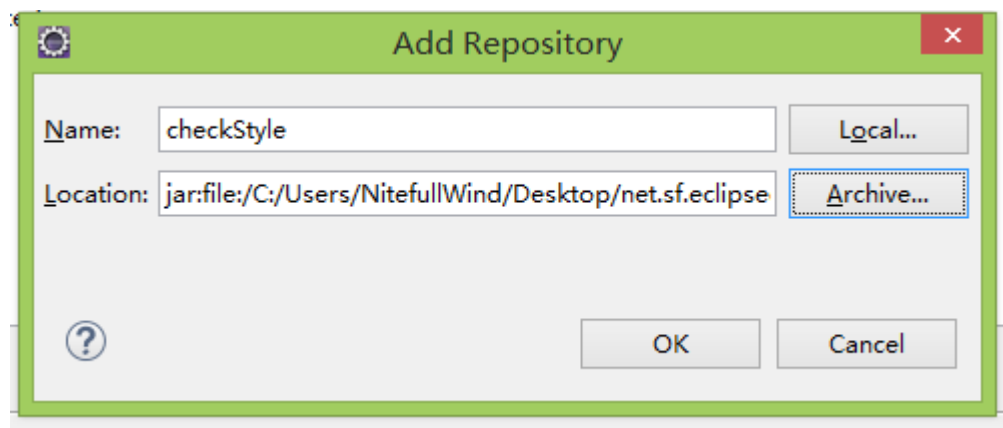


CheckStyle 使用心得

1. 安装：

我先使用在线安装的方式，安装未成功，网上查到错误的原因是版本不符。因此我又使用了其他方法安装：从网上下载 Eclipse CheckStyle Plug 的压缩包，打开 Eclipse→Help→Install New Software→add，Name 栏填 checkStyle，再点 Location 栏后的 Archive，选择刚刚下载的安装包。

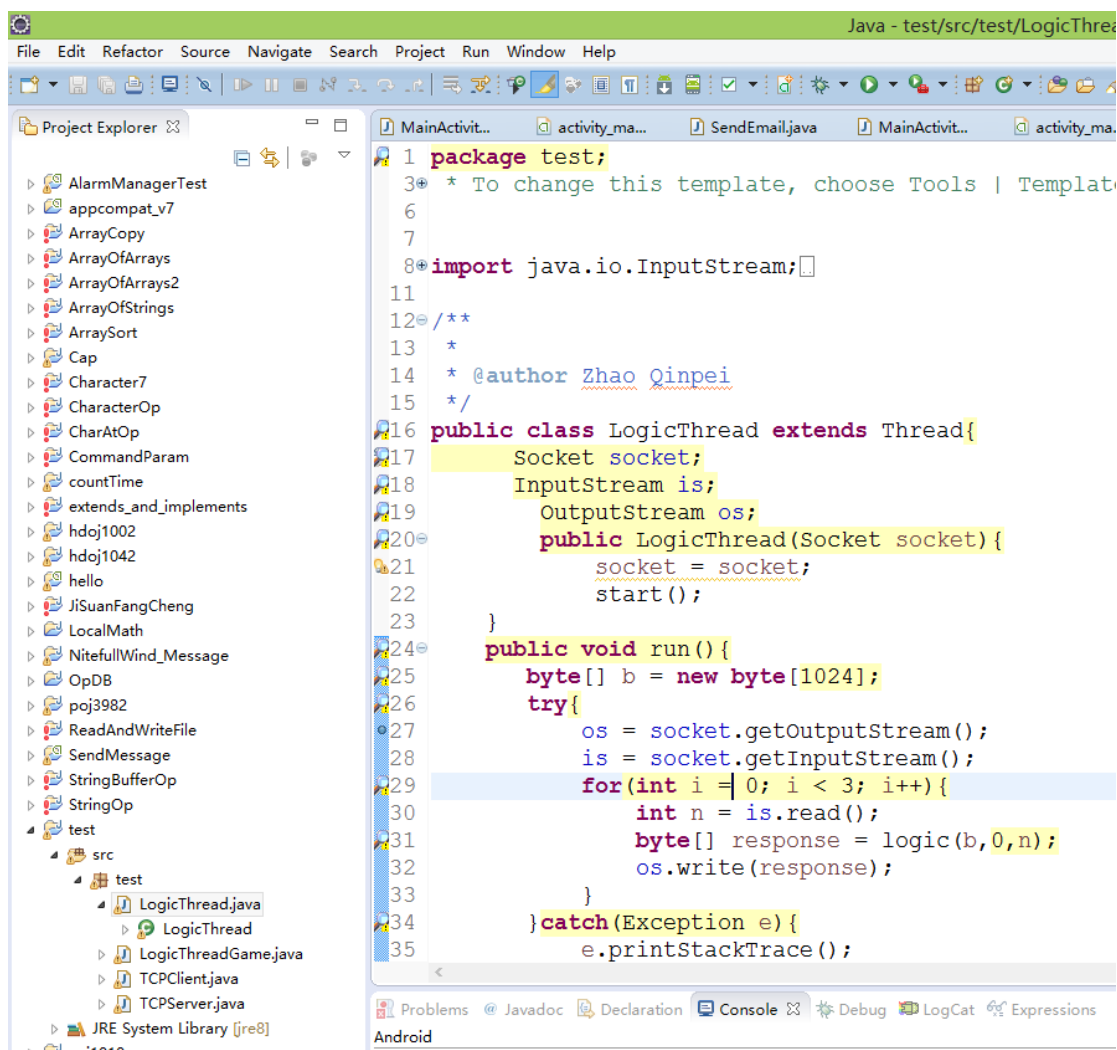



点击 OK，随后即可正常安装。

2. 使用：

安装完成后，重启 Eclipse 即可使用。在要测试的项目上，右键选择 CheckStyle→Check Code With CheckStyle


经过一系列工作，CheckStyle 就会找到代码中不规范的代码，并用符号标出，我们我可打开任一个源代码查看：



其中用图标所标注的行，就是被检测出有问题的代码行。可以点击任一图标查看错误详情。还可以在 Problems 栏查看所有的问题列表：

Problems @ Javadoc Declaration Console Debug LogCat Expressions					
28 errors, 211 warnings, 0 others (Filter matched 128 of 239 items)					
Description	Resource	Path	Location	Type	
Errors (28 items)					
Warnings (100 of 211 items)					
' ' is not followed by whitespace.	LogicThread...	/test/src/test	line 31	Checkstyle P...	
' ' is not followed by whitespace.	LogicThread...	/test/src/test	line 31	Checkstyle P...	
' ' is not followed by whitespace.	LogicThread...	/test/src/test	line 46	Checkstyle P...	
' ' is not followed by whitespace.	LogicThread...	/test/src/test	line 73	Checkstyle P...	

任一找一个警告信息，如：

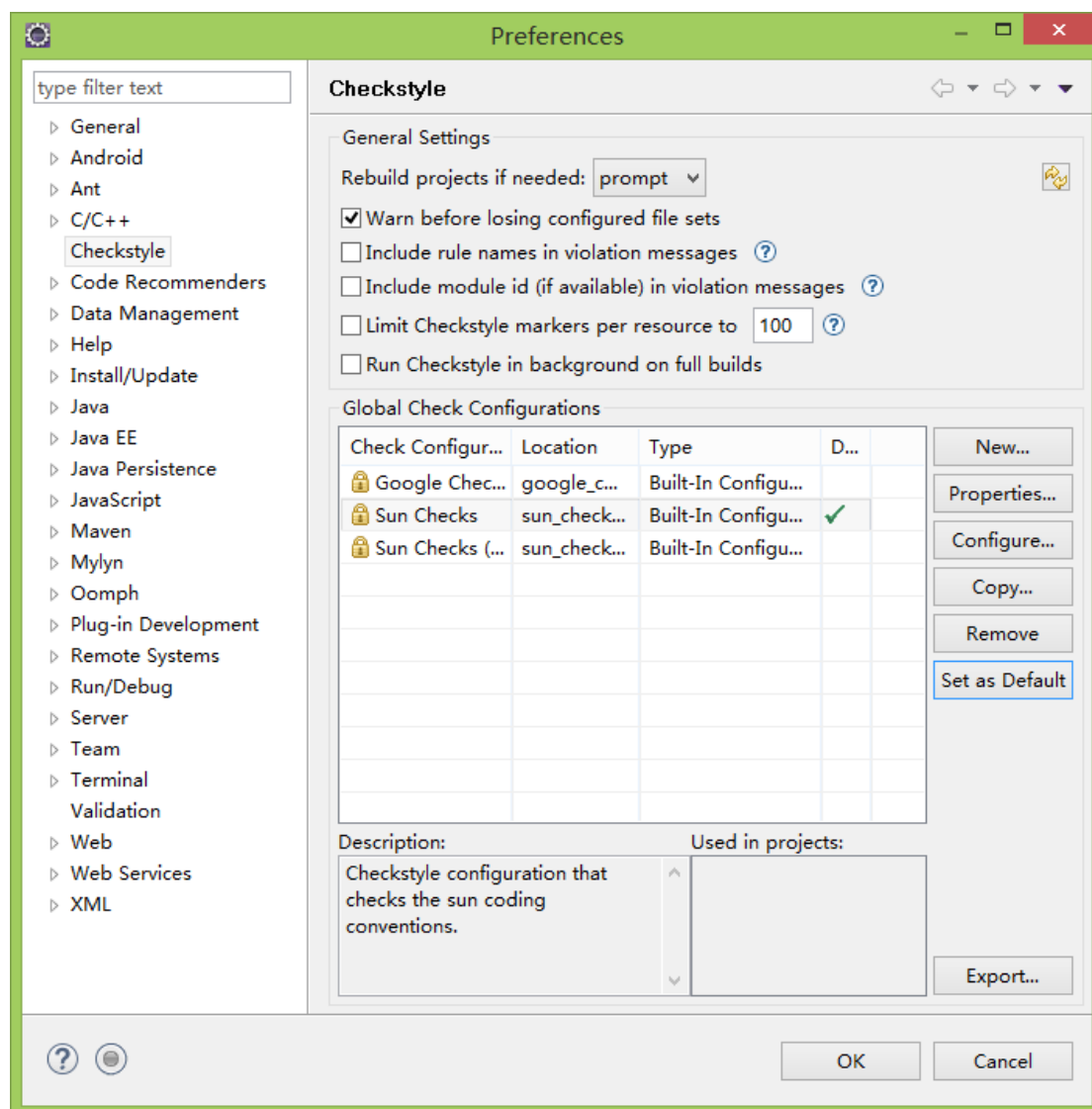
 16 public class LogicThread extends Thread{

警告说大括号前没有空格，我加上空格保存，再次检查，警告信息已消失：

```
16 public class LogicThread extends Thread {
```

3. 配置：

打开 Window→Preferences→CheckStyle




这里是 CheckStyle 的常用配置，包括配置检查规则，自己可以新建规则或者使用默认的规则，这里我使用了 Sun Checks 规则。

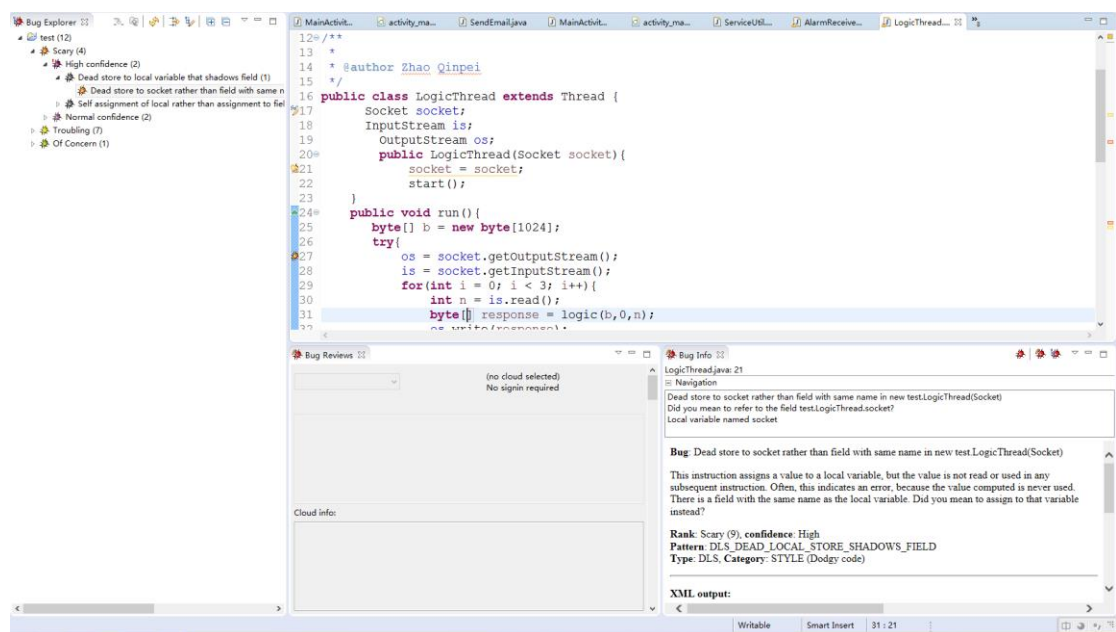
FindBugs 使用心得

1. 安装：

使用在线安装即可，很方便，中途没出现问题。

2. 使用：

安装完成后重启即可使用。在需要检查的项目上右键选择 Find Bugs→Find Bugs，经过一段时间后，程序就会使用  标示出代码中可能有 Bug 的代码。我们也可以在 Bug Explorer 中查看更详细的 Bug 说明：



这里将 bug 按威胁度、确信度等进行分类，方便我们修改 Bug。

双击任意一个 Bug 即可定位到出现问题的那一行，修改即可。

3. 配置：

可以在 Window→Preference→Java→FindBugs 对该软件进行常用的设置。

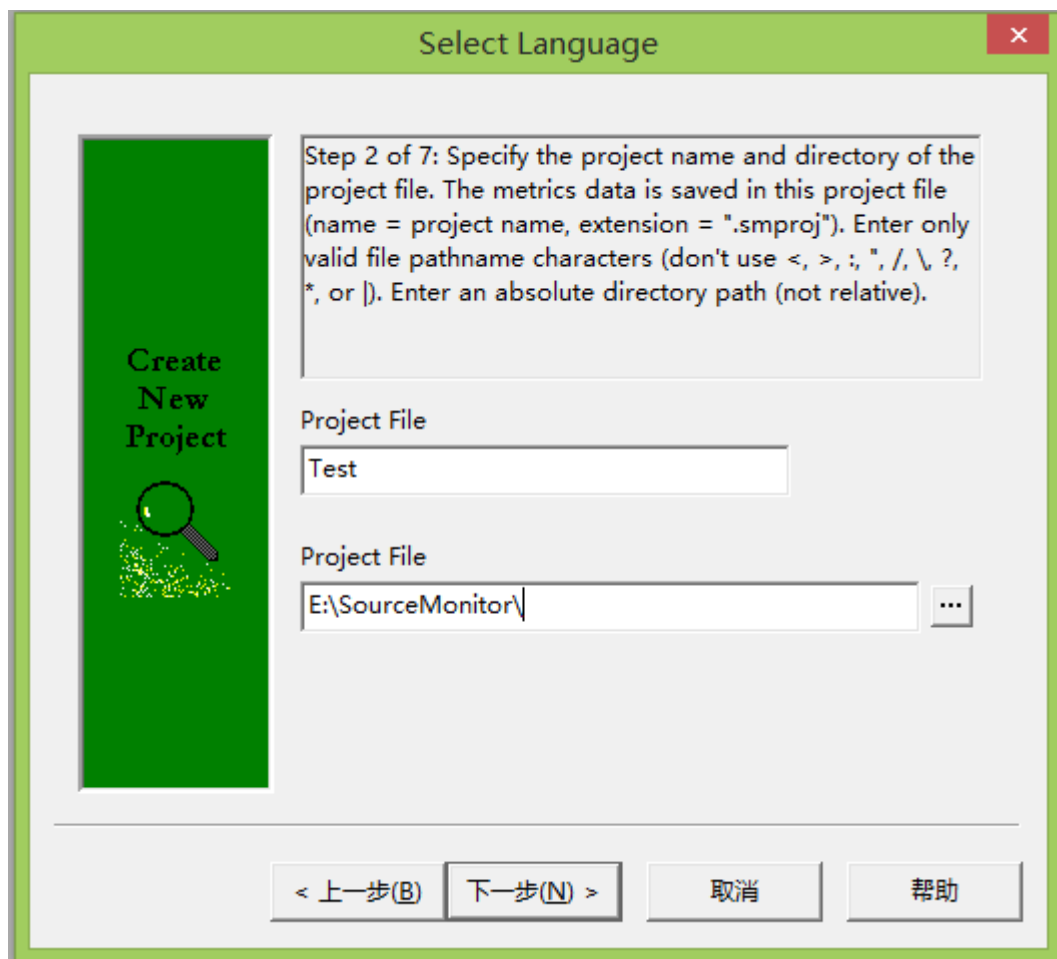
SourceMonitor 使用心得

1. 安装：

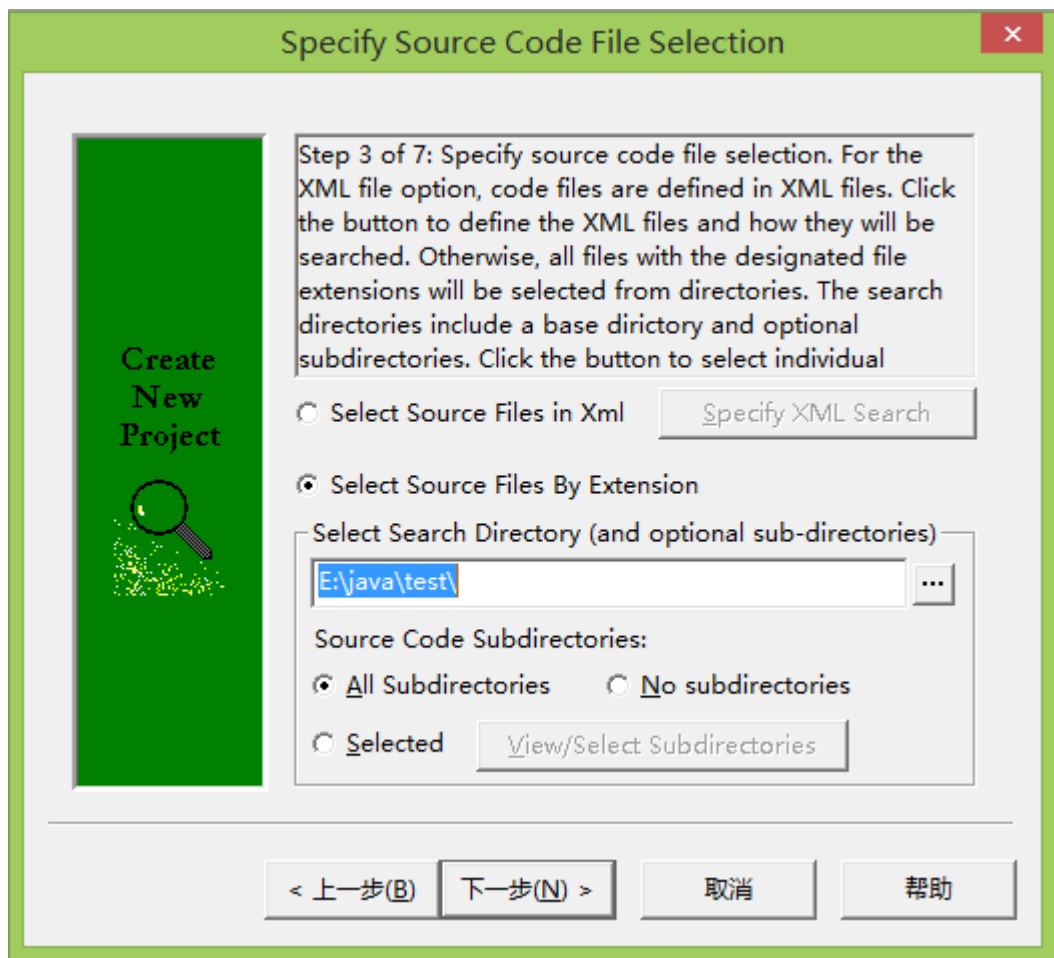
网上下载 SourceMonitor.exe 后打开按照引导安装即可。

2. 使用：

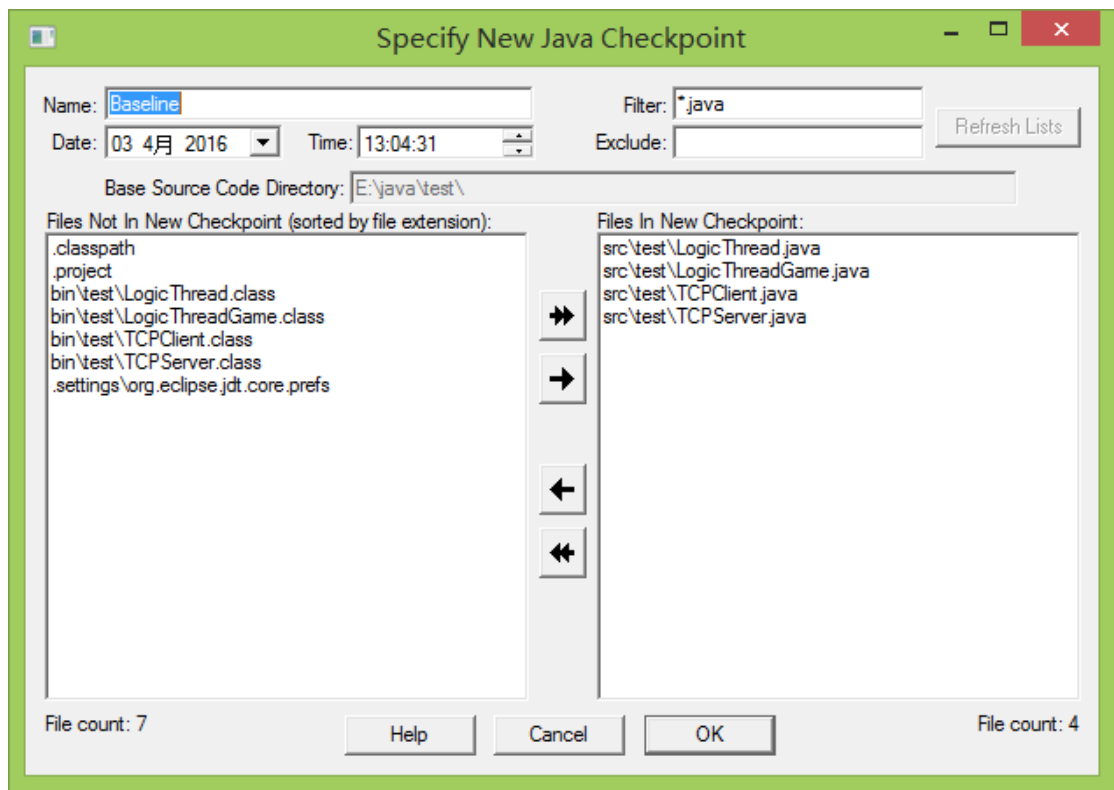
在 SourceMonitor 主界面选择 **New Project**。在弹出的新窗口中 **Project Source Code Language** 选择 **Java**。下一步后第一栏填入新工程的名字，第二栏填入新工程保存的位置。



再下一步，从文件管理器选择要分析的项目源代码位置：



继续下一步，使用默认配置，一直到配置新的 checkpoint 页，这里可以增加或删除要分析的源代码：




然后点击 **OK**，即可看到项目的分析结果：

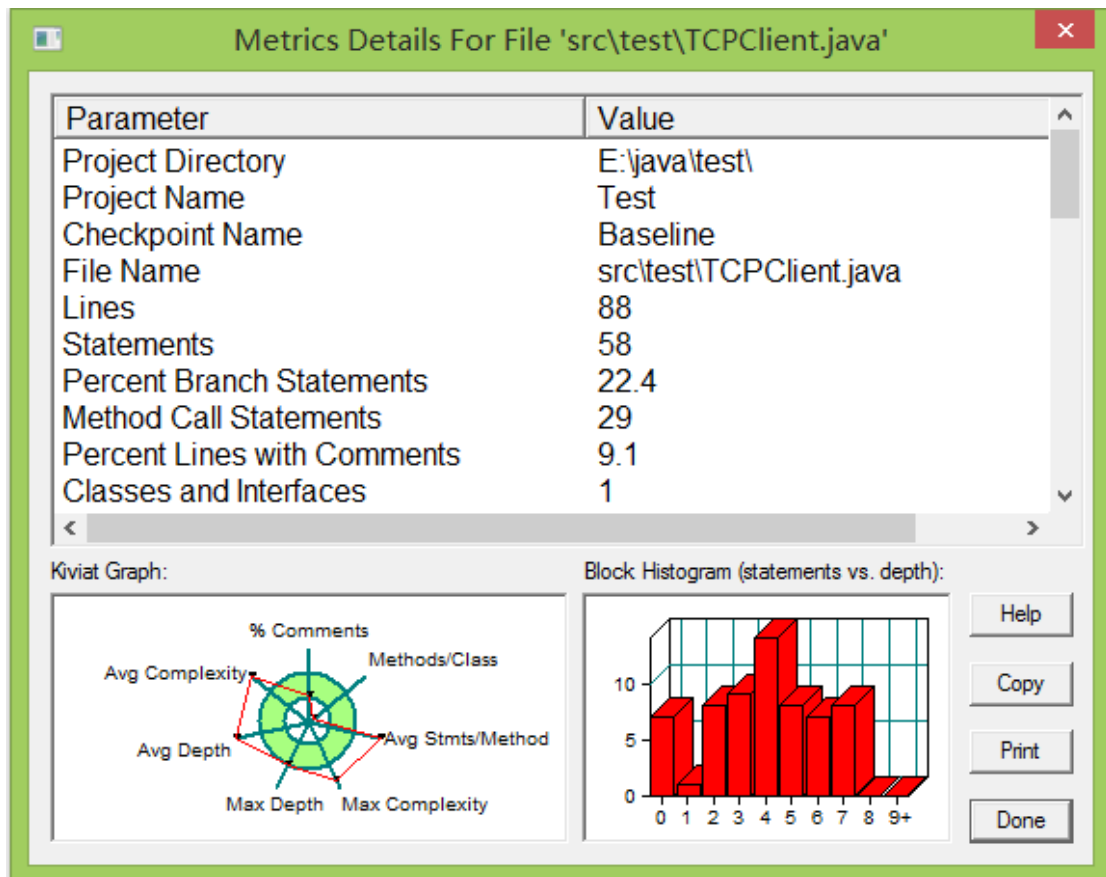
SourceMonitor - [Java Checkpoints In Project 'Test']

Checkpoint Name	Create ...	Files	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Strmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
Baseline	3 Apr 2016	4	262	154	13.0	69	16.0	4	2.00	14.88	12	7	3.05	4.00

双击这条记录查看详细信息：

SourceMonitor - [Files in Java Project 'Test', Checkpoint: 'Baseline' (Base Directory: E:\java\test\)]													
<div>File Edit View Window Help</div> <div></div>													
File Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmt/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity	
src\test\LogicThread.java	65	33	3.0	14	24.6	1	4.00	5.25	3	4	1.91	1.75	
src\test\LogicThreadGame.java	76	50	10.0	21	13.2	1	2.00	20.00	9	6	3.19	5.00	
src\test\TCPClient.java	88	58	22.4	29	9.1	1	1.00	50.00	12	7	3.84	12.00	
src\test\TCPServer.java	33	13	7.7	5	24.2	1	1.00	8.00	3	4	1.93	3.00	

我们可以看到第三条记录，即 **TCPClient.java** 的很多指标值很高，双击它查看详细信息：



根据结果可知这段代码的复杂度、深度等已经超出了正常范围。我们可以根据这一结果对代码进行重构，重构后再在此工程中新建 CheckPoint，进行新的分析。