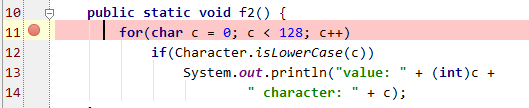
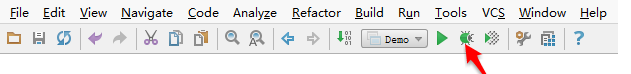
先编译好要调试的程序。

### 1.设置断点

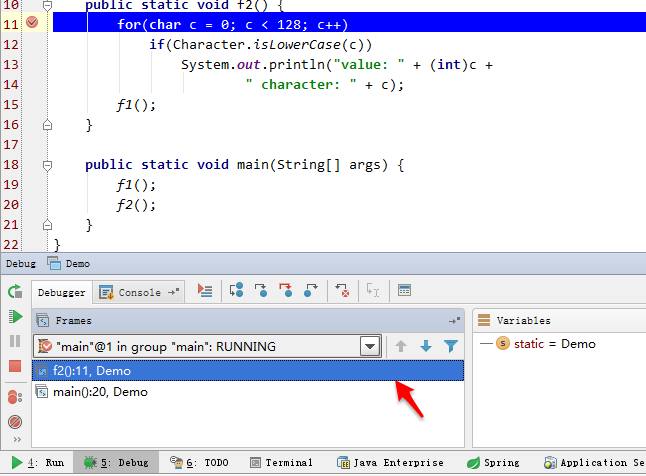


选定要设置断点的代码行，在行号的区域后面单击鼠标左键即可。

### 2.开启调试会话



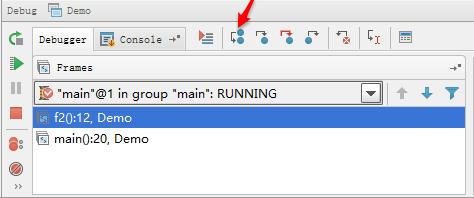
点击红色箭头指向的小虫子，开始进入调试。



IDE下方出现Debug视图，红色的箭头指向的是现在调试程序停留的代码行，方法f2()中，程序的第11行。红色箭头悬停的区域是程序的方法调用栈区。在这个区域中显示了程序执行到断点处所调用过的所用方法，越下面的方法被调用的越早。

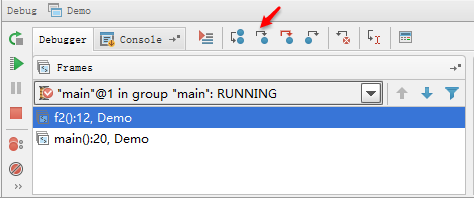
### 3.单步调试

**3.1 step over**



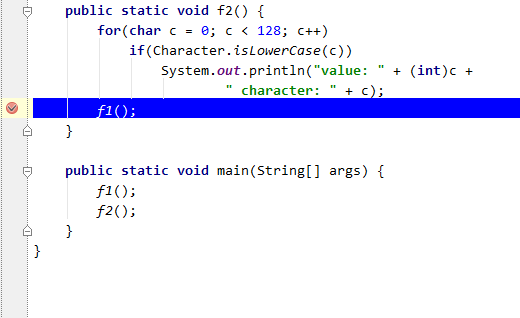
点击红色箭头指向的按钮，程序向下执行一行（如果当前行有方法调用，这个方法将被执行完毕返回，然后到下一行）

#### 3.2 step into

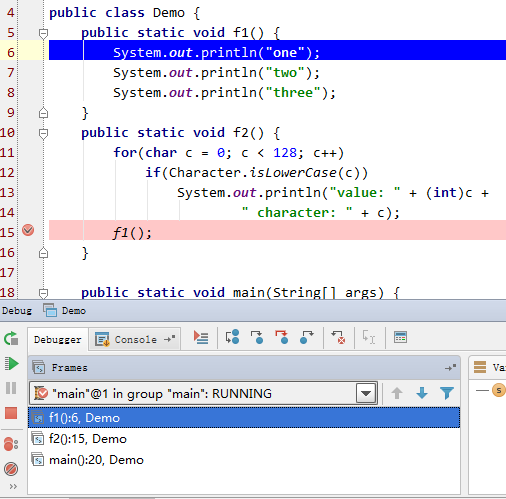


点击红色箭头指向的按钮，程序向下执行一行。如果该行有自定义方法，则运行进入自定义方法（不会进入官方类库的方法）。具体步骤如下：

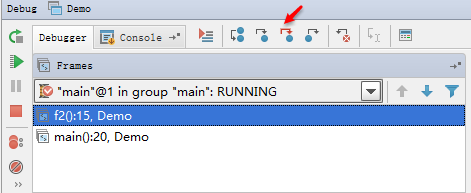
在自定义方法发f1()处设置断点，执行调试



点击IMG_262

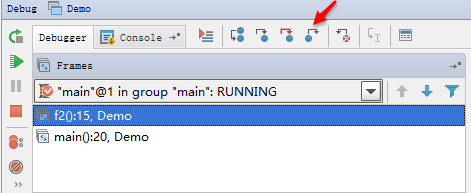


#### 3.3 Force step into



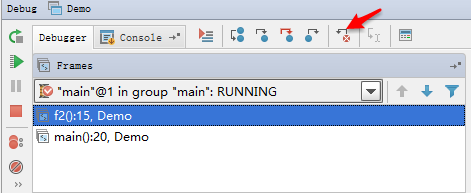
该按钮在调试的时候能进入任何方法。

#### 3.4 step out



如果在调试的时候你进入了一个方法(如f2())，并觉得该方法没有问题，你就可以使用stepout跳出该方法，返回到该方法被调用处的下一行语句。值得注意的是，该方法已执行完毕。

**3.5 Drop frame**

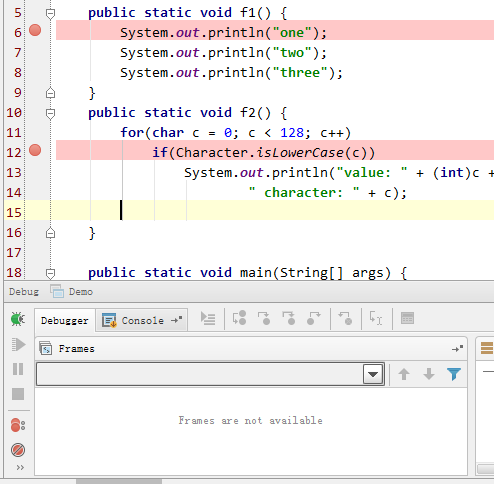


点击该按钮后，你将返回到当前方法的调用处（如上图，程序会回到main()中）重新执行，并且所有上下文变量的值也回到那个时候。只要调用链中还有上级方法，可以跳到其中的任何一个方法。

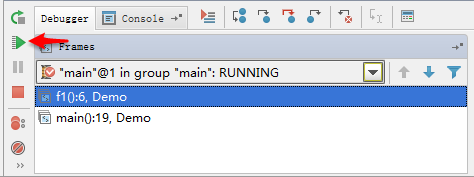
**4. 高级调试**

#### 4.1 跨断点调试

设置多个断点，开启调试。

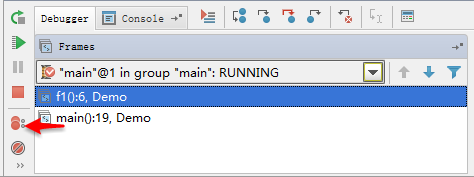


想移动到下一个断点，点击如下图：

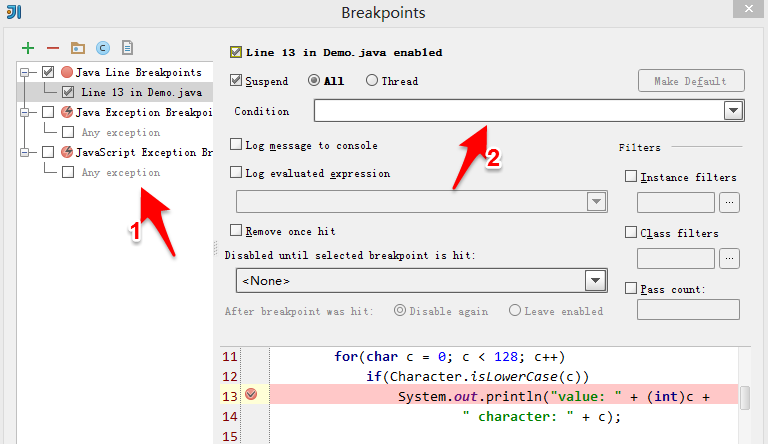


程序将运行一个断点到下一个断点之间需要执行的代码。如果后面代码没有断点，再次点击该按钮将会执行完程序。

**4.2 查看断点**

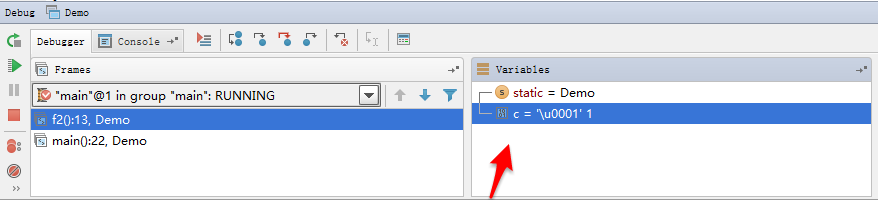


点击箭头指向的按钮，可以查看你曾经设置过的断点并可设置断点的一些属性。



箭头1指向的是你曾经设置过的断点，箭头2可以设置条件断点（满足某个条件的时候，暂停程序的执行，如 c==97）。结束调试后，应该在箭头1处把所设的断点删除(选择要删除的断点后，点击上方的红色减号）。

**4.3 设置变量值**



调试开始后，在红箭头指向的区域可以给指定的变量赋值（鼠标左键选择变量，右键弹出菜单选择setValue...）。这个功能可以更加快速的检测你的条件语句和循环语句。