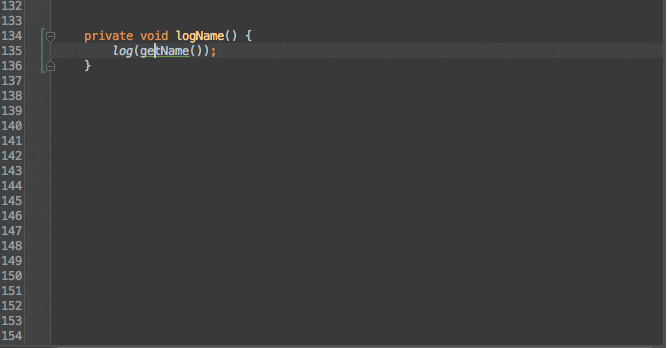
**50. 分析传入数据流（Analyze data flow to here）**

* **描述：**这个操作将会根据当前选中的变量、参数或者字段，分析出其传递到此处的路径。 当你进入某段陌生的代码，试图明白某个参数是怎么传递到此处的时候，这是一个非常有用的操作。
* **调用：**Menu → Analyze → Analyze Data Flow to Here
* **快捷键：**无，可以在设置中指定。
* **相反的操作：**分析传出数据流（Analyze data flow from here），这个将会分析当前选中的变量往下传递的路径，直到结束。



**51. 堆栈追踪分析（Analyze Stacktrace）**

* **描述：** 这个操作读取一份堆栈追踪信息，并且使它像logcat中那样可以点击。当你从bug报告中或者终端复制了一份堆栈追踪，使用该操作可以很方便地调试。
* **调用：**Menu → Analyze → Analyze Stacktrace
* **快捷键：**无，可以在设置中指定。
* **更多：**通过使用“ProGuard Unscramble Plugin”插件，也可以分析混淆过的堆栈追踪。

**52. 关联调试程序（Attach Debugger）**

* **描述：**随时启动调试程序，即使你没有以调试模式启动你的应用。这是一个很方便的操作，因为你不必为了调试程序而以调试模式重新部署你的应用。当别人正在测试应用，突然遇到一个bug而将设备交给你时，你也可以很快地进入调试模式。
* **调用：**点击工具栏图标或者Menu → Build → Attach to Android Process
* **快捷键：**无，可以在设置中指定，或者点击工具栏对应的图标。

**53. 条件断点（Conditional Breakpoints）**

* **描述：**简单说，就是当设定的条件满足时，才会触发断点。你可以基于当前范围输入一个java布尔表达式，并且条件输入框内是支持代码补全的。
* **调用：**右键需要填写表达式的断点，然后输入布尔表达式。

**54. 禁用断点（Disable Breakpoints）**

* 这个操作将使得断点。当你有一个设置过复杂条件的断点或者是日志断点，当前不需要，但是下次又不用重新创建，该操作是很方便的。
* **调用：**按住Alt，然后单击断点即可。

**55. 计算表达式（Evaluate Expression）**

* **描述：**这个操作可以用来查看变量的内容并且计算几乎任何有效的java表达式。需要注意的是，如果你修改了变量的状态，这个状态在你恢复代码执行后依然会保留。
* **快捷键：**处在断点状态时，光标放在变量处，按Alt + F8，即可显示计算表达式对话框。

**56. 审查变量（Inspect Variable）**

* **描述：**该操作可以在不打开计算表达式对话框就能审查表达式的值。
* **快捷键：**调试状态下，按住Alt键，然后单击表达式即可。

**57. 日志断点（Logging Breakpoints）**

* **描述：**这是一种打印日志而不是暂停的断点，当你想打印一些日志信息但是不想添加log代码后重新部署项目，这是一个非常有用的操作。
* **调用：**在断点上右键，取消Suspend的勾选，然后勾选上Log evaluated Expression，并在输入框中输入你要打印的日志信息。

**58. 标记对象（Mark Object）**

* **描述：**当你在调试的时候，这个操作可以让你给某个特殊的对象添加一个标签，方便你后面很快地辨认。在调试时，当你从一堆相似的对象中查看某个对象是否和之前是一样的，这就是一个非常有用的操作。
* **调用：**右键你需要标记的对象，选中Mark Object，输入标签；
* **快捷键：**选中对象时，按F3(OS X)、F11(Windows/Linux)；

**59. 显示当前运行点（Show Execution Point）**

* **描述：**该操作会立刻把你的光标移回到当前debug处。

通常的情况是： 1. 你在某处触发了断点 2. 然后在文件中随意浏览 3. 直接调用这个快捷键，快速返回之前逐步调试的地方。

* **快捷键：**（Debug时) Alt + F10；

**60. 终止进程（Stop Process）**

* **描述：**该操作会终止当前正在运行的任务。如果任务数量大于一，则显示一个列表供你选择。在终止调试或者中止编译的时候特别有用！
* **快捷键：**Cmd + F2(OS X)、Ctrl + F2（Windows、Linux）；

**61. 临时断点（Temporary Breakpoints）**

* **描述：**通过该操作可以添加一个断点，这个断点会在第一次被命中的时候自动移除。
* **快捷键：**Alt + 鼠标左键 点击代码左侧（鼠标）、Cmd + Alt + Shift + F8(OS X)、Ctrl + Alt + Shift + F8(Windows/Linux)

**62. 调用层级树弹窗（The Call Hierarchy Popup）**

* **描述：**该操作会给你展示 在一个方法的声明和调用之间所有可能的路径。
* **快捷键：**Ctrl + Alt + H

