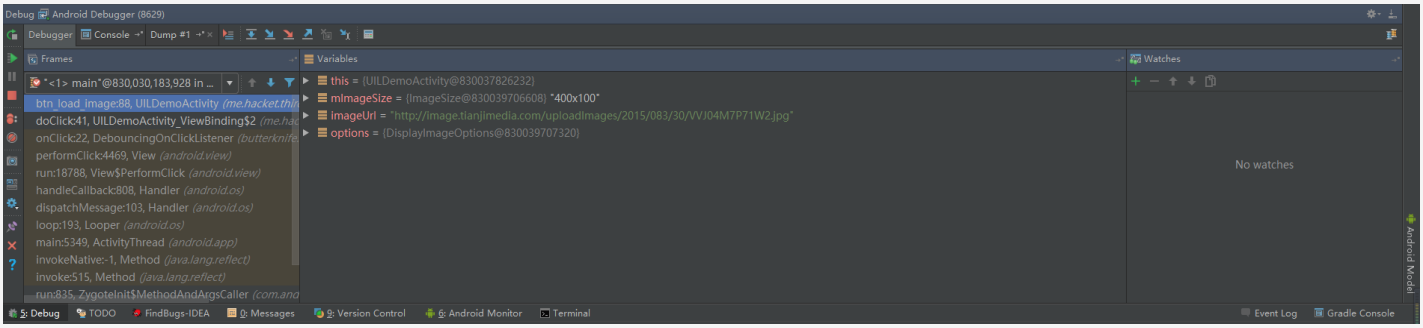
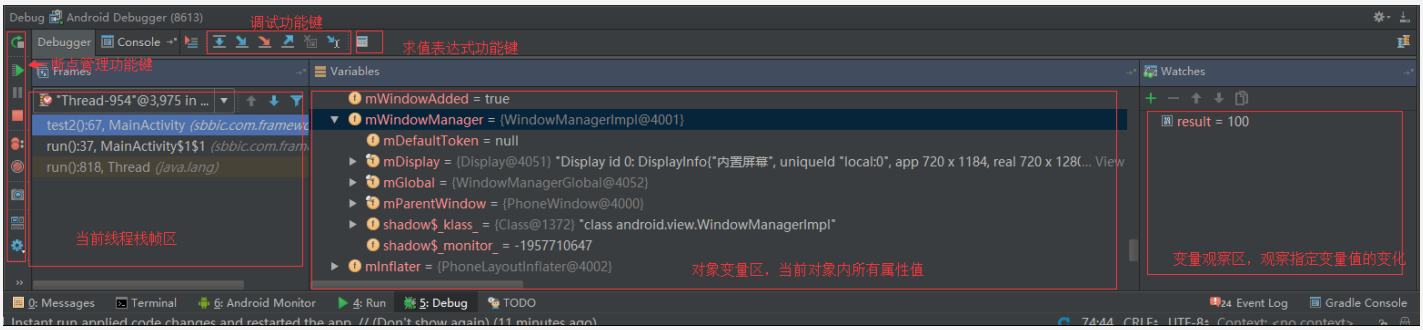
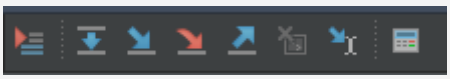
Android Studio调试技巧

调试面板





调试功能键区



- **Show Execution Point (Alt+F10)**

  点击该按钮,光标将定位到当前正在调试的位置

- **Step Over (F8)**

  单步跳过，点击该按钮将导致程序向下执行一行。如果当前行是一个方法调用，此行调用的方法被执行完毕后再到下一行

- **Step Into (F7)**

  单步跳入，执行该操作将导致程序向下执行一行。如果该行有自定义的方法，则进入该方法内部继续执行，需要注意如果是类库中的方法，则不会进入方法内部。

- **Force Step Into (Alt+Shift+F7)**

  强制单步跳入，和step into功能类似，主要区别在于：如果当前行有任何方法，则不管该方法是我们自行定义还是类库提供的，都能跳入到方法内部继续执行

- **Step Out (Shift+F8)**

- **Force Run to Cursor**

  非常好用的一个功能,可以忽视已经存在的断点,跳转到光标所在处

- **Drop Frame**

  没有好记的名字，大意理解为中断执行,并返回到方法执行的初始点,在这个过程中该方法对应的栈帧会从栈中移除.换言之,如果该方法是被调用的,则返回到当前方法被调用处，并且所有上下文变量的值也恢复到该方法未执行时的状态

- **Evaluate expression**

  点击该按钮会在当前调试的语句处嵌入一个交互式解释器，在该解释器中，你可以执行任何你想要执行的表达式进行求值操作

  > 快捷键：Alt+F8

断点管理区

- **Return**

  点击该按钮会停止目前的应用,并且重新启动.换言之,就是你想要重新调试时,可以使用该操作,嗯,就是重新来过的意思。

- **Pause Program**

  点击该按钮将暂停应用的执行.如果想要恢复则可以使用下面提到的Resume Program.

- **Resume Program**

该操作有恢复应用的含义,但是却有两种行为:

a. 在应用处在暂停状态下，点击该按钮将恢复应用运行.

b. 在很多情况下，我们会设置多个断点以便调试。在某些情况下，我们需要从当前断点移动到下一个断点处，两个断点之间的代码自动被执行，这样我们就不需要一步一步调试到下一个断点了，省时又省力。

- **Stop**

点击该按钮会通过相关的关闭脚本来终止当前进程.换言之,对不同类型的工程可能有不同的停止行为,比如:对普通的Java项目,点击该按钮意味着退出调试模式,但是应用还会执行完成.而在Android项目中,点击该按钮,则意味这app结束运行

- **View Breakpoints**

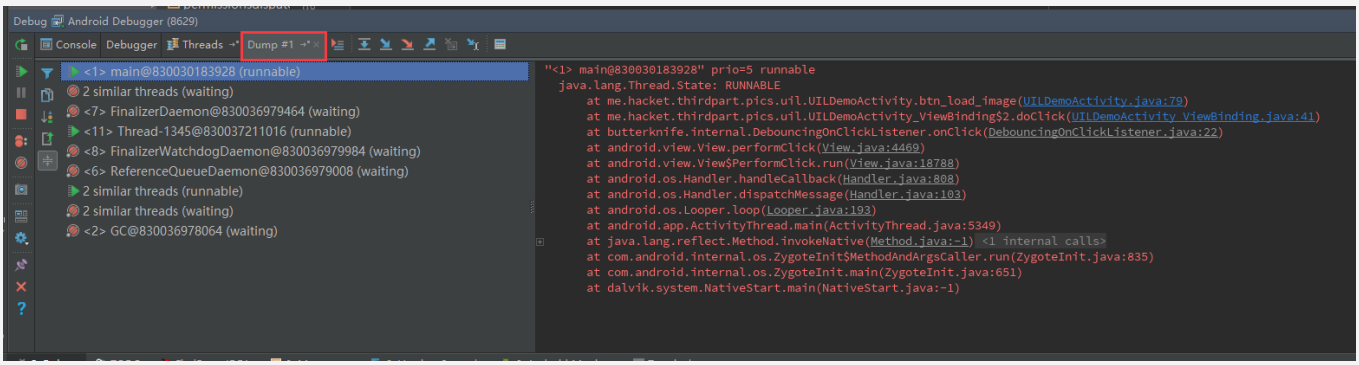
点击该按钮会进入断点管理界面，在这里你可以查看所有断点,管理或者配置断点的行为,如:删除，修改属性信息等

- **Mute Breakpoints**

使用该按钮来切换断点的状态:启动或者禁用.在调试过程中,你可以禁用暂时禁用所有的断点,已实现应用正常的运行.该功能非常有用,比如当你在调试过程中,突然不想让断点干扰你所关心的流程时,可以临时禁用断点

- **Get thread dump**

获取线程Dump,点击该按钮将进入线程Dump界面:



dump界面:

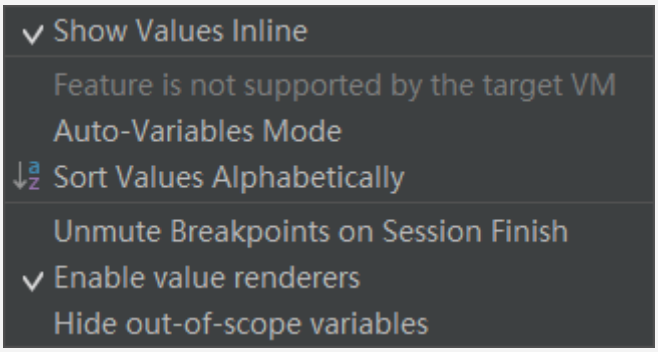
可以用来过滤线程

- **Restore Layout**

重置debug所有界面

- **Settings**

点击该按钮将打开有关设置的列表:



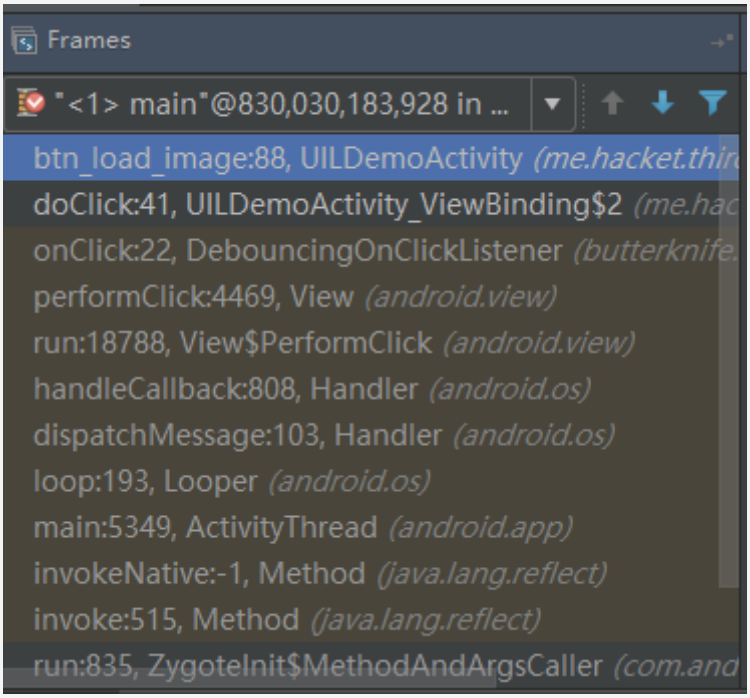
我们对其中的几个进行说明:

- Show Values Inline
  调试过程中开启该功能,将会代码右边显示变量值,即下图中红框所示部分:
- Auto-Variables Mode
  开启这个功能后,idea的Debugger会自动评估某些变量,大概就是当你执行在某个断点时,Debugger会检测当前调试点之前或者之后的变量的状态,然后在变量区选择性输出.举个例子来说明,未开启该功能之前,变量区输出所有的变量信息,开启之后,Debugger检测到某个变量在之后没有被使用,那么在变量区就不会输出该变量的信息。
- Sort values alphabetically
  开启这个功能的化,变量区中的输出内容会按照按字母顺序进行排序,很简单,不常用,还是按照默认的顺序好.

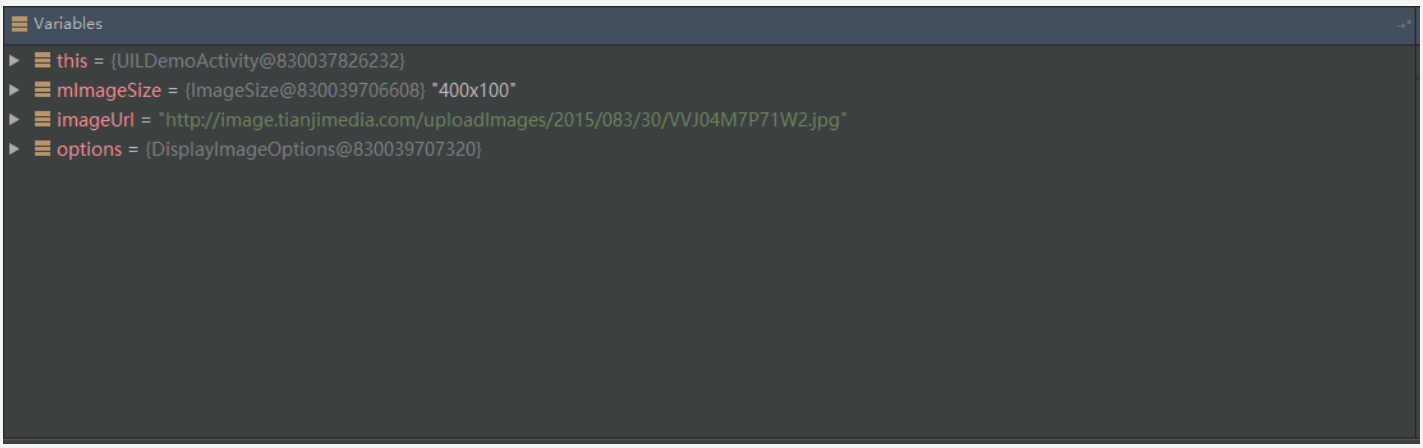
解析来我们来认识一下线程的类型,表示为不同的图标:

线程状态描述	图标
Thread is suspended.	
Thread is waiting on a monitor lock.	
Thread is running.	
Thread is executing network operation, and is waiting for data to be passed.	
Thread is idle.	
Event Dispatch Thread that is busy.	
Thread is executing disk operation.	

当前线程栈帧区



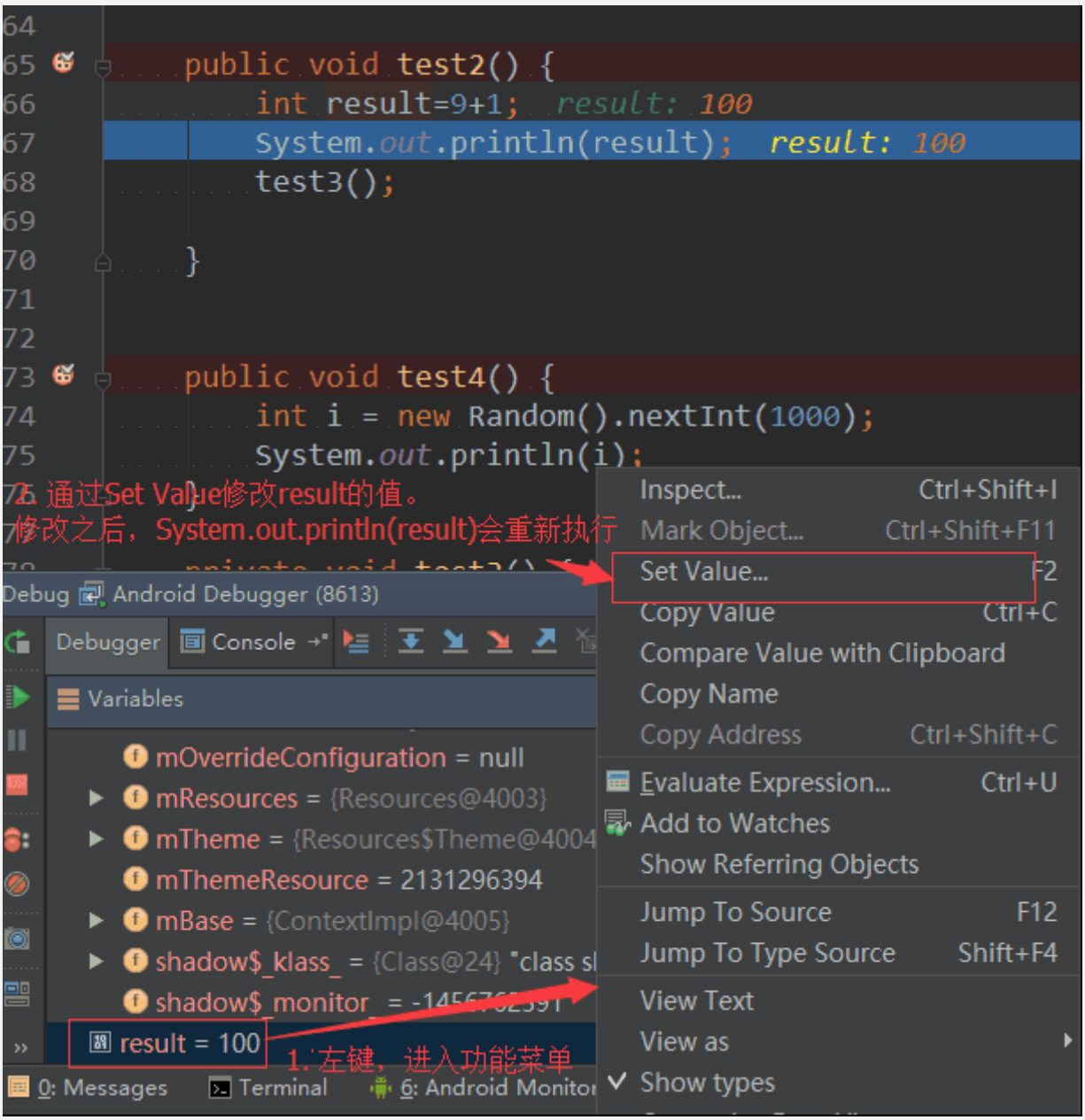
对象变量区(当前对象内所有属性值)



变量观察区

修改变量值

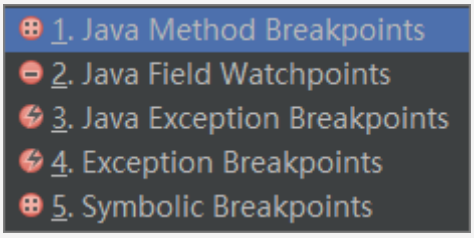
在调试过程中，我们可以方便的修改某个变量的值：



断点的分类

断点是调试器的功能之一，可以让程序暂停在需要的地方，帮助我们进行分析程序的运行过程。
在Android Studio中，断点又被以下五类：

1. 条件断点
2. 日志断点
3. 异常断点（Java Exception Breakpoints）
4. 方法断点（Java Method Breakpoints） 是我们最熟悉的断点类型
5. 属性断点（Java Field Breakpoints）

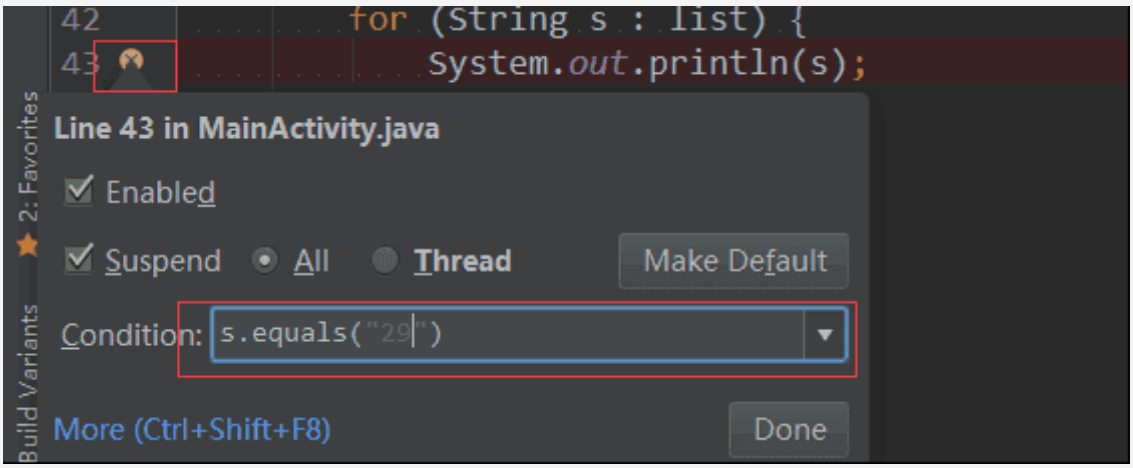


条件断点

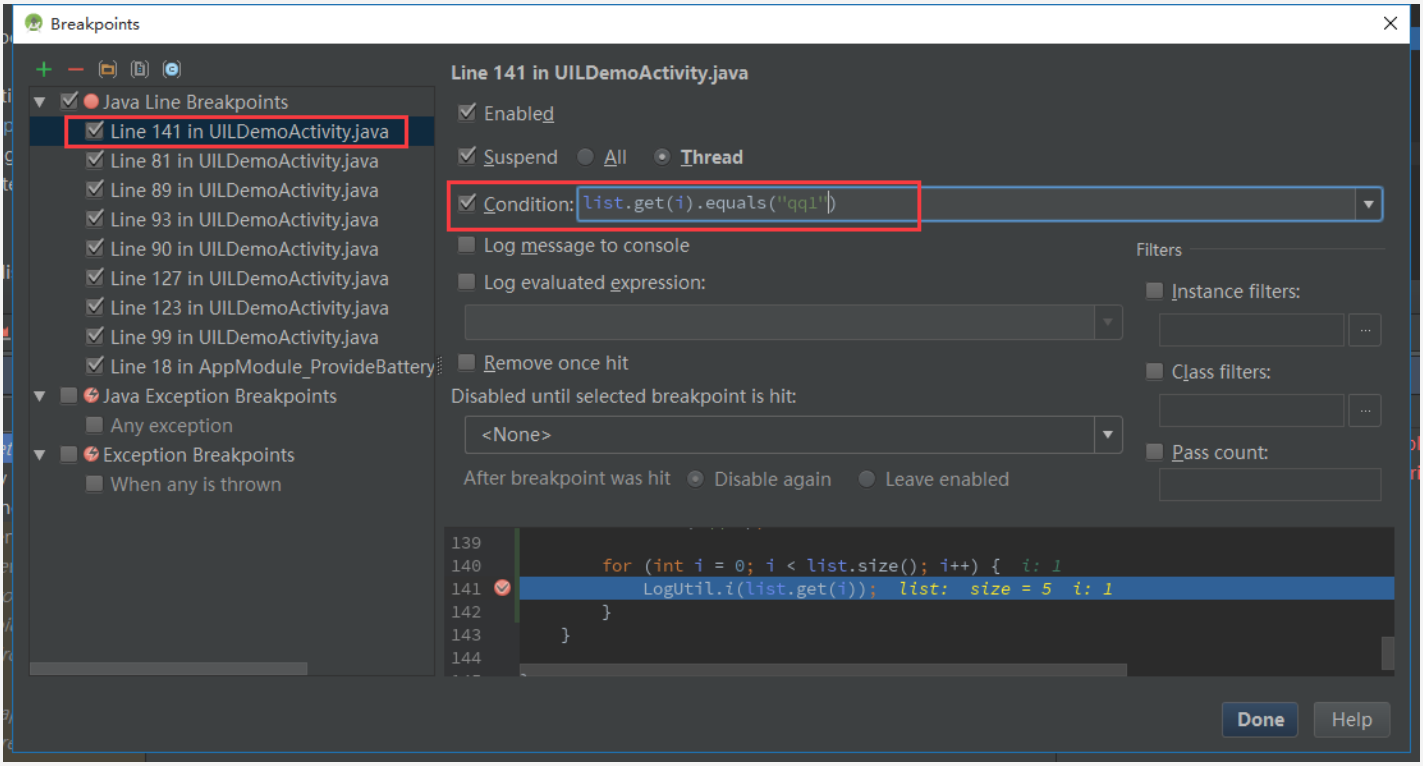
所谓的条件断点就是在特定条件发生的断点，也就是，我们可将某个断点设置为只对某种事件感兴趣，最典型的应用就是在列表循环中，我们希望在某特定的元素出现时暂停程序运行。比如，现在我们有个list中，其中包含了q，1q，2q,3q四个元素，我们希望在遍历到2q时暂停程序运行：

```
35      public void test() {
36          List<String> list = new ArrayList<>()
37          list.add("q");
38          list.add("1q");
39          list.add("2q");
40          list.add("3q");
41
42          for (String s : list) {
43              System.out.println(s);
44          }
45
46      }
```

断点处右键单击，在Condition处填写过滤条件.此处我们只关心2q，因此填写 `s.equals("2q")`



也可以在 View Breakpoints 编辑

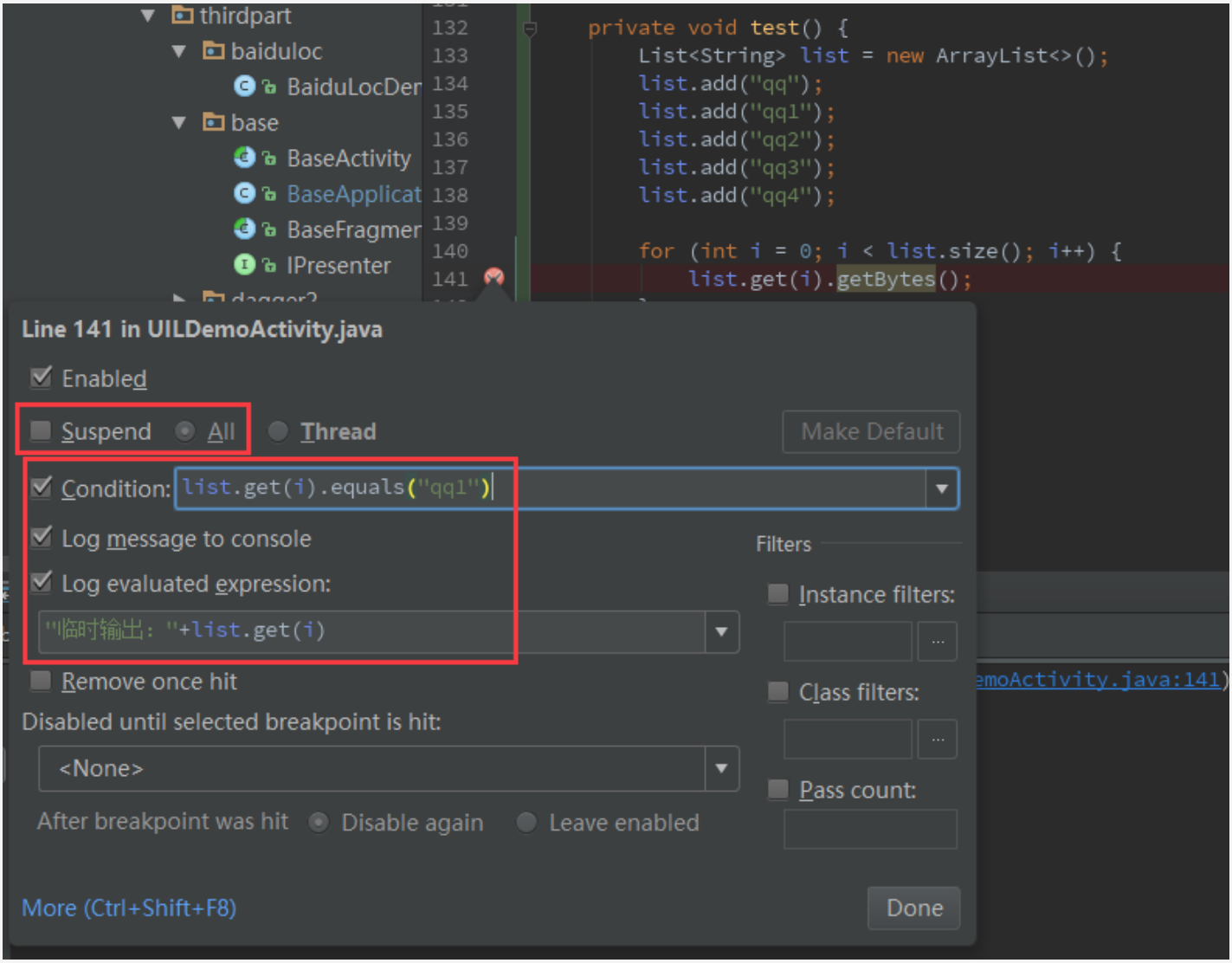


日志断点

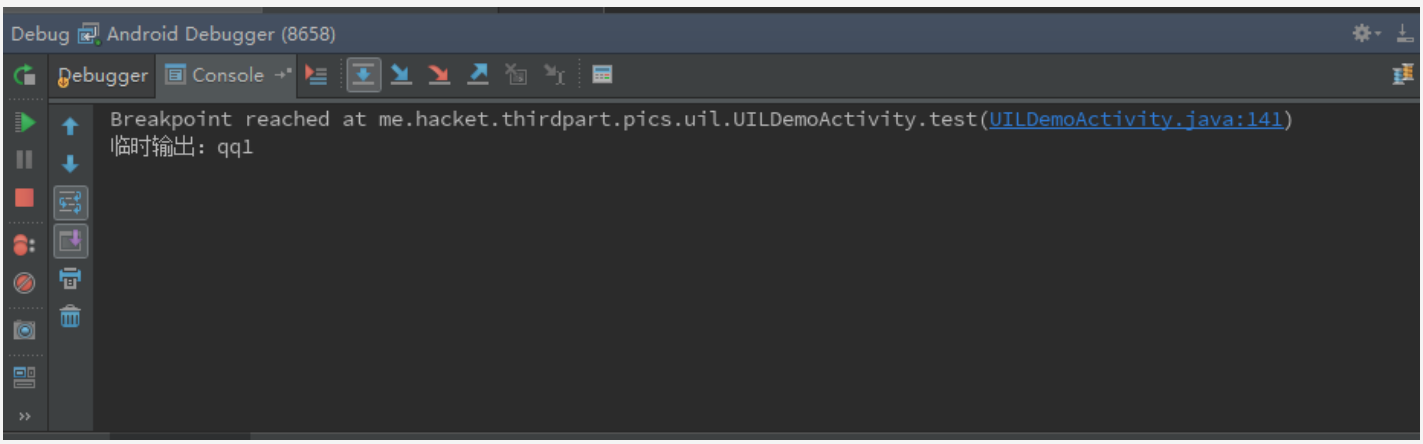
该类型的断点不会使程序停下来，而是在输出我们要它输出的日志信息，然后继续执行。

具体操作如下：

同样在断点处右键单击，在弹出的对话框中取消选中Suspend，在弹出的控制面板中，选中 Log message to console 和 Log evaluated expression ，然后再填写想要输出的日志信息，如下：



当调试过程遇到该断点将会输出结果，注意是在console输出，不是logcat

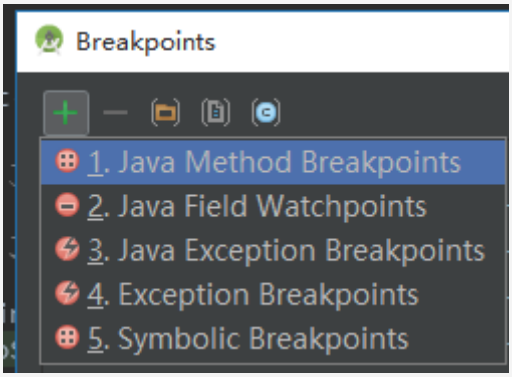


异常断点

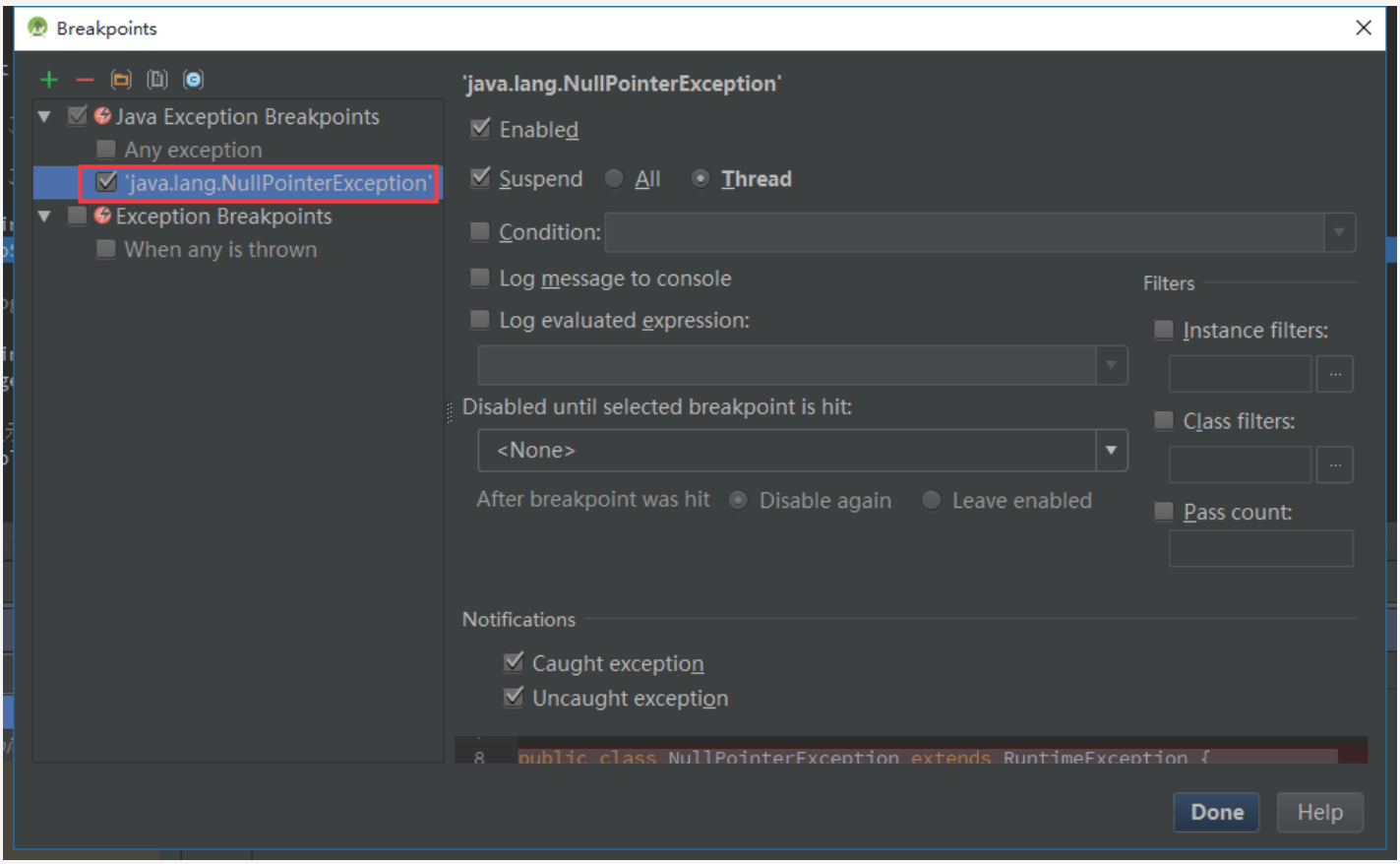
所谓的异常断点就是在调试过程中，一旦发生异常（可以指定某类异常），则会立刻定位到异常抛出的地方。比如在调试异常中，我们非常关注运行时异常，希望在产生任何运行异常时及时定位，那么此时就可以利用该类型异常，在上线之前，进行异常断点调试非常有利于减少正式环境中发生crash的几率。

具体操作如下：在Run菜单项中，选择 `View Breakpoints` （也可以在断点管理面板中点击）。

在管理断点面板中，点击**+**，在弹出的下拉选择列表中，我们选择Java Exception Breakpoints



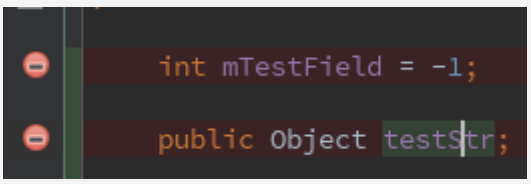
这里我们选中Search By Name,在下面的输入框中输入我们所关心的异常类型。此处我们关心NullPointerException，在调试过程一旦发生NullPointerException，调试器就会定位到异常发生处。



Note:不需要在这里打断点，如果出现了NPE在debug环境会自动停在这里。

属性断点 无效?

Filed WatchPoint是本质上是一种特殊的断点，也称为属性断点：当我们某个字段值被修改的时候，程序暂停在修改处。通常在调试多线程时尤为可用，能帮我们及时的定位并发错误的问题。其使用和添加普通的断点并无不同，断点图标稍有不同。比如你是要在变量访问的时候停下来还是在变量改变的时候停下来。
使用步骤：在属性处，打个断点



Reference

你所不知道的Android Studio调试技巧
http://www.jianshu.com/p/011eb88f4e0d