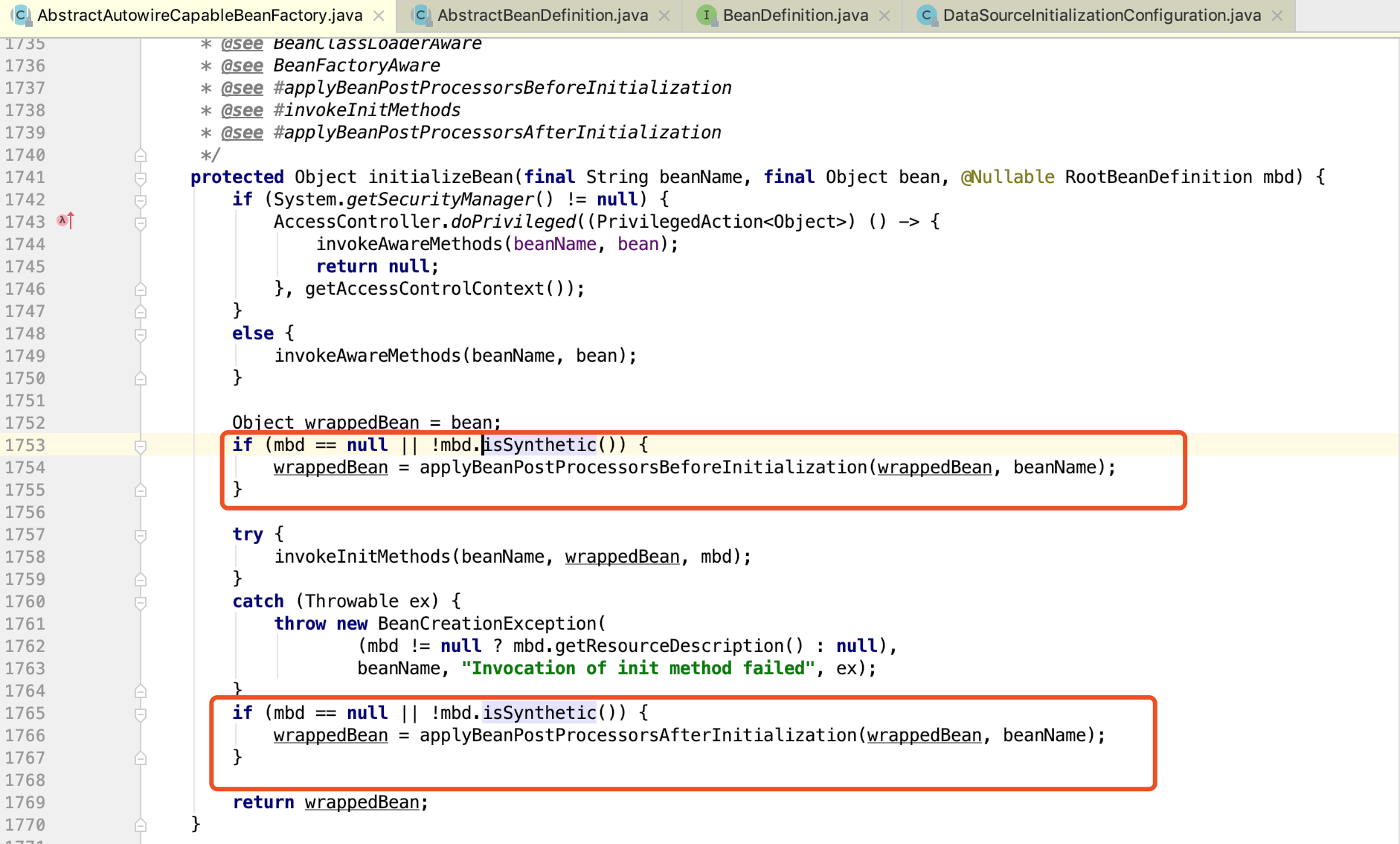
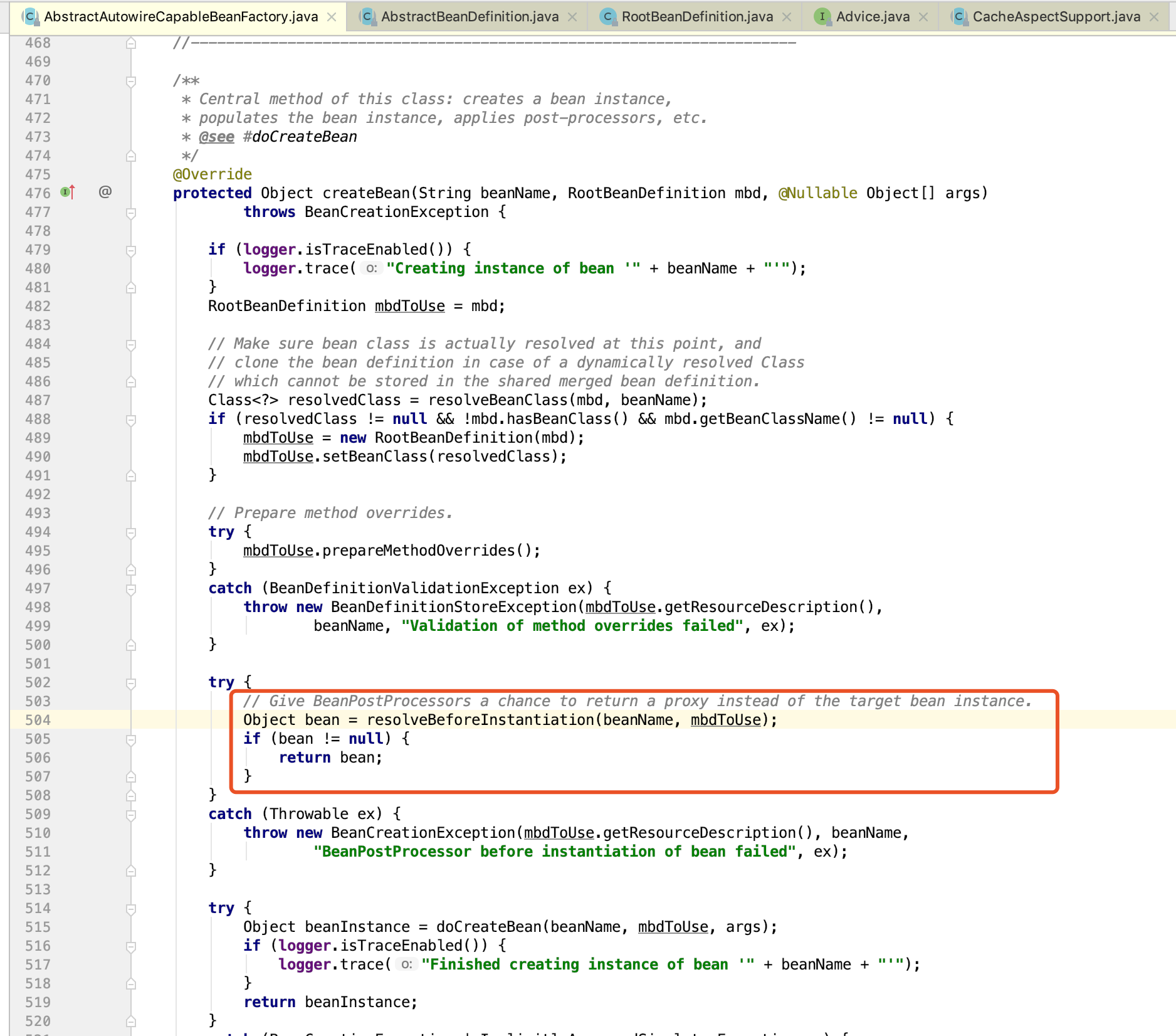
**查看spring源码时发现AbstractBeanDefinition包含有**

synthetic属性含义就是在通过BeanDefintion生成实例，并将其注入到spring容器中时，如果该属性为true，这表示这个bean不会进行bean的后置处理。代码如下：

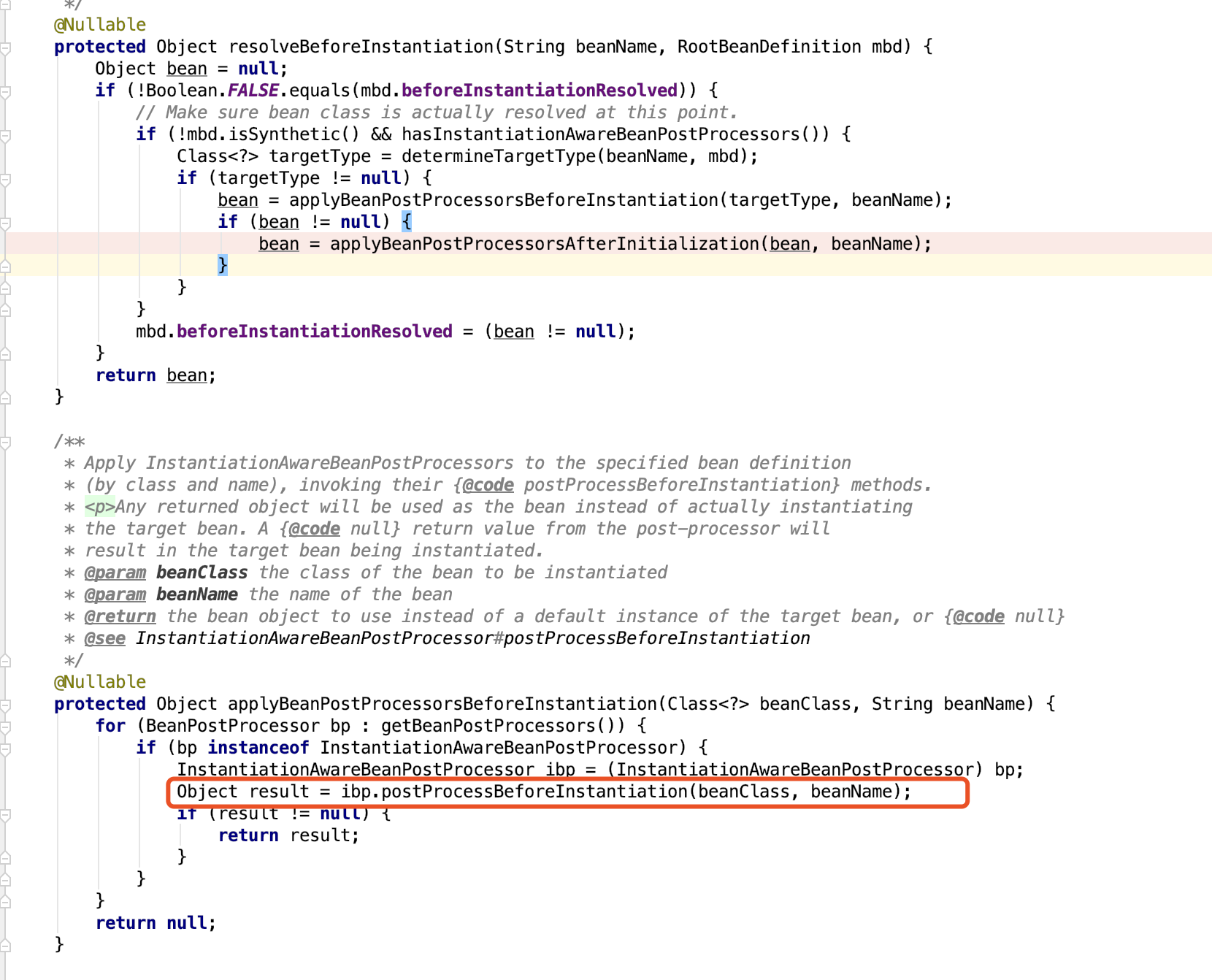


spring 在实例化bean时，会调用AbstractAutowireCapableBeanFactory

的createBean方法，如下



该方法中会调用resolveBeforeInstantiation方法，此时会判断BeanDefintion是否允许被代理（synthetic属性），同时是否包含InstantiationAwareBeanPostProcessor的实例，如果两个条件满足，就会调用InstantiationAwareBeanPostProcessor实例的postProcessBeforeInstantiation方法，如下：



InstantiationAwareBeanPostProcessor实例有如下：

[AbstractAdvisorAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/AbstractAdvisorAutoProxyCreator.html), [AbstractAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/AbstractAutoProxyCreator.html), [AnnotationAwareAspectJAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/aspectj/annotation/AnnotationAwareAspectJAutoProxyCreator.html), [AspectJAwareAdvisorAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/aspectj/autoproxy/AspectJAwareAdvisorAutoProxyCreator.html), [AutowiredAnnotationBeanPostProcessor](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/beans/factory/annotation/AutowiredAnnotationBeanPostProcessor.html), [BeanNameAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/BeanNameAutoProxyCreator.html), [DefaultAdvisorAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/DefaultAdvisorAutoProxyCreator.html), [InfrastructureAdvisorAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/InfrastructureAdvisorAutoProxyCreator.html), [InstantiationAwareBeanPostProcessorAdapter](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/beans/factory/config/InstantiationAwareBeanPostProcessorAdapter.html), [RequiredAnnotationBeanPostProcessor](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/beans/factory/annotation/RequiredAnnotationBeanPostProcessor.html), [ScriptFactoryPostProcessor](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/scripting/support/ScriptFactoryPostProcessor.html)

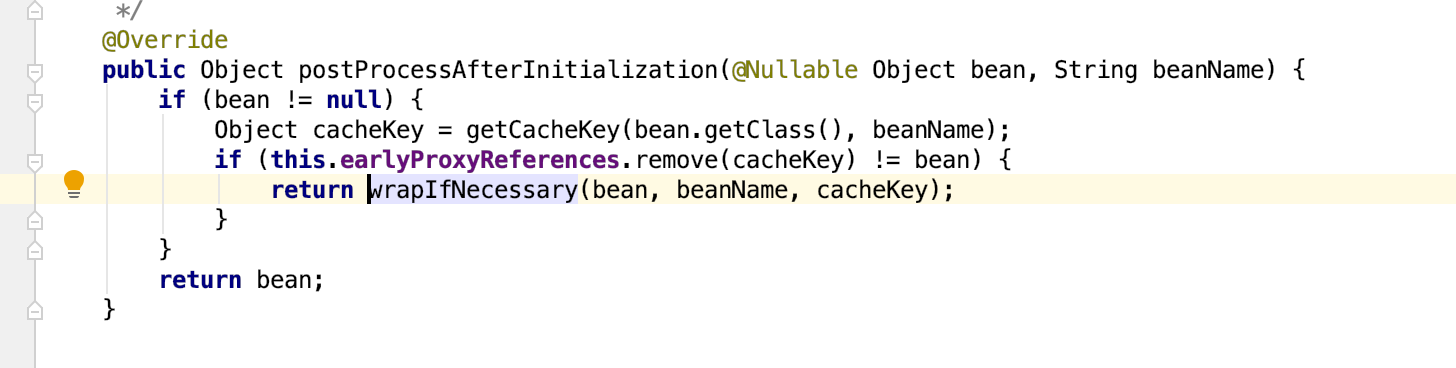
当我们开启事物时，会注入[AbstractAutoProxyCreator](https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/aop/framework/autoproxy/AbstractAutoProxyCreator.html" \o "class in org.springframework.aop.framework.autoproxy)，查看其postProcessBeforeInstantiation

方法实现



具体逻辑，就是先获取缓存对应的key，然后判断beanClass 是否实现了Advice，Pointcut

，Advisor，AopInfrastructureBean这4个接口，如果没有实现这4个接口，则再判断是否跳过，最终**advisedBeans**会存放实现了上述4个接口的bean。调试可以看到，这个方法不会生成代理，最后还是要执行后者处理器，如下：

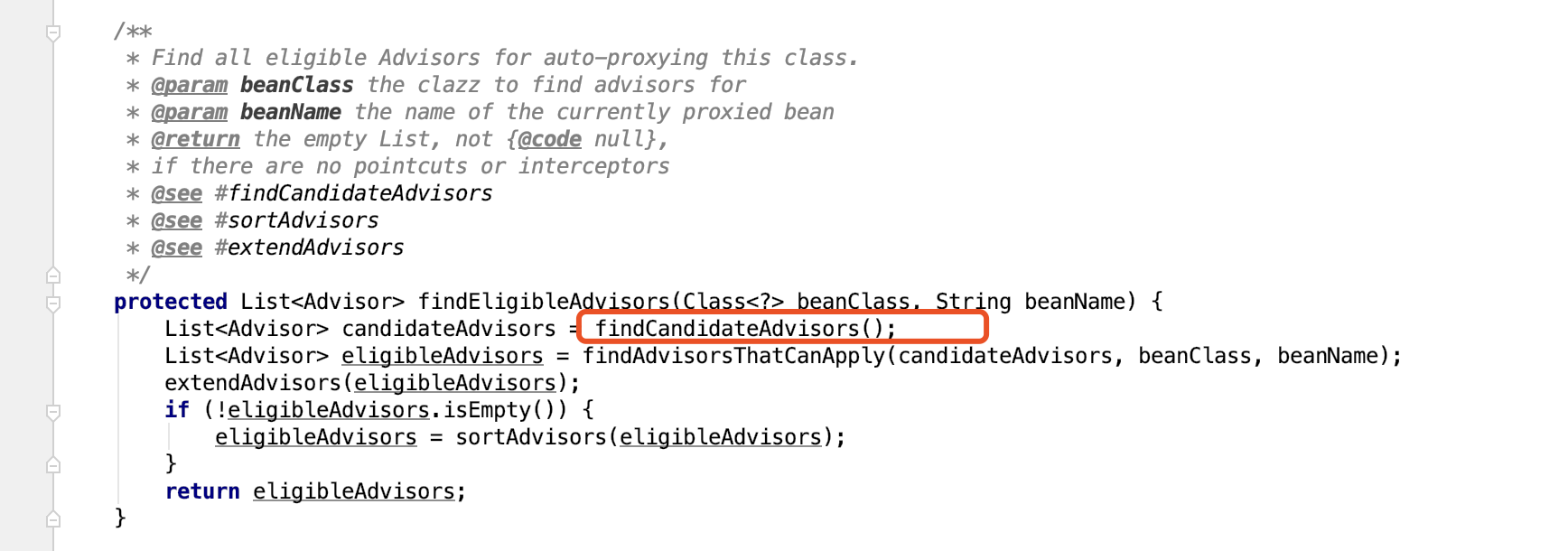




getAdvicesAndAdvisorsForBean方法的大致作用：

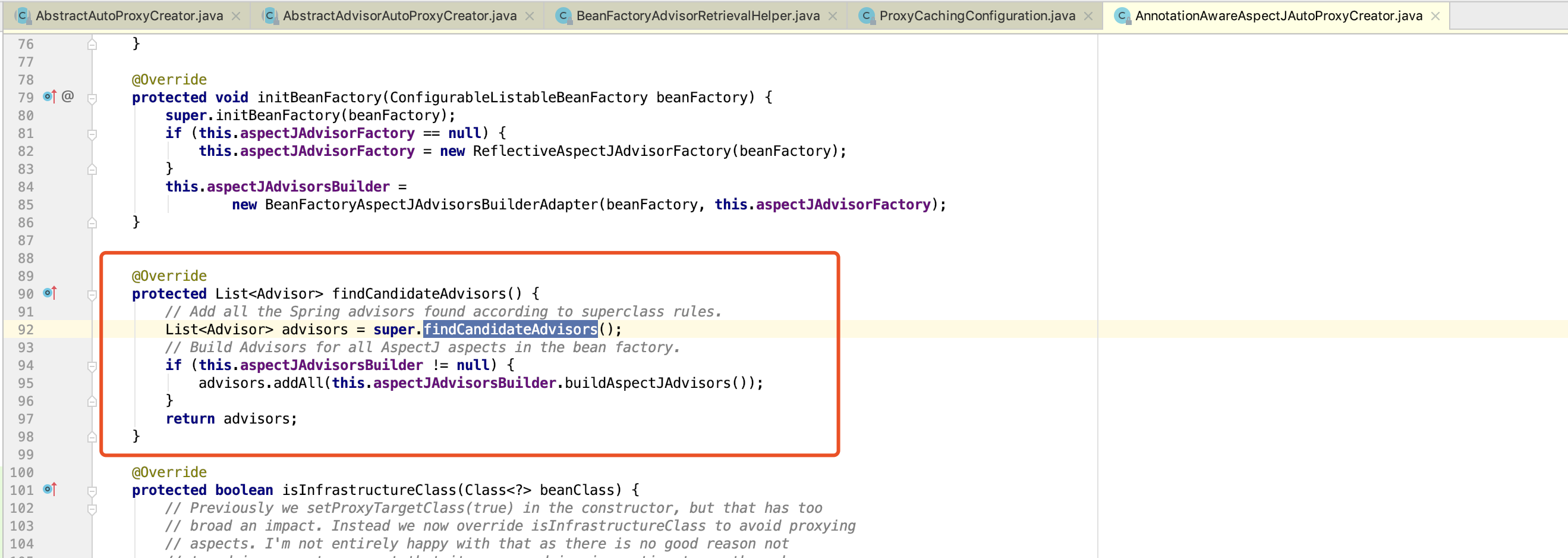
先获取所有advice和advisors。然后根据bean的class，获取其所有的父接口，然后扫描每个方法，再根据每个advisors去匹配每个方法，确认其方法是否需要advisors，需要就将advisors保存下来，用于后续代理生成后，将advisors塞到代理中。

具体如下：



先通过方法findCandidateAdvisors寻找advisors。

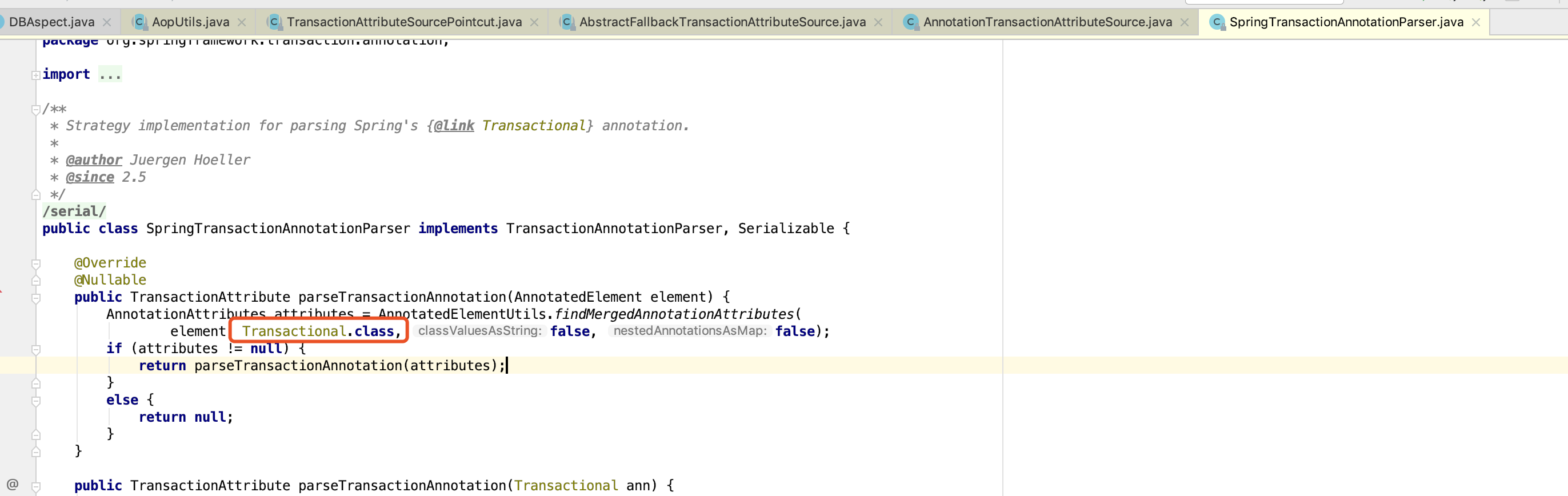
findCandidateAdvisors方法如下：



实际就是通过beanFactory寻找所有实现了Advisor接口的bean，与事物相关的就是BeanFactoryTransactionAttributeSourceAdvisor，与缓存相关的就是BeanFactoryCacheOperationSourceAdvisor,接着找由@Aspect注解生成的Advisor。

由@Aspect注解生成的Advisor都是继承InstantiationModelAwarePointcutAdvisor

然后调用findAdvisorsThatCanApply方法去将每个Advisor与bean去匹配，扫描bean及其所有父接口的每个方法使用的注解，类上使用的注解。查看是否有使用相关的注解，例如事物：



最终获取该bean需要的Advisor集合，然后对Advisors进行排序，如下：