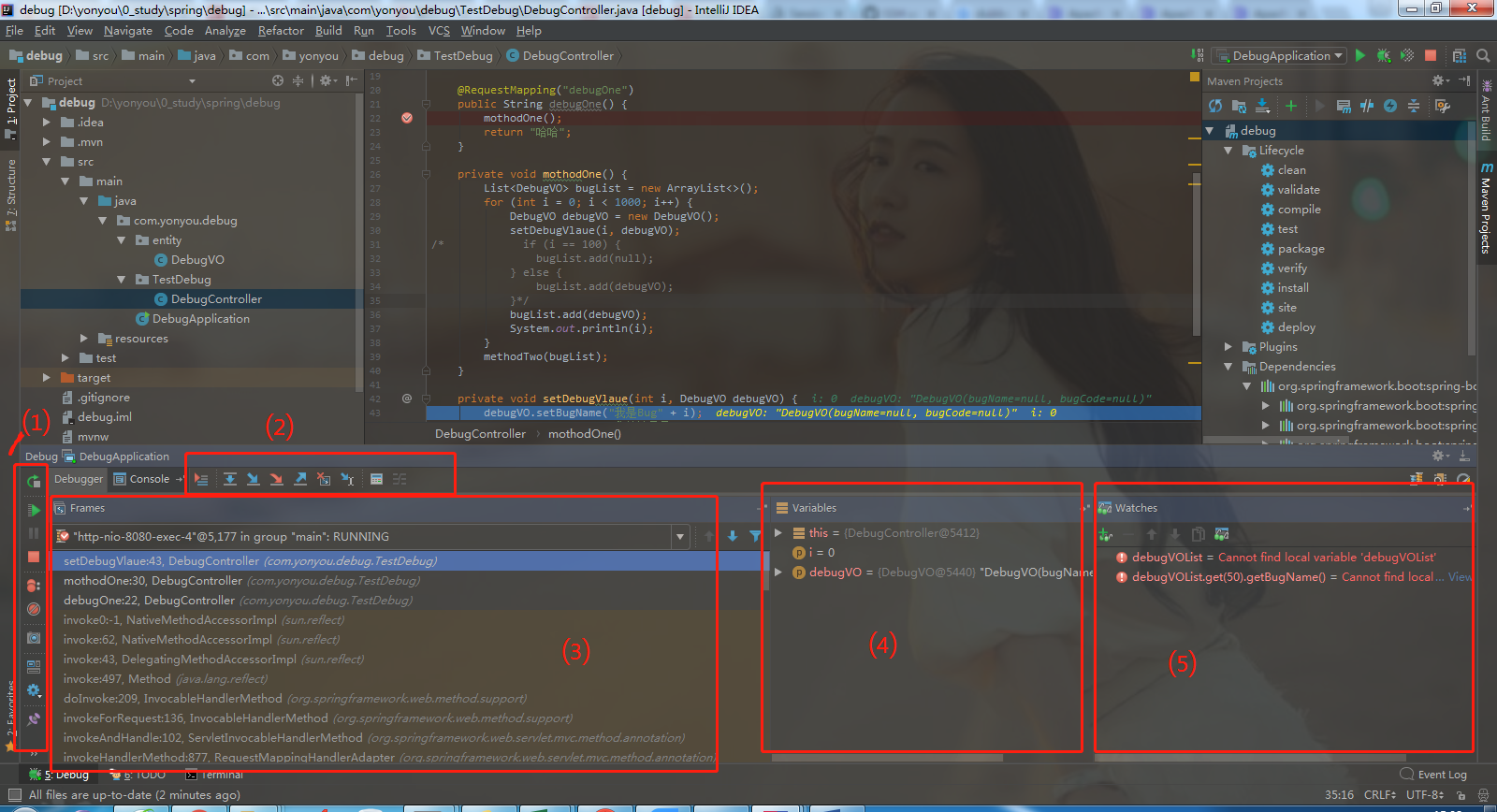
## 1工具面板基本介绍



以下命名为自定义命名，旨在更好理解，不代表官方

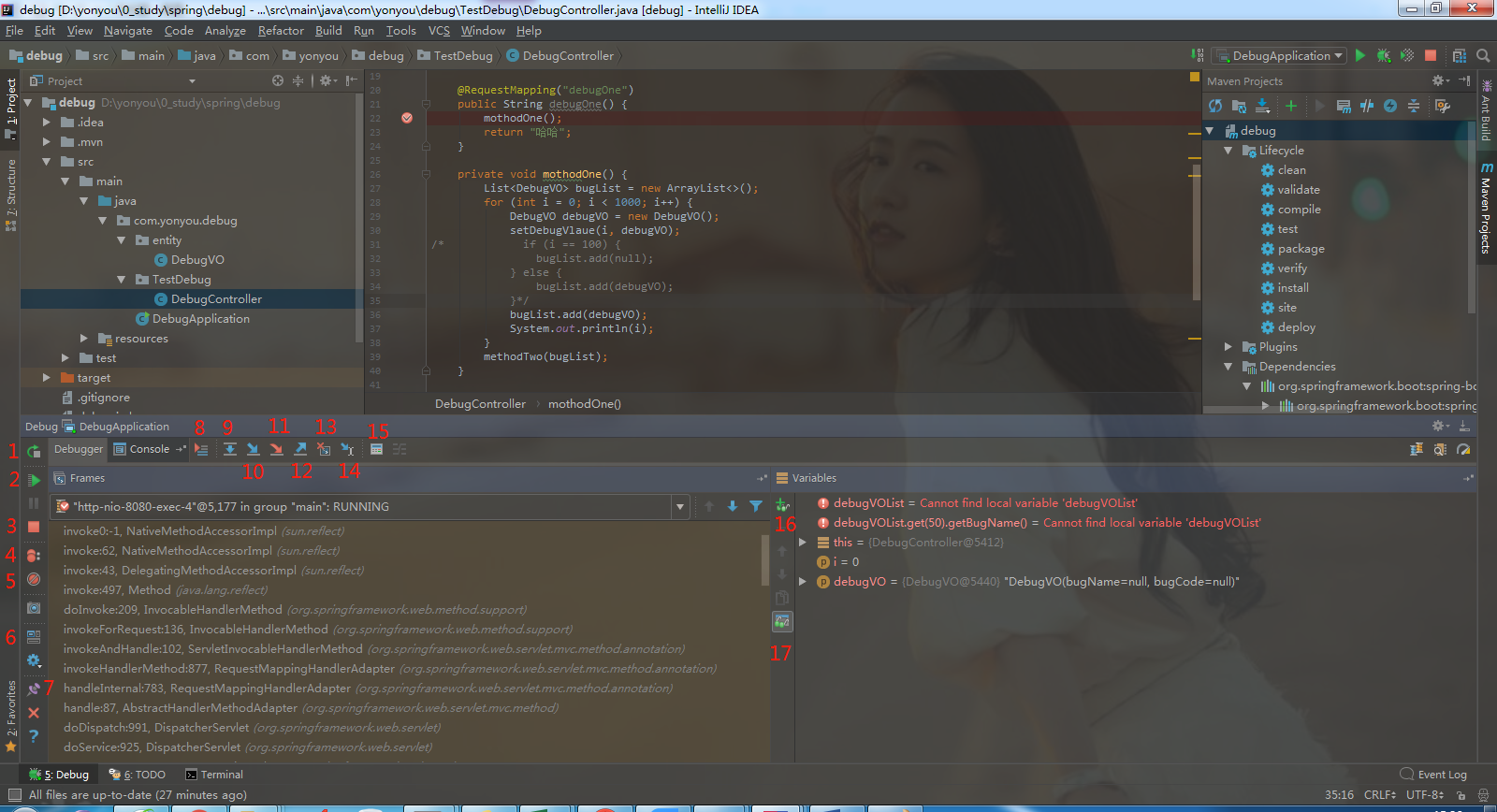
⑴⑵两个区域为控制区，可以控制断点和程序，稍后会每个功能按钮详细讲解

⑶是方法调用区，展现了方法调用的一个堆栈信息，越是上面越是后被调用的方法，最上面的为当前断点所停在的方法

⑷ 变量区，断点处目前生效的变量对应的值会在这个区域显示

⑸ 观察区，用于自定义内容的观察，后续会详细讲解

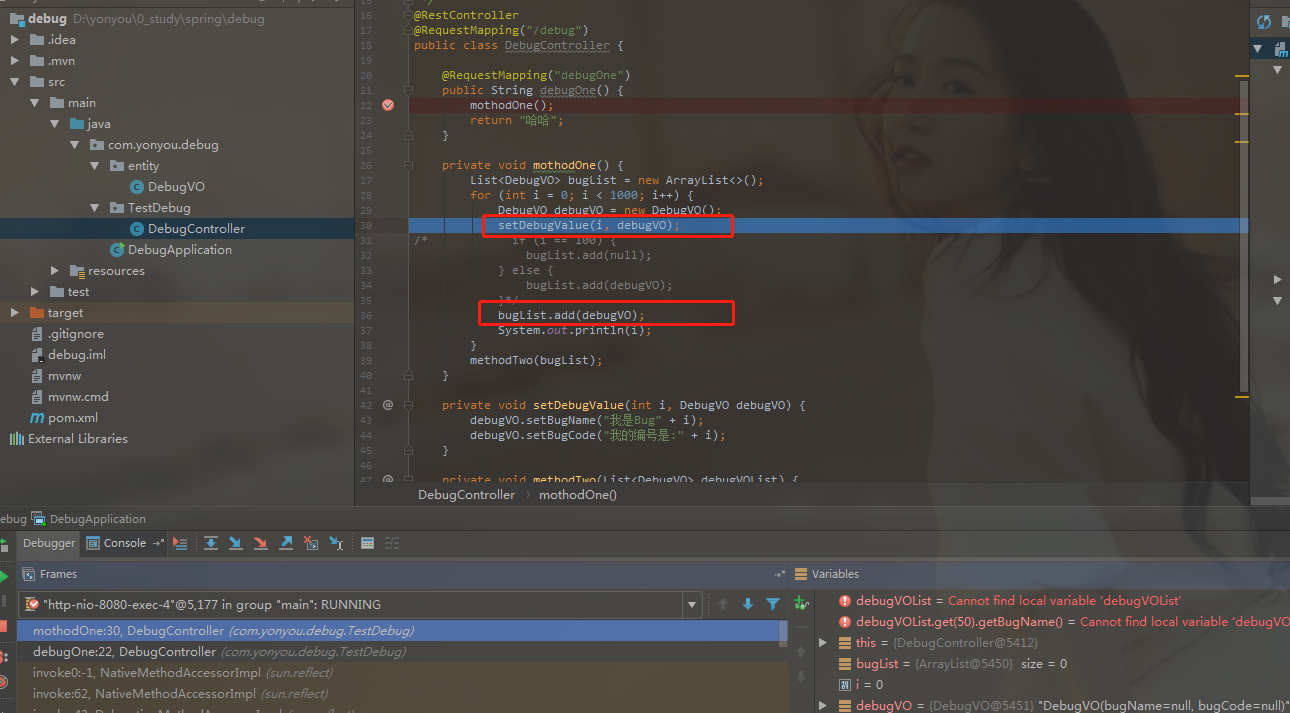
## 2.具体功能点说明



1、rerun “当前启动的程序名称”，这个就是直接重新跑这个项目。  
2、直接往下跑程序，直到遇到下一个断点停下，没有就直接跑完程序。  
3、停止项目或者程序。  
4、查看所有的断点设置情况。具体详情，下面有示意图，再细细解释。  
5、让所有断点无效。再次点击全部生效  
6、要是你一不小心把这个下面的布局给弄乱了，你点这个，就把下面的布局给还原咯。

7、锁住当前程序tab页，可以防止误删tab页（不常用）  
8、跳转到当前代码所执行的地方，也就是说你在看代码的时候，点到其他地方，一点这个按钮，就到了程序执行到当前哪行的代码的地方。  
9、下一步，如果是方法，他是不会跳进去的。就是一行行的往下走。

10、进入方法，如果当前断点是一个方法调用代码，点击后能进入到方法（非强制，非自己定义的方法可能不会进入，屈居于点击时如果是在自己定义的类中只能进入自己定义的方法中，不能进入底层代码的方法中，这块有点绕，举个例子）



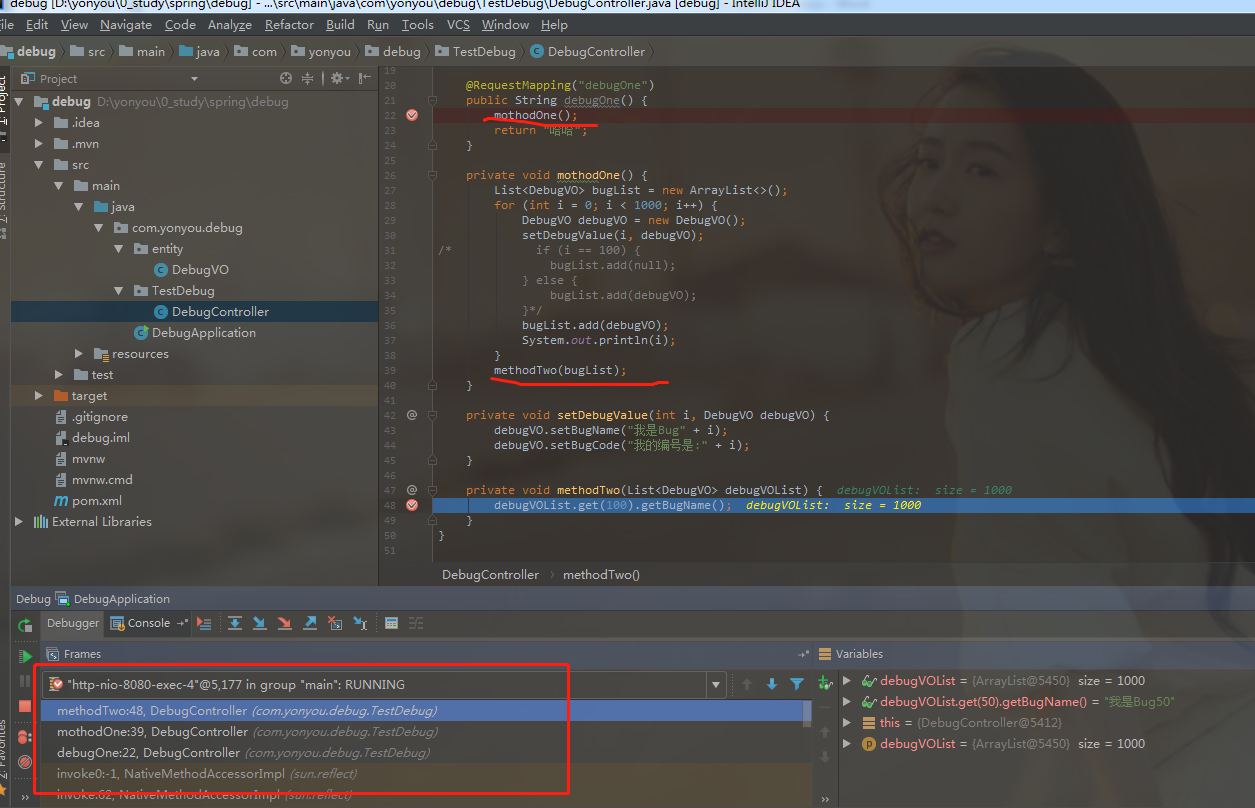
现在断点处于setDebugValue方法调用处，如果点击按钮10，就可以进入，因为此处的方法时我自己的源码，如果再往走，走到bugList.add方法，这时再点击按钮10因为list.add方法是底层的代码了，所以不会进入，这时可以点击按钮11

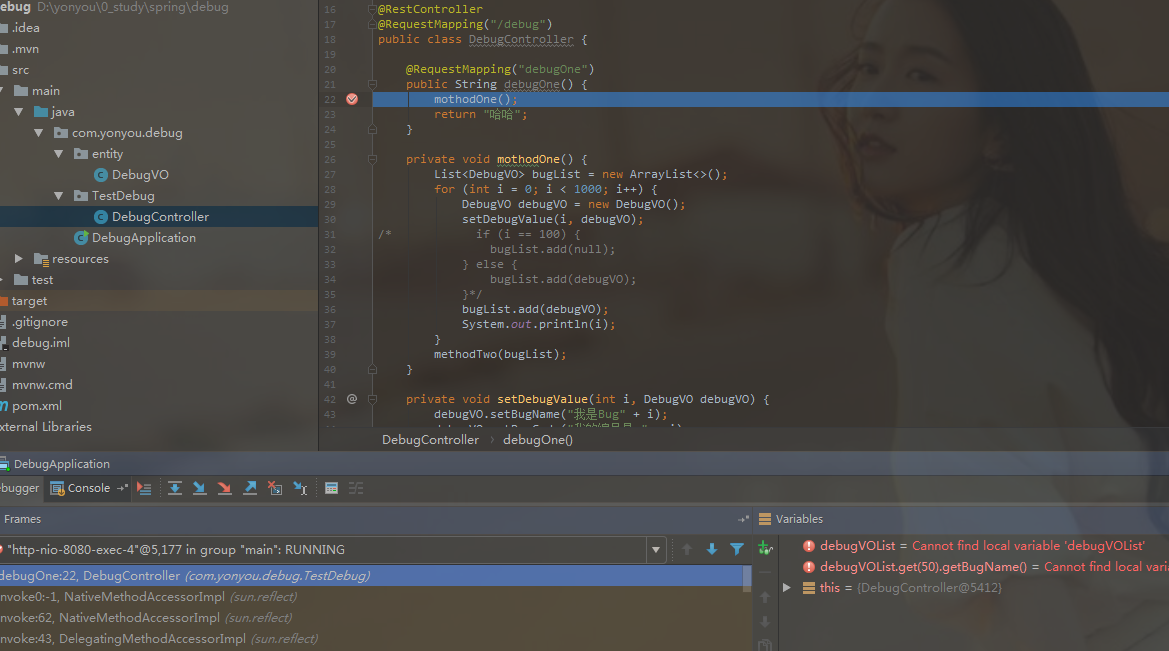
11、强制进入，只要是方法我就进入。

12、从方法中跳出去，和上面的10相反。

13、有时候当我们步入方法体之后，还想回退到方法体外，点这个按钮后，断点重新回到方法体之外。在继续还是可以再次进到方法内（这个功能很有用，可以回到方法调用栈中任意方法调用的位置，错过了还可以再回头，因为重要，所以举个例子，看下图）

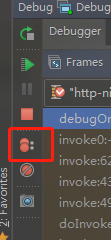
分析：看方法调用栈数据debugOne方法在22行调用了方法methodOne，methodOne在39行调用了methodTwo，现在我想从新回到之前，因为我跟问题跟过头了（比方说），咱们一种方式是让断点全走完，然后重新访问，重新跟，这样很明显很浪费时间，还有一种就是选择一个对应的方法栈中的方法调用，是的是选择一个，然后点击按钮13，就回到了方法调用的点（记住是调用的点），例如我选择methodOne，就是代表我想回到这个方法调用的地方，点击按钮13，就回到了22行，方法调用的地方。看图，是不很爽！！！很有用，很有用，很有用！！！



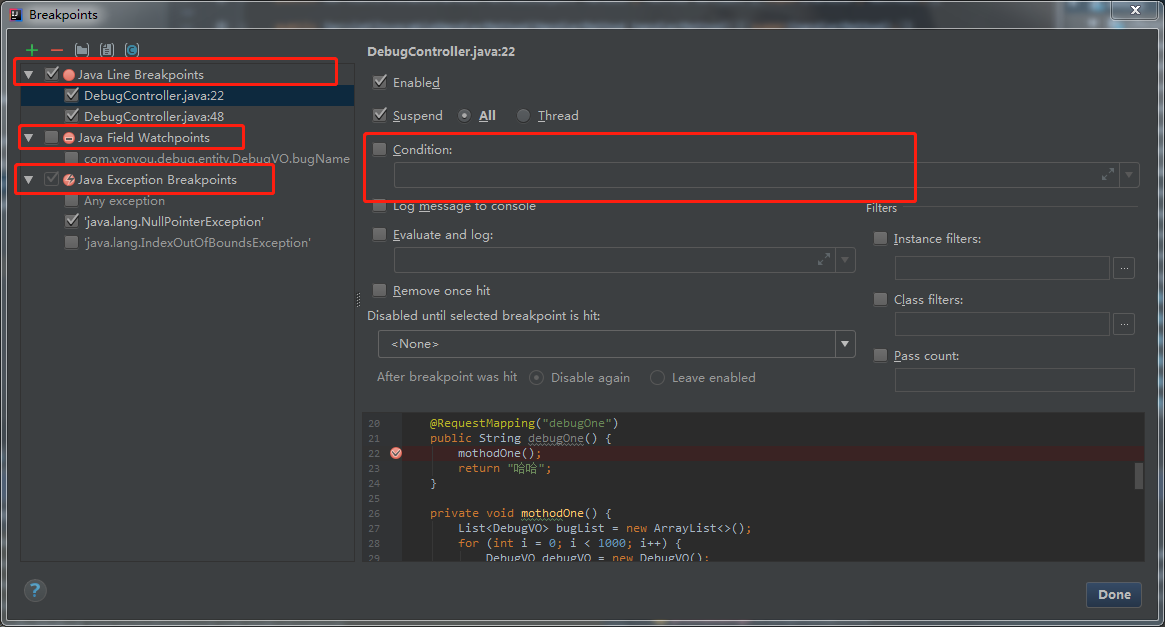
  
14，看字面意思就是跳转到那个光标所在的地方，前提是他能路上没断点拦着。  
15，这个点开之后，可以计算你想要看的代码段的值，后面详细上图。  
16，查看某个对象或表达式的值，自定义，前提是表达式中的变量在当前断点处事有意义的。  
17，把自定义的查看对象的值，分开到另一个tab页。

## 3.具体类型断点讲解

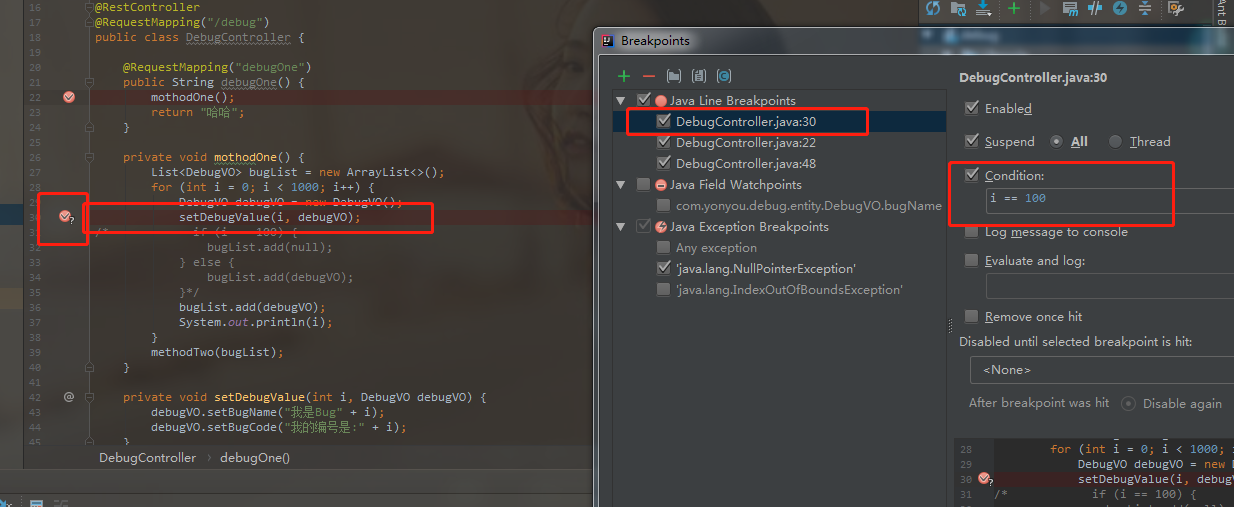
常用断点包括：**行断点**（普通断点+条件断点），**属性字段断点**（又叫观察断点），**异常断点**

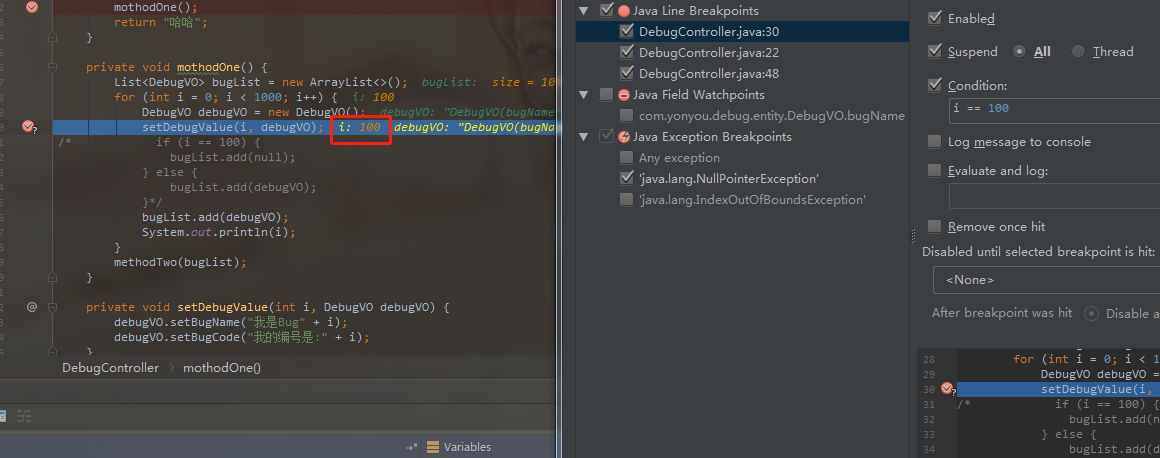
点击按钮4 

3.1 行断点



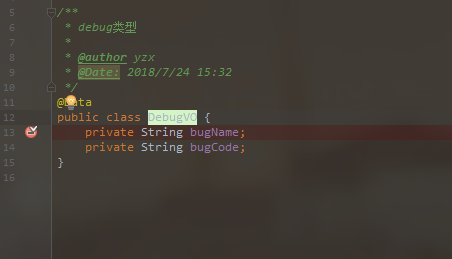
左侧，3种断点一目了然，其中行断点右侧condition勾选上，并写上表达式（表达式是booleanl类型作为结果的表达式）当表达式的值为true时，断点就停止了。例如





表达式我这里举的例子比较简单，根据自己的实际情况，循环中用的时候比较多

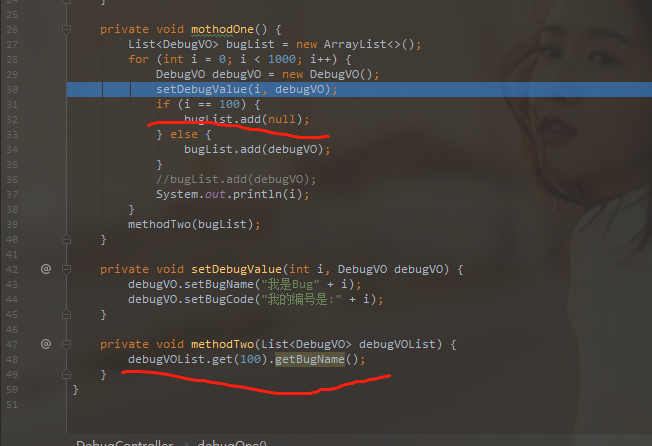
3.2 属性断点



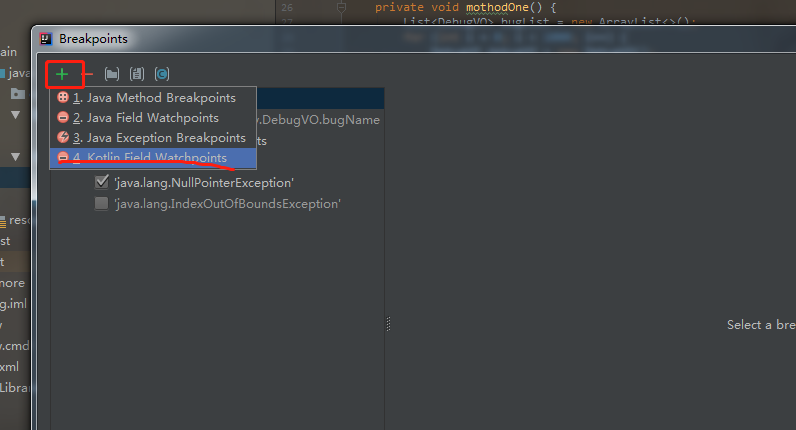
打在属性上，当这个值发生改变的时候就会进入断点，使用场景，不知道究竟哪改变了这个属性的值，可以通过这种方式跟踪。

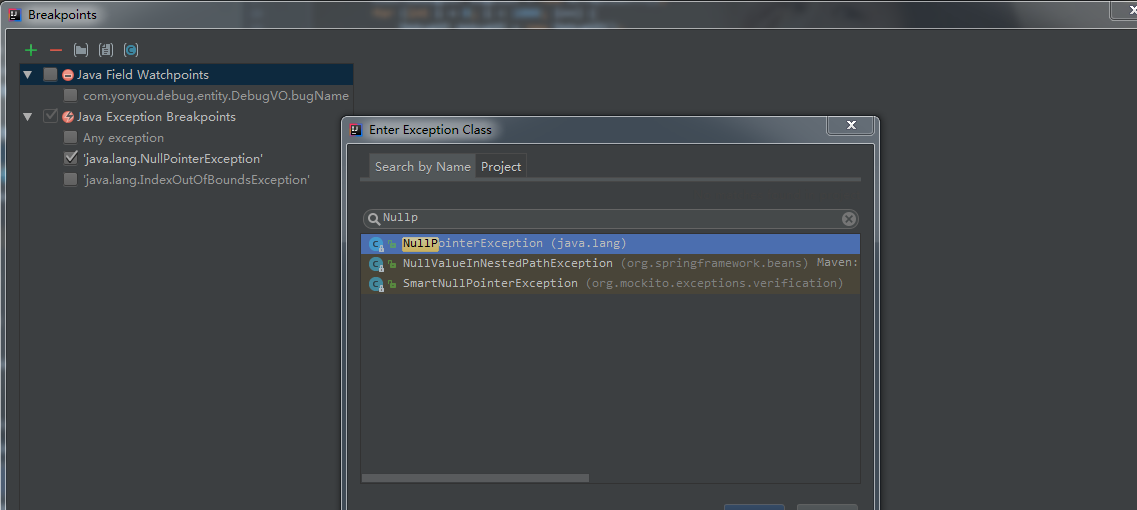
3.3 异常断点

看代码



我在i=100的时候list中加了一个null ,在methodTwo的时候我缺取了第100个，然后并取了它的名字，相当于null.getBugName()，会报异常，





输入异常类，然后确定就加完了，这种断点没有确切点在哪，哪里报错停在哪



这个位置我并没有打断点，但是走到这报异常了，它就停下来了