软件测试的工具使用和实验指导

he Use of Tools and Experimental Instruction for Software Testing

主编　李枚毅

湘潭大学出版社

软件测试的工具使用和实验指导

he Use of Tools and Experimental Instruction for Software Testing

主　编　李枚毅

副主编　彭　成　潘丽丽

湘潭大学出版社

前　言

软件测试是软件工程的一个重要分支，它对测试人员的专业知识要求极全、专业技术要求极强、专业能力要求极高。目前，企业期望测试人员有较丰富的测试经验及较强的测试工具应用能力。随着国家、企业和个人对计算机系统软件质量的日益重视，计算机软件测试也走向了另一个新的阶段，为了紧跟计算机软件开发的迅速发展，培养计算机软件测试的能力，解决目前软件测试实验教学的紧迫需要，我们安排了本实验教材的编写。

本书使用目前最新版本的软件测试常用工具软件，并且提供了主要的实验指导内容。本教材突出基础性，并结合实际案例讲解软件测试常用工具软件的使用，深入浅出，详略得当，通用性好。针对软件测试的实验内容全面，实验方案完整，实践环境建设可行，实验步骤及过程讲解清晰，实验案例丰富实用，可作为高等院校软件工程及计算机相关专业的“软件测试实验课程”教材，也可以作为软件测试实战培训教材。同时，本书也是软件开发或管理人员、测试或质量保证人员非常好的自学参考书。

全书共有 15 章，包括 CheckStyle 的使用、PC－Lint 的使用、Junit 的使用、Google Test 的使用、UFT 的使用、Autorunner 的使用、Selenium 的使用、HP LoadRunner 的使用、JMeter 的使用、Appscan 的使用、Testcenter 的使用、单元测试 JUnit 实验、功能测试 UFT 实验、Web 功能测试实验、LoadRunner 性能测试实验等。

全书由李枚毅担任主编并统稿，彭成和潘丽丽担任副主编。第 1 章与第 15 章由王常飞和李枚毅编写，第 2 章和第 14 章由蔡伟彪和李枚毅编写，第 3 章与第 12 章由曹圣灵和李枚毅编写，第 4 章由曹圣灵和朱江编写，第 5 章由张晓编写，第 6 章由蔡伟彪和程莉莉编写，第 7 章由蔡伟彪和胡灿编写，第 8 章由王常飞和唐欢容编写，第 9 章由蔡伟彪和彭伟编写，第 10 章由王常飞和李枚毅编写，第 11 章由蔡伟彪和潘丽丽编写，第 13 章由张晓和李枚毅编写。

湘潭大学出版社为本书的出版提供了大力支持，出版社领导和相关编辑的出色工作给本书的质量和按时出版提供了保障。本书部分工作得到了湖南省教育厅教改项目“突出软件安全性能的《软件质量保证与测试》课程建设研究”和湘潭大学本科教材编写项目以及研究生院教材项目支持。有些章节多次重写和修改，作者们为此付出了大量辛苦劳动。主编们多次认真仔细地审订内容，保障了本书的质量。另外，本书编写过程中参考了一些同行的资料，在此一并对他们表示衷心的感谢。

本书配有一些实验中可能需要使用的素材，如果需要的话，请通过邮箱 cslmy888@sohu.com 与我们联系。

本书虽经多次讨论并反复修改，但限于编者水平和时间限制，不当之处仍在所难免，敬请广大读者批评指正。

编者

2016 年 4 月

内容简介

本书内容包括当前测试领域主要工具的使用方法和典型的实验项目。采用深入浅出、详略得当的方式结合案例讲解软件测试常用工具软件的使用，使用多年教学经验中总结出来的经典案例讲解实验项目，同时也是国内第一本正式出版的软件测试工具使用和实验指导书，能够有效地指导学生掌握软件测试的基本技能、提高软件质量和可靠性方面的能力，具有很强的指导和借鉴意义。

本书可作为高等院校相关专业不同学历教育（如本科生、研究生，甚至高职生或高专生等）软件测试类课程的实验教材或实验教学参考书，也可供从事计算机软件设计和开发的各类技术人员参考，或用作软件测评自学实验教材和培训资料。

目　录

第 1 章　CheckStyle 的使用

CheckStyle 是 SourceForge 下的一个项目，它根据设置的编码规则来检查代码，有效地帮助用户统一代码编写标准，特别适用于小组开发时彼此间的样式规范和统一。它默认提供 Javadoc 注释、命名约定、标题、Import 语句、体积大小、空白、修饰符、块、代码问题、类设计和混合检查等方面的检查规范。

CheckStyle 的配置性极强，用户可以根据自己的需求设定检查的规则和条数。另外，CheckStyle 几乎支持所有主流 IDE，包括 Eclipse、IntelliJ、NetBeans 和 Jbuilder 等。本章主要讲述 CheckStyle 在 Eclipse 环境下进行 Java 代码检查。

1.1　安装 CheckStyle 插件

使用 CheckStyle 之前检查是否安装了 CheckStyle 插件，方法是在 Eclipse 的“Window”菜单上单击“Preferences” 项，如图 1－1 所示。

图 1－1　在窗口菜单中选择首选项

在弹出的首选项窗口中，检查“Ant”名称下方是否有“Checkstyle”项，如果有如图 1－2 所示的“Checkstyle”项，则已经安装了 CheckStyle 插件；否则需要安装 CheckStyle 插件，方

法是打开“Help”主菜单，单击其子菜单“Install New Software”，在弹出的“Install”对话框中单击“add”按钮，输入一个名称（可以自己取，例如“checkstyle”）和网址：http://eclipse-cs.sourceforge.net/update，如图 1－3 所示。然后单击“OK”按钮，并按提示完成操作。

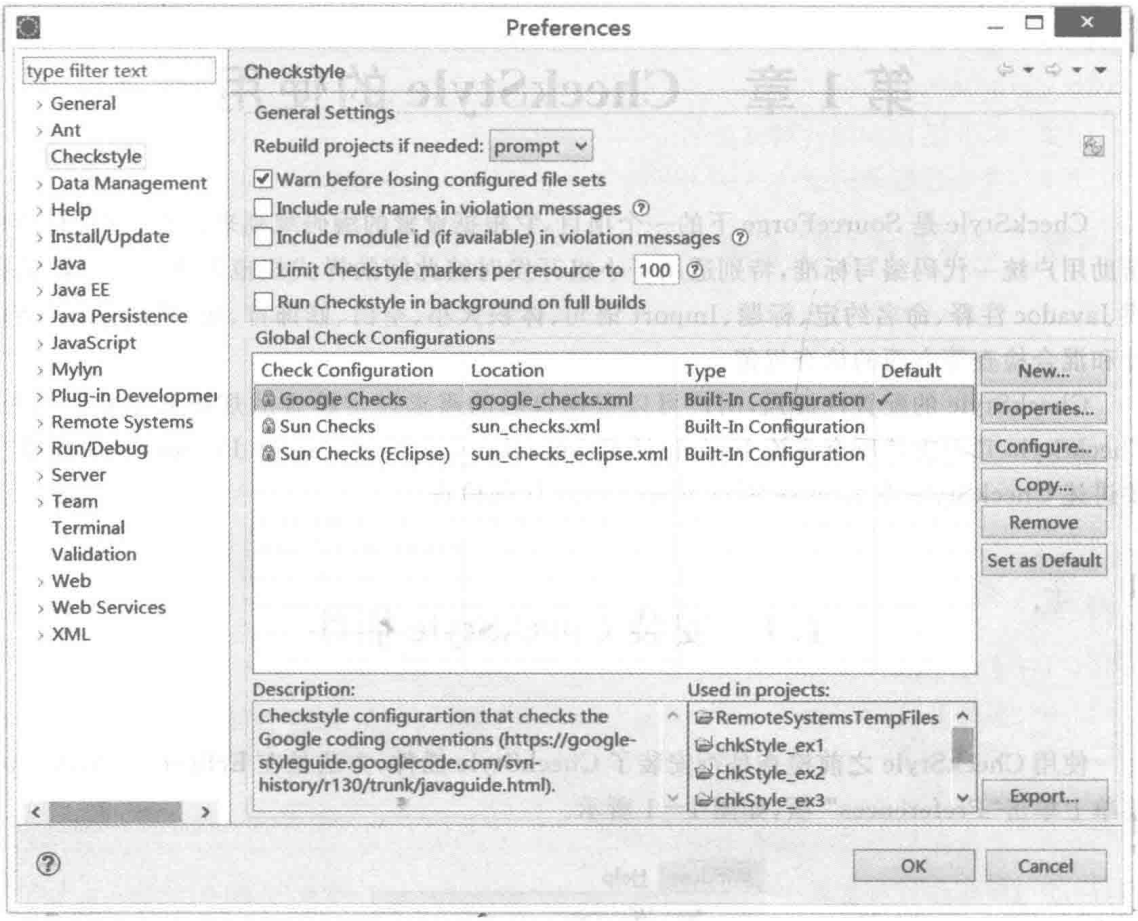

图 1－2　首选项窗口

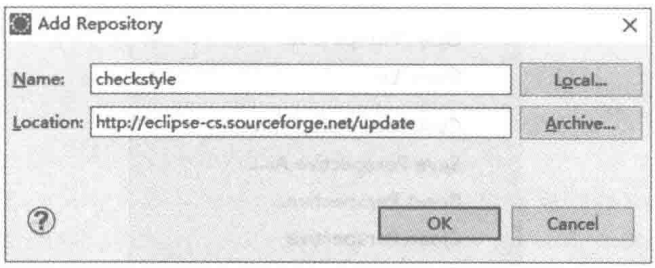

图 1－3　添加 CheckStyle 插件

在 CheckStyle 安装好以后，单击图 1－2 左侧窗格的 CheckStyle，在右侧窗格可以看到系统默认的 3 个检查配置：Google Checks、Sun Checks 和 Sun Checks(Eclipse)。而正在使用的检查配置为 Google Checks。

1.2　用 CheckStyle 检查 Java 代码

1.2.1　在 Eclipse 中生成一个类

首先创建一个 Eclipse 项目，方法是在“File”菜单中选择“New”子菜单下的“Java Project”，在弹出的“New Java Project”窗口中给项目起名为“checkproject”。

然后创建一个包，方法是右击“checkproject”，选择“New”子菜单下的“Package”，在“New Java Package”对话框中给包起名为“checkpackage”。

最后新建一个类，方法是右击“checkpackage”，选择“New”子菜单下的“Class”，在弹出的“New Java Class”窗口中给类起名为“CheckClass”并勾选“public static void main(String [] args)”选项，如图 1－4 所示。

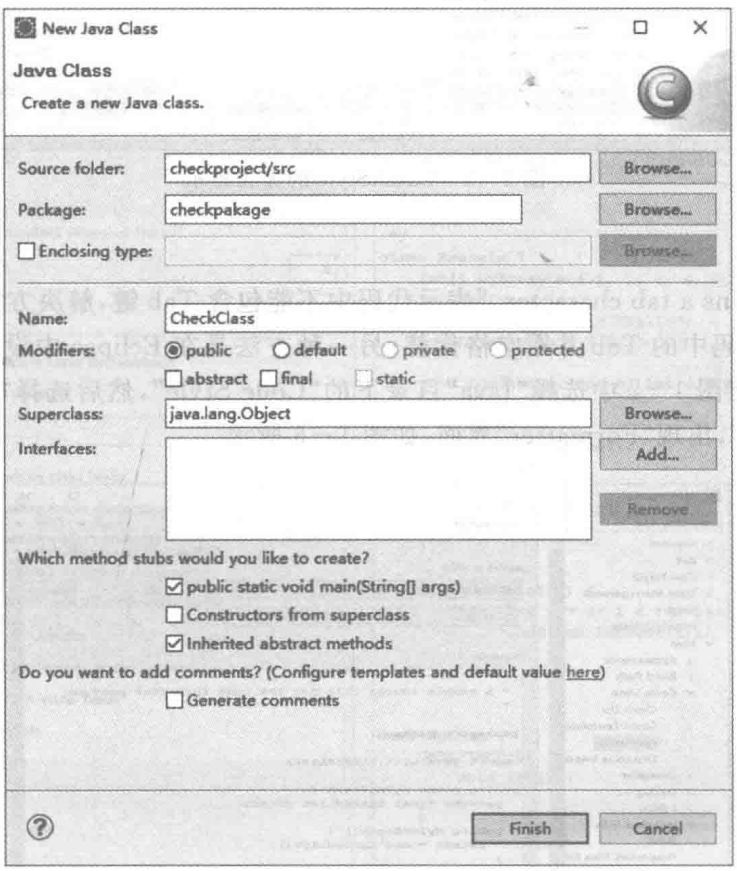

图 1－4　创建一个带主函数的类

1.2.2　用 Google Check 配置检查代码

右击“CheckClass”，在弹出的“右击菜单”中选择“Checkstyle”子菜单下的“Check Code with Checkstyle”，此时代码框中会有警告标志，同时在“Problems”选项卡中也会有警告提

示的说明，如图 1－5 所示。

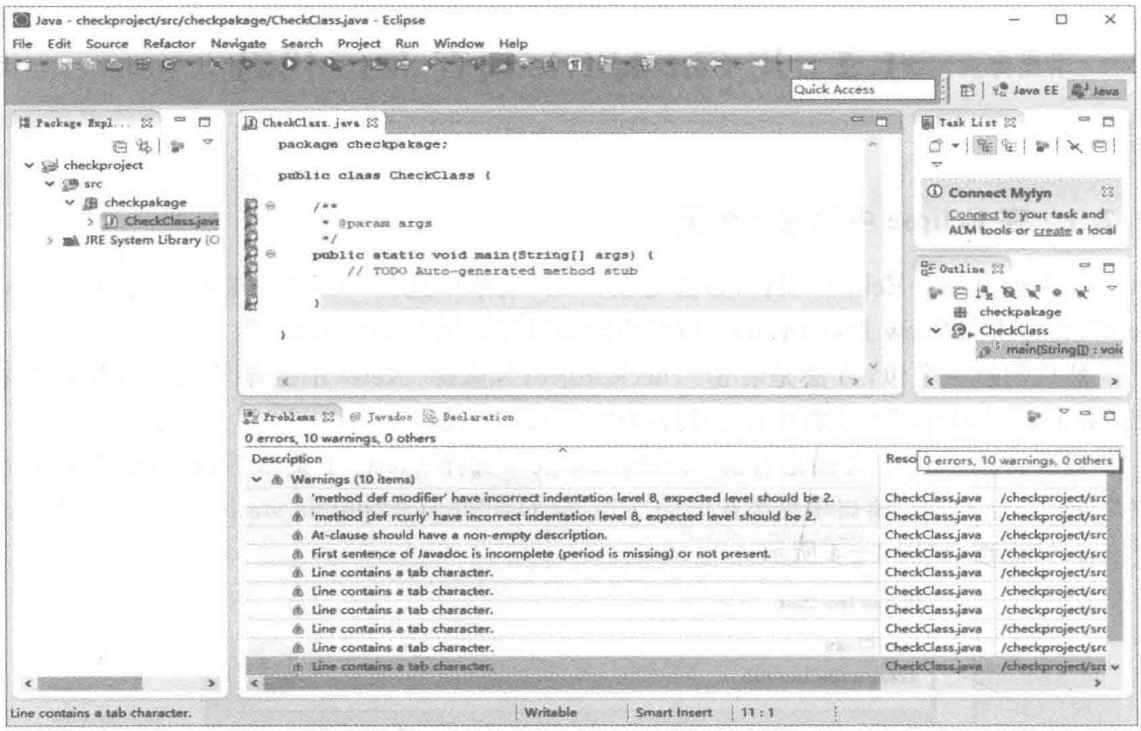

图 1－5　CheckStyle 的警告提示

1. Tab 问题

“Line contains a tab character.”表示代码中不能包含 Tab 键，解决方法有两种，一种方法是手动去掉代码中的 Tab 并用空格代替；另一种方法是在 Eclipse 中设置自动将 Tab 键用空格代替，即在图 1－2 中选择“Java”目录下的“Code Style”，然后选择“Code Style”目录下的“Formatter”，出现“Formatter”界面，如图 1－6 所示。

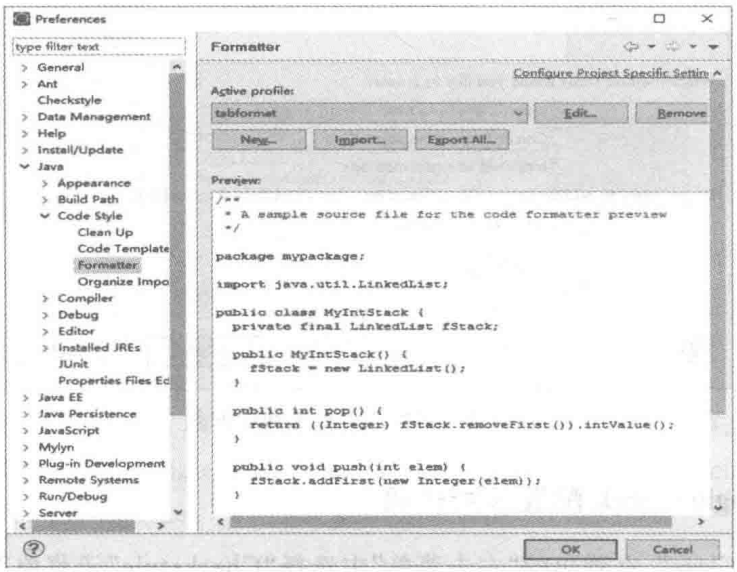

图 1－6　Formatter 界面

在“Formatter”界面中单击“New...”按钮来新建一个配置，弹出“New Profile”对话框，如图 1－7 所示。

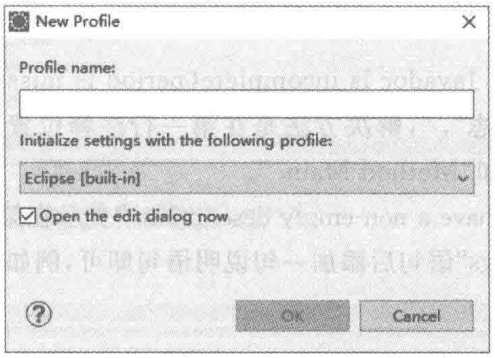

图 1－7　“New Profile”对话框

给新的配置起名为“tabformat”，单击“OK”按钮。然后在图 1－6 所示界面中单击“Edit”按钮，弹出“Profile 'tabformat'”窗口，如图 1－8 所示。

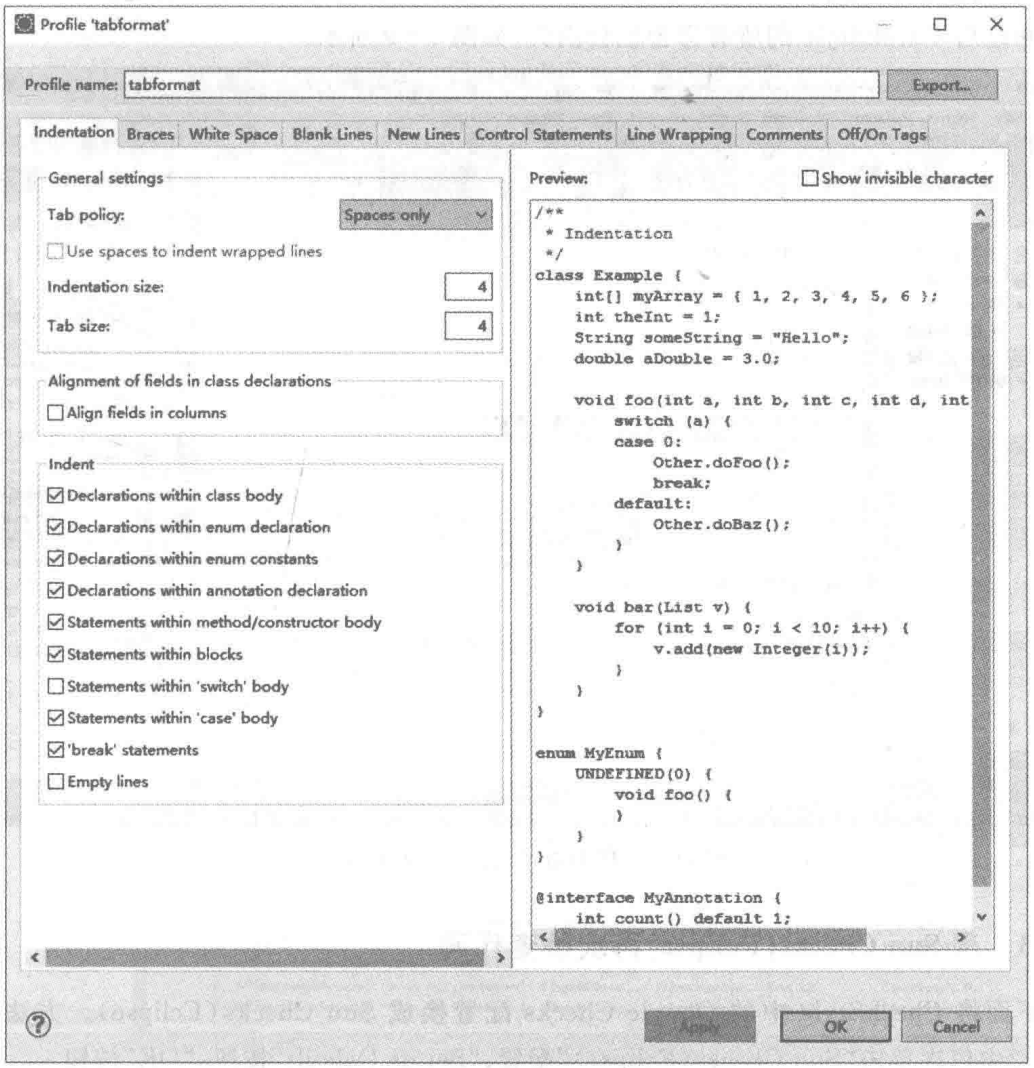

图 1－8　Profile 'tabformat'窗口

在“Tab policy”下拉列表中选择“Space only”，然后单击“OK”按钮。此时 Tab 键就被 4 个空格键代替了。

2. 注释问题

(1) “First sentence of Javadoc is incomplete(period is missing) or not present.”表示在 Java 注释中没有结束标志“.”，解决方法是在第一行注释位置添加任意语句后加上英文句号“.”作为结束标志，例如“Method Main.”。

(2) “At-clause should have a non-empty description.”表示在第六行的注释中应该有非空的描述，此时在“@param args”语句后添加一句说明语句即可，例如“which is...”。

3. 缩进问题

“'method def modifier' have incorrect indentation level 8, expected level should be 2.”表示缩进级别为 8 的缩进不正确，预期应该是 2。解决方法是在图 1－8 所示窗口中将“Indentation size”后文本框中的数字“4”改成“2”。

注：用空格键代替 Tab 键和缩进级别的设置在下一次创建类的时候生效。

修改后 CheckStyle 的所有警告就会消失，如图 1－9 所示。

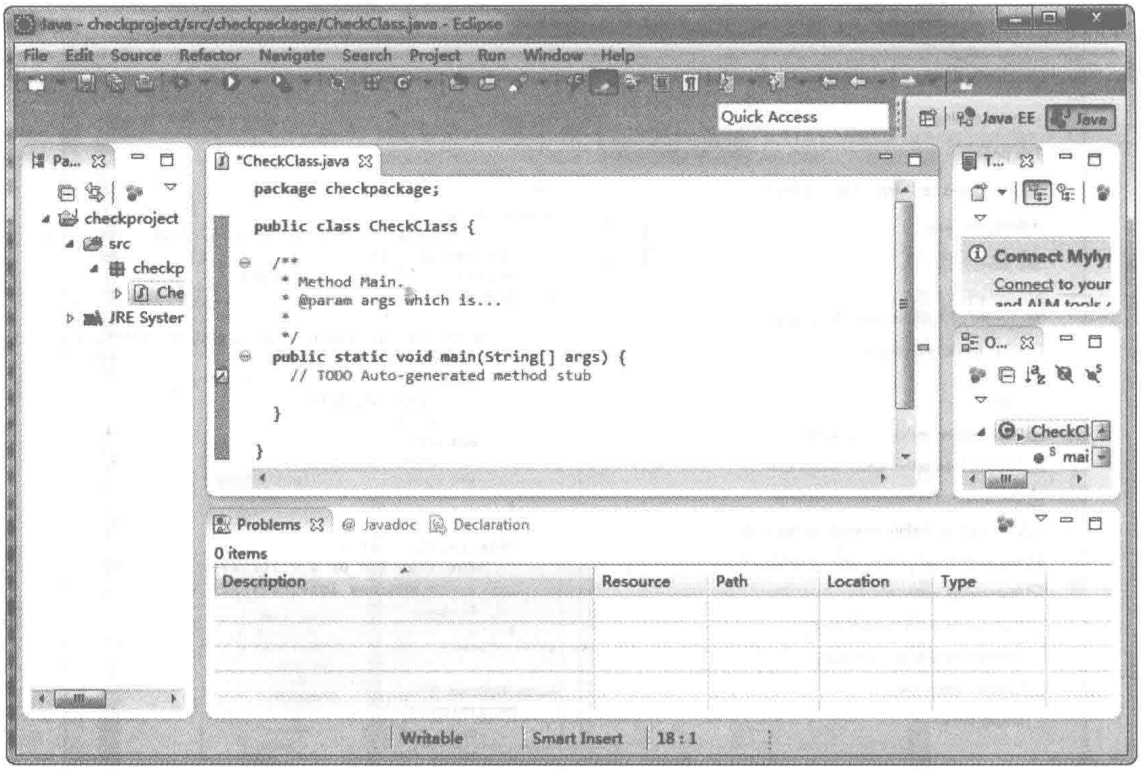

图 1－9 代码格式更正后的检查结果

1.2.3 用 Sun Checks(Eclipse)再次检查代码

下面将 CheckStyle 中的 Google Checks 配置换成 Sun Checks(Eclipse)。方法是在图 1－2中依次单击“Sun Checks(Eclipse)”配置、“Set as Default”按钮、“OK”按钮。

此时用“Sun Checks(Eclipse)”配置检查原来修改好的代码，结果如图 1－10 所示。

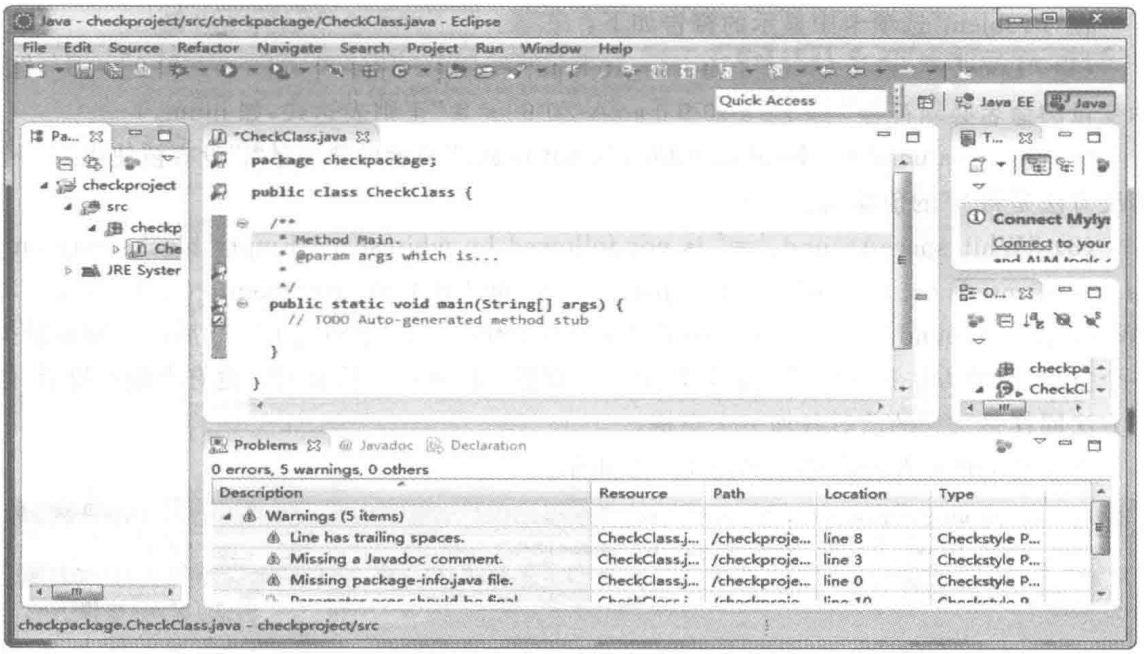

图 1－10　用 Sun Checks(Eclipse)配置文件检查

图中显示了 5 处警告，由此看出，不同的配置规定的代码风格也不一定相同。目前，“Sun Checks(Eclipse)”配置检查的严格程度明显高于“Google Checks”配置，甚至有些过头，因此本书使用“Google Checks”配置进行介绍。

1.2.4　运算符书写和变量命名规范

在主函数中加入一个变量，并给它赋值，代码为：“int i＝9％2;”，将配置重新换成 Google Checks，然后检查代码，结果如图 1－11 所示。

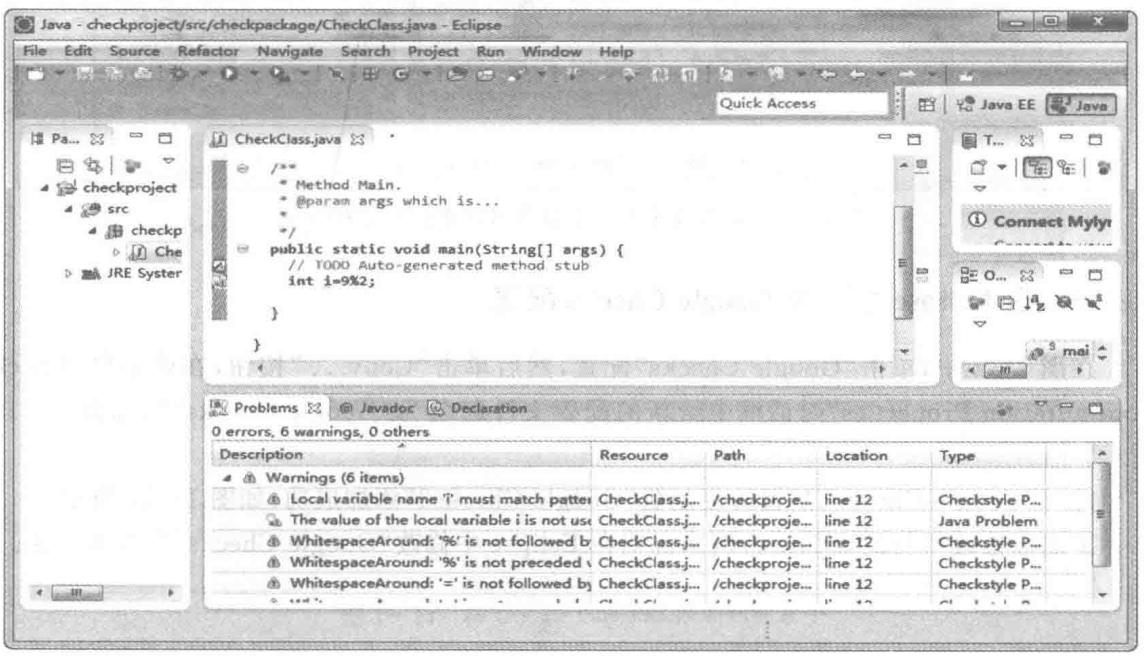

图 1－11　在代码中加入“int i=9％2;”后的警告提示

在“Problem”选项卡中显示的警告如下：

(1) “Local variable name 'i' must match pattern '^[a-z][a-z0-9][a-zA-Z0-9]* $'.”表示变量的命名必须符合“^[a-z][a-z0-9][a-zA-Z0-9]* $”正则表达式，如 inum。

(2) “The value of the local variable i is not used.”表示局部变量“i”没有被使用。一个解决方法是将“i”的值输出。

(3) “WhitespaceAround: '=' is not followed by whitespace. Empty blocks may only be represented as { } when not part of a multi-block statement (4.1.3).”和“WhitespaceAround: '=' is not preceded with whitespace.”分别表示“=”后面和前面缺少一个空格，解决方法是在“=”前后各加上一个空格。同样对运算符“%”也有类似的提示，修正方法同样是“%”的前后各加一个空格。

修改后的代码及检查结果如图 1－12 所示。

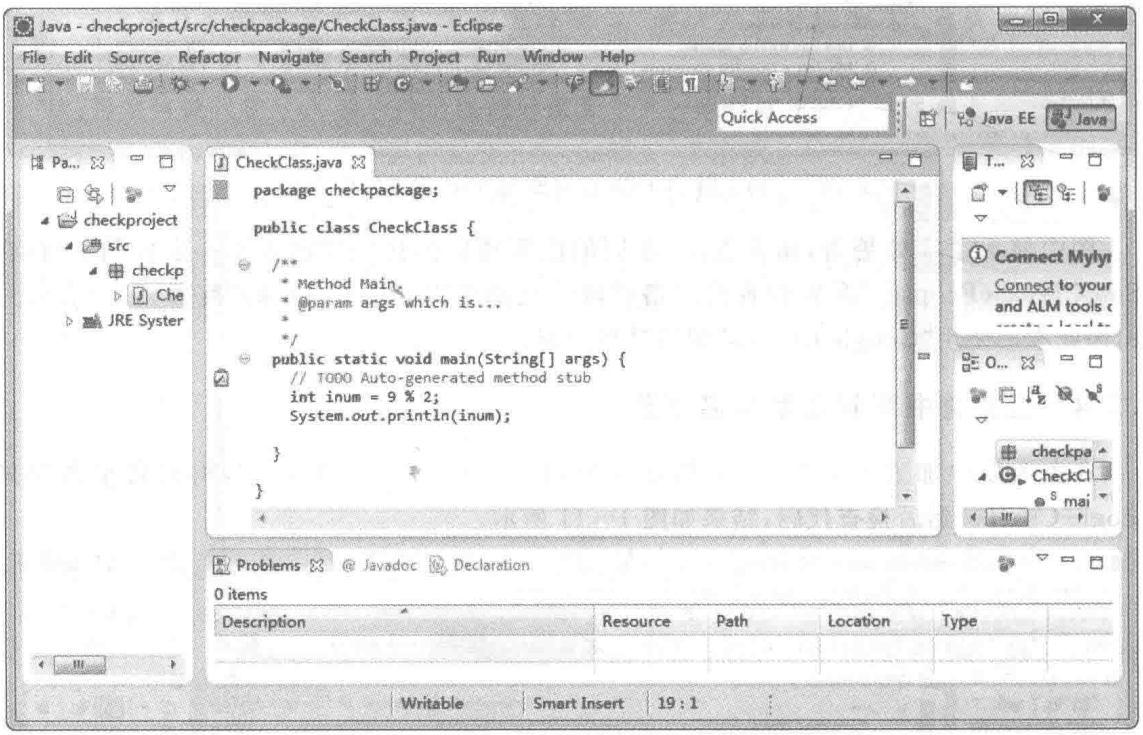

图 1－12 运算符书写和变量命名修正后的检查结果

1.2.5 在 Eclipse 中修改 Google Checks 配置

在图 1－2 中，单击“Google Checks”配置，然后单击“Copy...”按钮，在弹出的“Check Configuration Properties”对话框中给新的配置文件起名为“Google Checks2”，如图 1－13 所示。

此时“Google Checks”配置的复制版“Google Checks2”添加成功，如图 1－14 所示。

下面以修改每行最大长度为例来说明在 Eclipse 中修改“Google Checks”配置的方法。

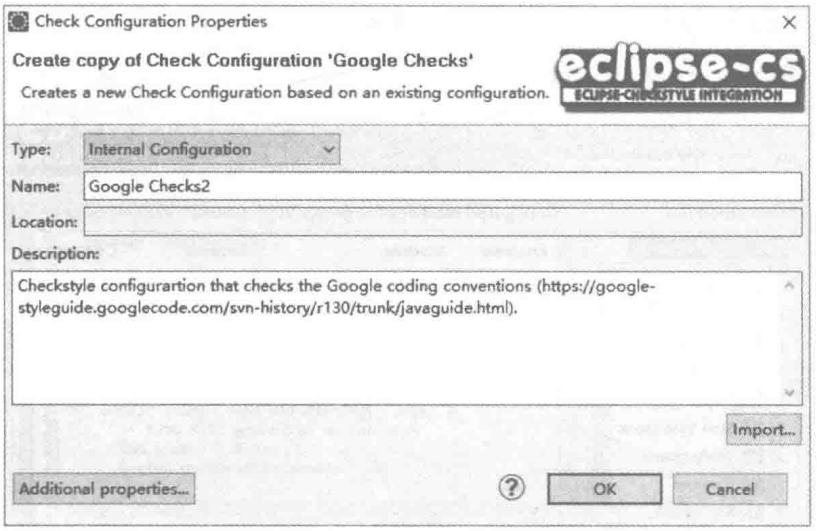

图 1－13　“Check Configuration Properties”对话框

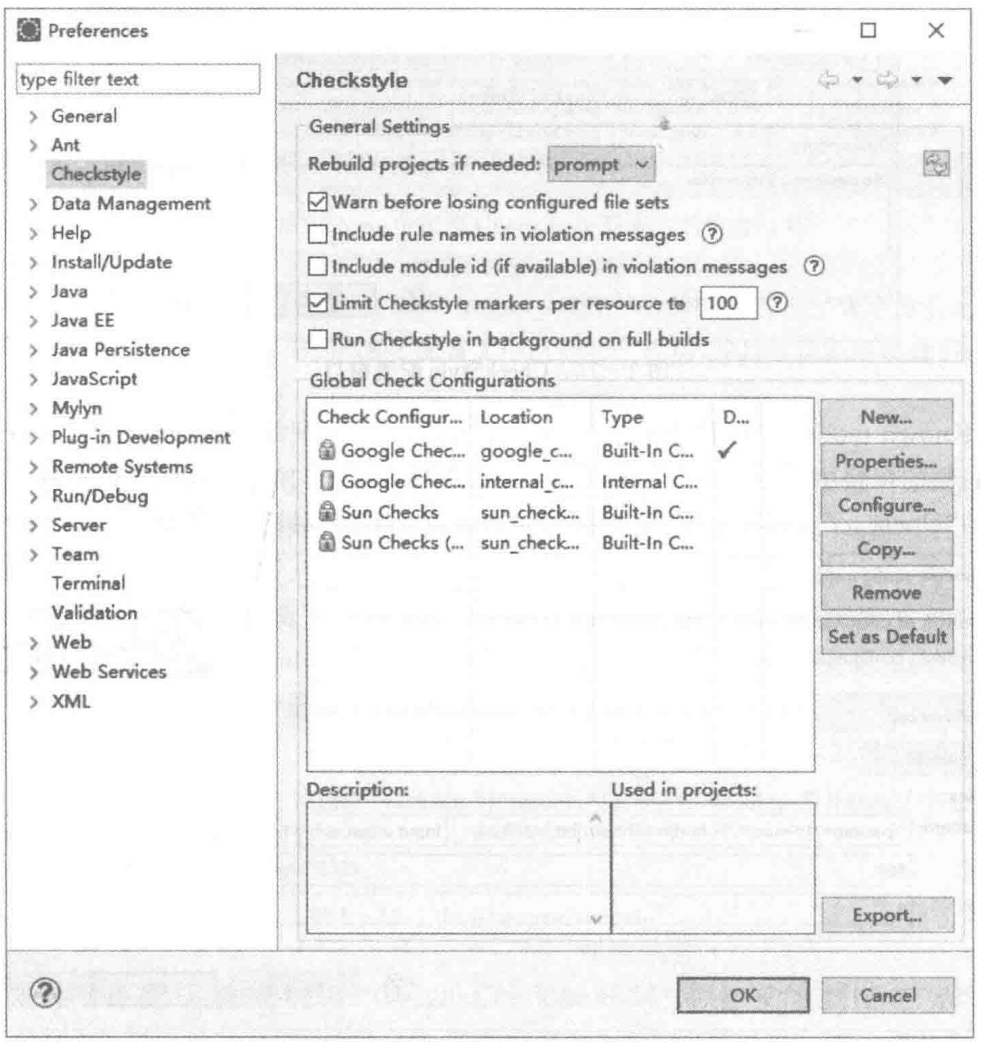

图 1－14　将 Google Checks2 添加到配置中

首先单击图 1－14 中的“Google Checks2”配置，然后单击“Configure...”按钮，弹出