# 第3讲 多线程程序设计技术

**教学与实践目的**：**学会在网络应用开发中运用Java多线程技术。**

## 一、IDE平台程序的基本调试技术

程序无语法错误、能运行，但没有出现预期的结果，程序可能存在逻辑错误，解决这类错误的主要方法是查看程序运行过程中的内存变量值。一个常用的手段是通过打印语句打印出变量的值，例如使用System.out.println(待排查的变量)。但更强大的方法是使用IDE提供的断点功能。

**在idea设断点并查看变量的方法**：

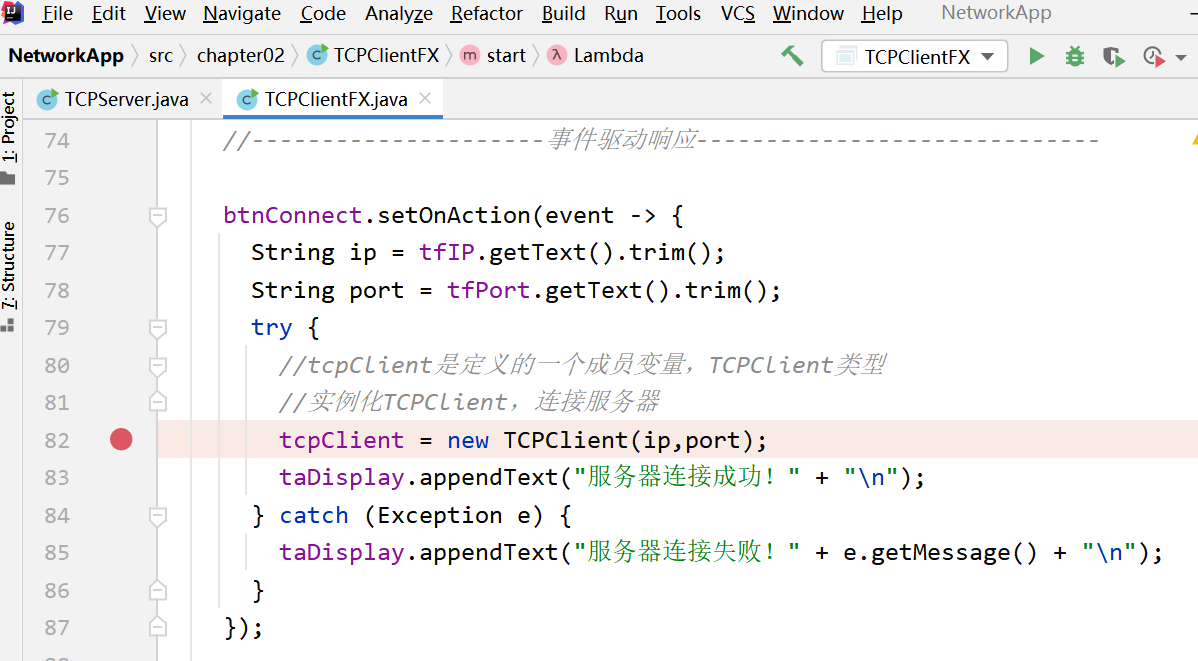
鼠标点击要查看变量所在代码行的行号右侧空白处，出现棕红色实心圆，即表示在此处打了断点，调试时程序会在此处停住，方便观察程序运行的状况和各变量的即时值。

首先新建一个包，命名为chapter03，然后将上一讲的TCPServer.java、TCPClient.java、TCPClientFX.java复制到这个包中，注意程序中第一行语句都要修改为package chapter03；

然后，在客户端窗口程序TCPClientFX中选择一行有变量的代码行(如图3.1，要查看获取的IP地址是否符合预期)，鼠标点击行号右侧标注断点；

右上角下拉框选中“TCPClientFX”，再点击“调试”图标，窗口程序运行到红色断点行时会停留，便于观察此时IP、port等变量的状态值，如图3.2所示。通过图3-2所示红色框区域，可以让程序单步执行，一步一步地观察程序执行的情况，如果当前行代码中有方法的调用，step over表示跳过方法之间运行下一行代码，而step into则继续下钻，可以进入方法内部，一般只是用于进入自定义方法。

如果要调试的代码行是在一个新的线程的代码块中，则需要右键点击断点图标，在弹出菜单中，将"Suspend:" 后面的单选按钮从"**All**" 改为 "**Thread**"。



**调试**

图3.1 调试断点行设置

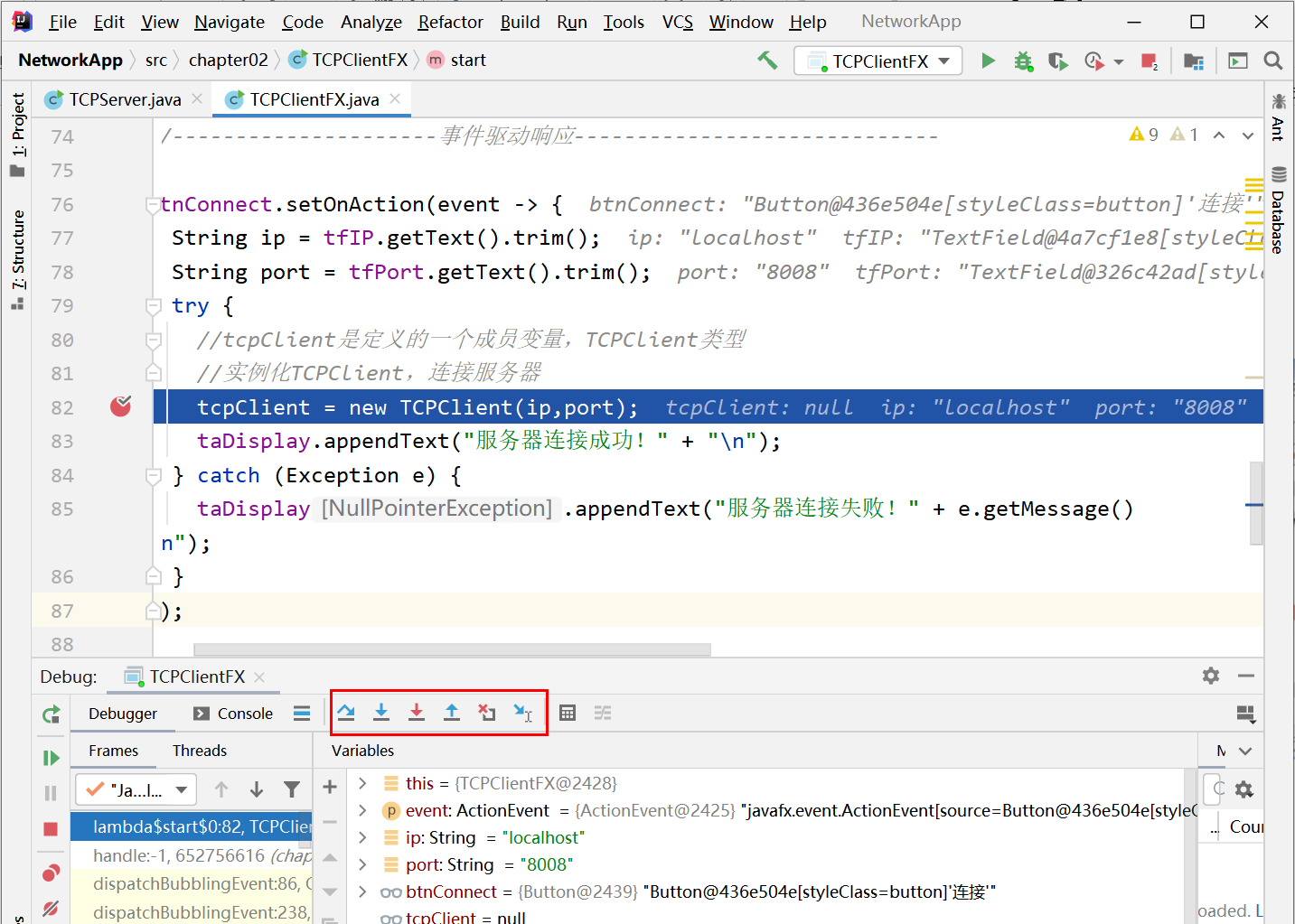


图3.2 调试暂停界面

## 二、理解阻塞语句

在同一个进程中，一条阻塞语句的执行影响着下条语句何时被执行。如果该条语句没有执行完，那么下条语句是不可能进入执行状态的，因此，从字面层上理解，该条语句阻塞了下面语句的执行。

JAVA类BufferedReader中readLine( )方法的调用是阻塞语句，若该套接字的输入流中没有带行结束符（如\n）的字符可读，则该语句会处于阻塞状态，直到条件出现行结束符，才会执行下面的语句。

**阻塞状态程序演示：**

（1）将TCPServer.java程序中的发送语句临时禁用（验证完再还原），如：

*//向输出流中输出一行字符串,远程客户端可以读取该字符串*

*//pw.println("来自服务器：" + msg); 临时禁用*

即服务器不回传信息；

（2）启动TCPServer.java服务程序，再启动TCPClientFX.java客户端程序，发送信息，发现客户程序不能正常运行，发送按钮甚至整个程序失去响应。

（3）强行终止TCPClientFX，在窗口程序的发送语句处设置断点，如图3.3所示。然后在调试状态运行该程序，逐行调试（遇到自定义的方法，建议使用step into跟踪进入）。在执行到receive()方法时，使用step into跟踪进方法会发现程序会阻塞在msg = br.readLine(); 处（因为服务器没有返回，客户端的输入流队列中是空的，所以被阻塞）。所以程序设计时一定要小心。

图3.3 断点位置

## 三、理解读一行功能

同理，若套接字的输入流中有多行信息，调用一次readLine()方法，只是读出当前的一行（当然你可以调用其他的“读”方法）。

**程序演示**：

（1）在TCPServer.java程序中多增加一条信息返回语句，如：

pw.println("来自服务器：" + msg);

*//下面多增加一条信息返回语句*

pw.println("来自服务器，重复发送: " + msg);

然后启动服务端程序；

（2）启动客户端TCPClientFX程序，发现客户显示区每次只显示一条信息，且与你发送的信息不同步。因为每一次互动，服务器返回两行信息，而客户端只是读取最前面的一行信息。

**如何解决阻塞和多行信息的读写问题? 一个常用的解决方案就是多线程。**

## **四、多线程技术**

多线程程序的执行如图3.4所示。

|  |  |  |
| --- | --- | --- |
| http://192.168.221.19:8080/network-programming/CH03ThreadClient.files/clip_image003.gif |  | http://192.168.221.19:8080/network-programming/CH03ThreadClient.files/clip_image004.gif |

图3.4 程序调用的顺序执行与线程调用的并行执行

有了多线程技术，我们就有了更多选择。

### 1. 编写读取服务器信息的线程

在TCPClientFX.java程序中，发送信息是可以通过“发送”按钮来实现主动控制，可接收信息是被动的，你不知道输入流中有多少信息。

为此，在窗口程序中添加一个线程专门负责读取输入流中的信息， 同时，“发送”按钮动作中，读取输入流信息的代码就需要删除。

现在右键选择TCPClientFX.java重构，重命名为TCPClientThreadFX.java（采用如图3.5所示的方式），

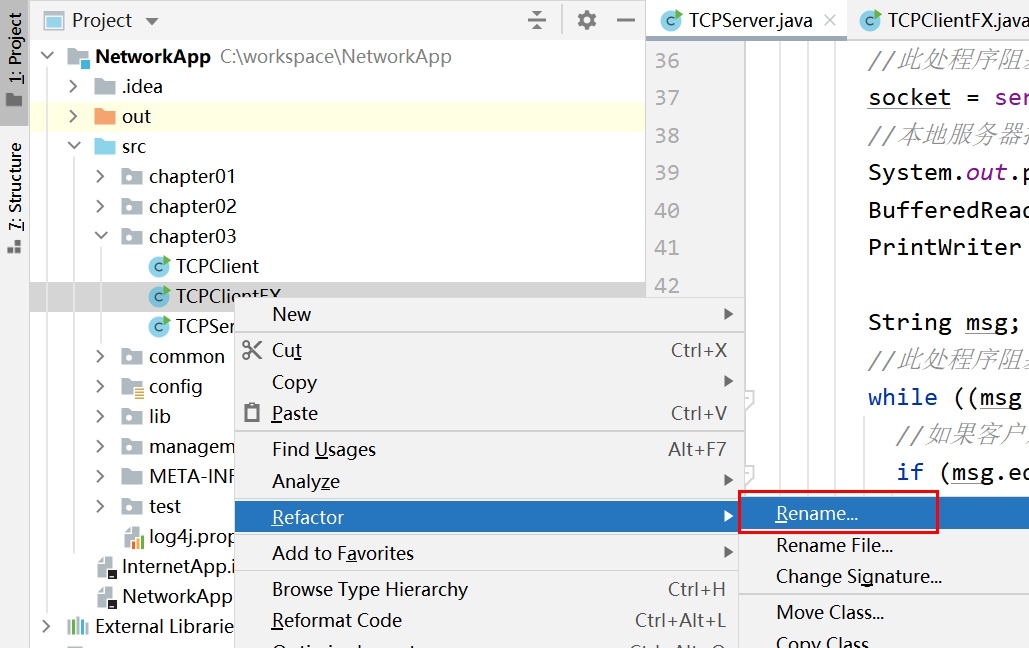


图3.5 重构TCPClientFX.java

并在合适的位置（例如，btnConnect的动作事件代码中，在连接服务器成功，接收了服务器第一条欢迎信息之后，添加第5行之后的代码）编写如下线程代码，用于接收服务器的信息，为了简洁，匿名内部类使用了lambda的写法：

public class TCPClientFX extends Application {

Thread readThread; *//定义成员变量，读取服务器信息的线程*

*//…… 省略……*

*//以下代码位于btnConnect.setOnActon方法中的合适位置*

*//用于接收服务器信息的单独线程*

readThread = new Thread(()->{  
 St**ri**ng msg = null;  
 *//不知道服务器有多少回传信息，就持续不断接收  
 //由于在另外一个线程，不会阻塞主线程的正常运行* while ((msg = tcpClient.receive()) != null) {  
 *//lambda表达式不能直接访问外部非final类型局部变量  
 //所以这里使用了一个临时变量* String msgTemp = msg;  
 Platform.*runLater*(()->{  
 taDisplay.appendText( msgTemp + "\n");  
 });  
 }

*//跳出了循环，说明服务器已关闭，读取为null，提示对话关闭*  
 Platform.*runLater*(()->{  
 taDisplay.appendText("对话已关闭！\n" );  
 });  
});

readThread.start(); *//启动线程*

*…… 省略……*

**以上代码中有三点注意**：

（1）由于是新开的一个线程循环读取服务器的信息，所以不用考虑服务器是否有发欢迎信息，就算读取不到信息也只是阻塞这个线程，主程序本身使用没有任何影响（单线程就会卡住）。事实上服务器发多少信息都没问题，该线程通过循环语句来读取，没信息过来就阻塞等待，当服务器关闭连接时，就会跳出循环语句，结束本线程；

（2）对于JavaFX窗体界面，在新线程中无法直接更新界面中有关控件的内容，只能将更新代码放在PlatForm.runLater(Runnable XXX)方法的Runnable子类实例中，如以上代码第15-17行所示；

（3）匿名内部类或lambda表达式中，不能访问外部类方法中的非final类型的局部变量，例如上面第16行代码， 如果直接使用taDisplay.appendText( msg + "\n");就会报错，所以代码第14行使用了个临时变量来解决这个问题（当然，如果msg是定义在类中的成员变量，就没有这个限制）

### 2. 程序退出部分思考

由于“退出”按钮和关闭窗体的事件响应都需要调用这部分代码，所以将之封装为exit()方法：

private void exit() {  
 if(tcpClient != null){  
 tcpClient.send("bye"); *//向服务器发送关闭连接的约定信息*  
 tcpClient.close();  
 }  
 System.*exit*(0);  
}

先成功连接服务器，在正常发送一些信息后，不通过按钮发送信息输入区的bye告知服务器，这时候直接点击退出，很大概率会抛出异常信息后结束程序，其实这不算问题，交互还是可以正常完成。如果你的程序出现了这种情况，请思考抛出异常的原因？能否提供一个方案解决抛出异常的问题。

## 五、制作登录查看平时成绩程序

专门新建一个查看平时成绩的Java包，命名为lookupscore，在该包下面创建2个程序：LookUpScore.java和LookUpScoreFX.java。

LookUpScore.java同TCPClient.java；

LookUpScoreFX.java同TCPClientThreadFX.java，只需要做一点修改：修改窗体title为“登录查成绩”；

以上操作可通过idea的重构来完成，即在idea的Project树形列表中将原始的两个java源文件拷贝到新包中，再使用Refactor->Rename来完成

成绩查看地址为（IP：172.16.229.253 端口：9009）。

**登录查看成绩方法**：运行你的LookUpScoreFX.java程序，在“信息输入区”输入你注册的学号、姓名和密码，中间用“&”连接（如：20180000111&程旭元&密码，注意不要带上空白符）。“信息显示区”查看成绩成功的同时（成功是指看到了你的成绩记录），即表示今天任务完成，并记录了你**本次课的平时成绩**（注意：不要用你的机器IP查看别人的成绩）。

## 六、特别提醒

**从今后的周开始，每次课堂的前5~8分钟为查看成绩的时间，查看成绩的同时也记录了你登录签到的信息。查成绩的IP记录和课堂提交作业的IP记录须一致（即查看成绩和提交作业的操作必须在同一台机器），才会记录本周的平时成绩。**