更简洁, 只包含核心业务代码.

3.AOP术语

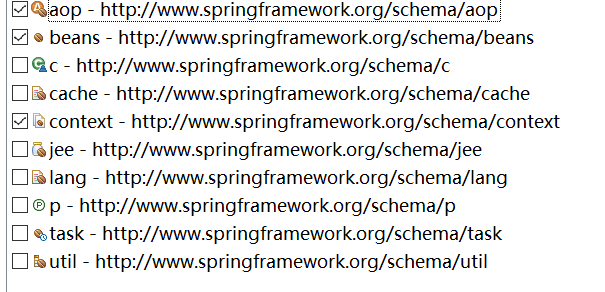
* 切面(Aspect): **横切关注点(跨越应用程序多个模块的功能)被模块化的特殊对象**
* 通知(Advice): **切面必须要完成的工作**
* 目标(Target): **被通知的对象**
* 代理(Proxy): **向目标对象应用通知之后创建的对象**
* 连接点（Joinpoint）：**程序执行的某个特定位置**：如类某个方法调用前、调用后、方法抛出异常后等。**连接点由两个信息确定：方法表示的程序执行点；相对点表示的方位**。例如 ArithmethicCalculator#add() 方法执行前的连接点，执行点为 ArithmethicCalculator#add()； 方位为该方法执行前的位置
* 切点（pointcut）：**每个类都拥有多个连接点**：例如 ArithmethicCalculator 的所有方法实际上都是连接点，即**连接点是程序类中客观存在的事务**。**AOP 通过切点定位到特定的连接点。类比：连接点相当于数据库中的记录，切点相当于查询条件**。切点和连接点不是一对一的关系，一个切点匹配多个连接点，切点通过 org.springframework.aop.Pointcut 接口进行描述，它使用类和方法作为连接点的查询条件。

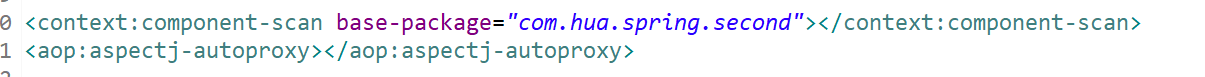
4．AOP体验

* + 1. 导入jar包

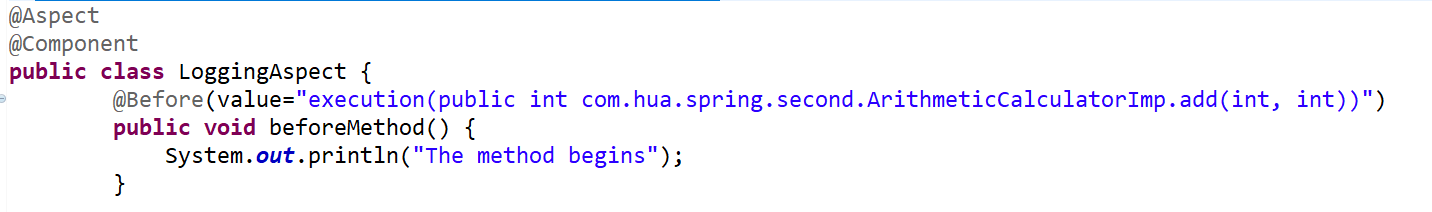


* + 1. 创建xml配置文件，导入context、aop命名空间

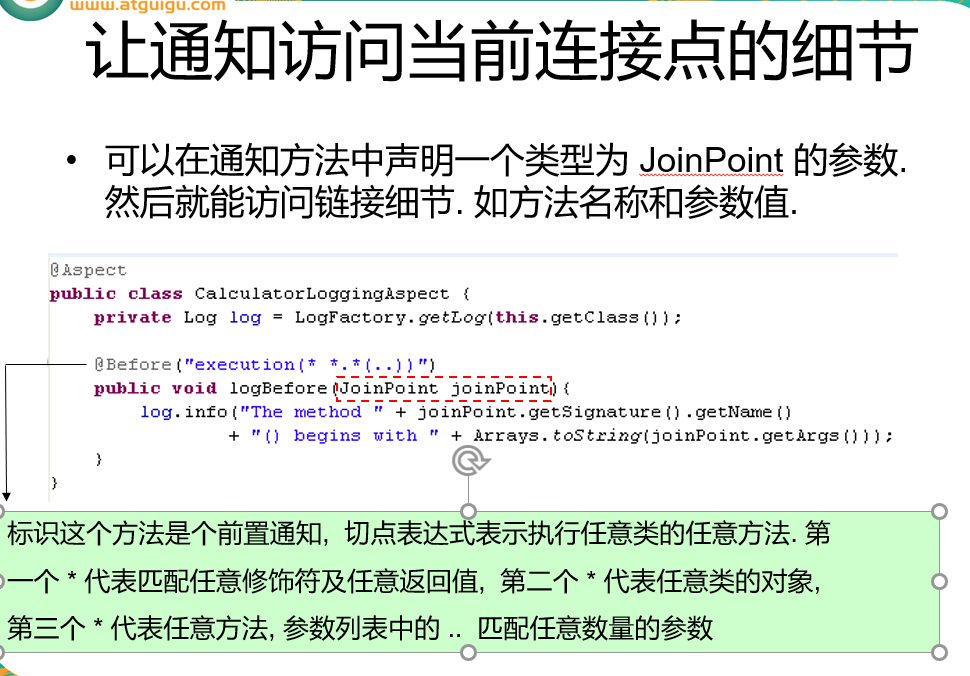




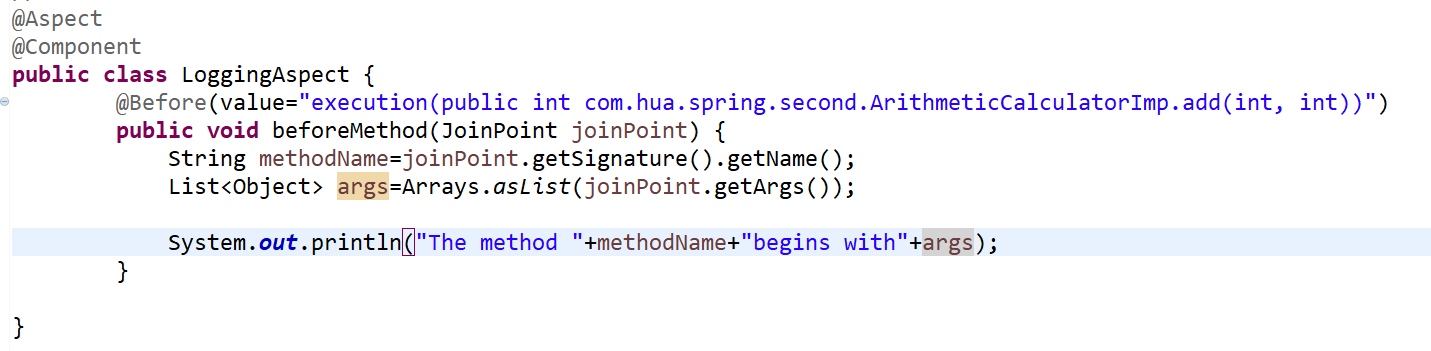
* + 1. 写一个切面类，把这个类声明为一个切面，首先需要把该类放入到IOC容器中，再声明为一个切面



* + 1. 可以用\*等符号来代替返回值、包名，方法名

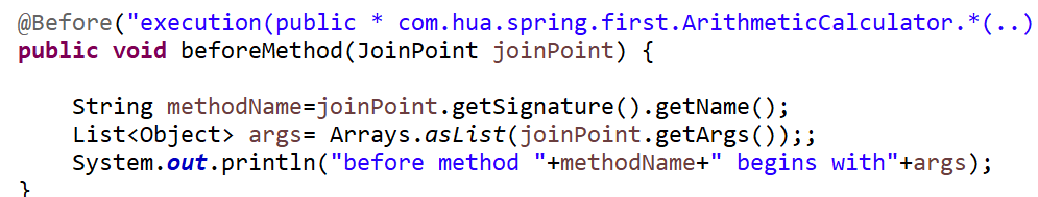


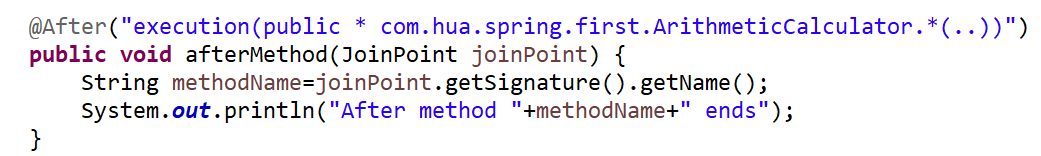
* + 1. 可以在方法的参数列表里加入JoinPoint类型参数以获取当前方法的信息



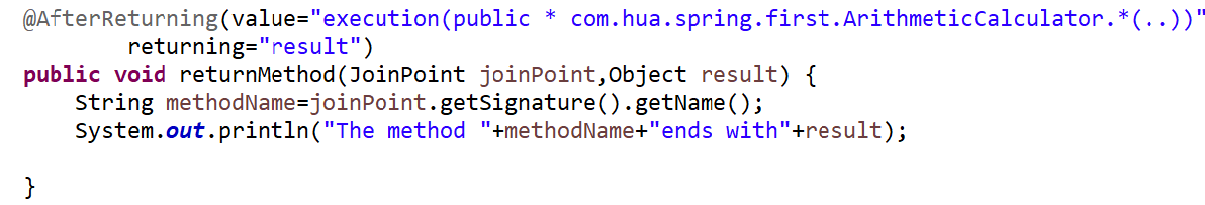
5.前置通知，后置通知，返回通知，异常通知，环绕通知

①.前置通知：在方法执行之前进行的通知

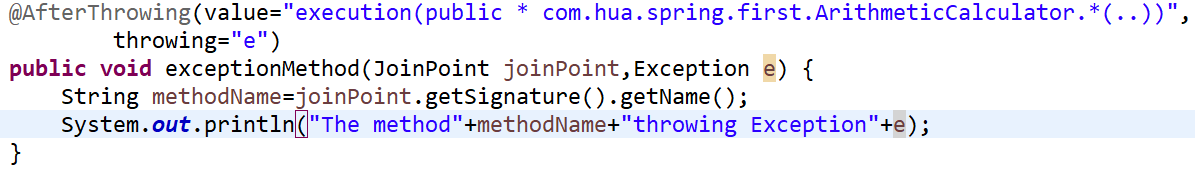
②.后置通知：一定会执行后置通知



③．返回通知：在方法正常结束执行的通知，返回通知是可以访问到返回值的



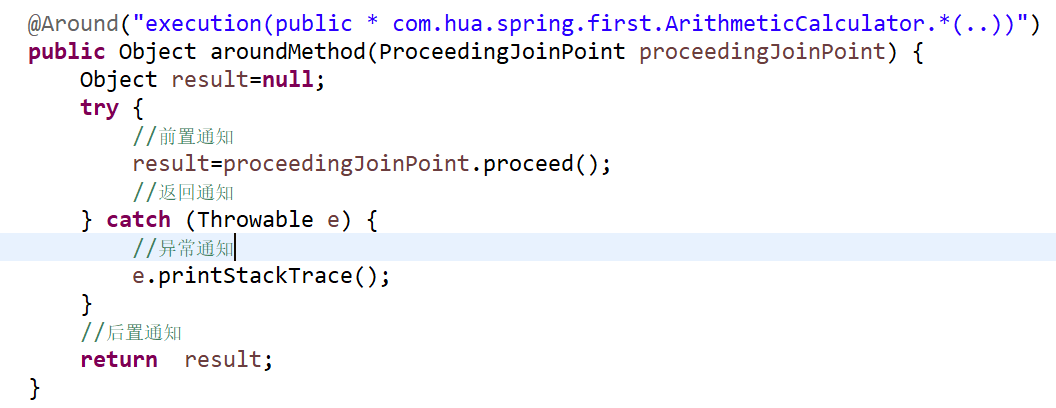
④．在目标方法出现异常时会执行的代码，，可以访问到异常对象，且可以指定在出现特定异常时在执行通知代码



1. .环绕通知：环绕通知需要携带ProceedingJoinPoint类型的参数

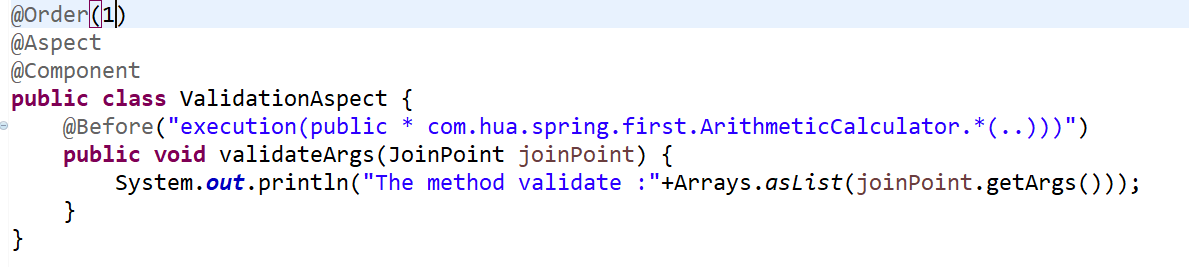
环绕通知类似于动态代理的全过程ProceedingJoinPoint类型的参数可以决定是否执行目标方法

且环绕通知必须有返回值，即为执行目标方法后的返回值



6.切面的优先级

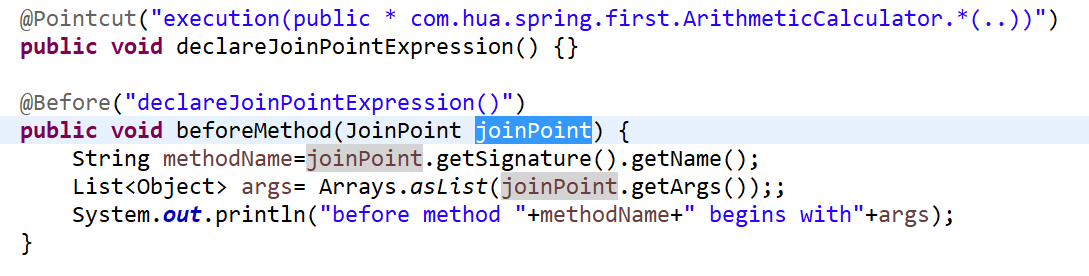
可以使用@order(i)注解来表明切面的优先级，值越小，优先级越高



7.重用切点表达式

定义一个方法，用于声明切入点表达式，一般地，该方法不需要填入其他的代码

后面的其他通知直接使用方法来引用切入点表达式



8.使用XML文件的方式来配置AOP

