CSC 216 Course Description

We recruited participants from CSC216, Programming Concepts in Java, a second semester programming course at NC State University. CSC216 covers topics on advanced object-oriented programming, design, testing, linear data structures (array-based lists and linked lists), recursion, and finite state machines. The third author taught two sections of CSC216 in Fall 2012.

The organization of this course gave us various opportunities to collect data for our study. The class is organized into two weekly lectures of 75 minutes. The instructor covers a topic and will periodically stop class to have students complete in-class exercises with their peers. We incorporated several in-class exercises for students to explore notifications from the TUI. Students completed three programming projects over the semester: two individual and one group project. The projects consisted of two parts: 1) a design rationale and black box test plan phase that considered a set of requirements and 2) the implementation, unit test, and black box test execution phase. The professor expected projects to meet the following standards: compile with no errors against Java 1.5; have 100\% method coverage when executing student written unit tests on non-GUI classes; and have no ``Scary'' or ``Scariest'' FindBugs notifications. Any notifications that go against this standard we will call ``critical'' notifications. The restriction of compilation against Java 1.5 is due to Web-CAT, an automated grading tool for the course. All students, whether participating in the study or not, had to follow the project guidelines and complete the in-class exercises.

As students worked on their projects, the professor expected them to ask questions directed at the teaching staff by emailing the mailing list for the course or by posting to the message board for the course. The instructor along with the coach responded to questions on the mailing list involving the TUI. The instructor also made sure to redirect any posts made on the course's message board regarding the TUI to the mailing list.

Course Materials

JUnit Tutorial
Code Coverage and Static Analysis Lecture Notes

CSC216 Eclipse Development Environment Setup

Target Audience: This tutorial is intended for those that are setting up their personal computers for Java application development in CSC216 at North Carolina State University.

Overview of Installation: This table includes the latest versions of the tools used in CSC216. Use these versions when creating a fresh install. Where possible, update to these versions if using an already existent install. The minimum version of Java required for CSC216 is 1.5.

Software	Version
Java Development Kit (JDK)	1.7 Update 10
Eclipse (32- or 64-bit as appropriate for your system)	Juno (4.2.x)
FindBugs	2.0.2
PMD	3.2.6
CheckStyle	5.6.0
EclEmma	2.2.0
Subversive (optional)	1.0.0

1.0 Overview of Development Environment

The Eclipse IDE provides a development environment for Java applications. The functionality of Eclipse is extended through the use of plug-ins. We'll use several plug-ins this semester:

- **FindBugs**: FindBugs is a static analysis plug-in that looks for potential problems with your code that compilation and some testing may not be able to find.
- **PMD**: PMD is another static analysis tool that looks for a different set of problems, some more related to style and programming language inconstancies.
- **CheckStyle**: CheckStyle is a third static analysis tool with a focus on style conventions of the Java programming language.
- **EclEmma**: EclEmma is a tool that measures the coverage (which methods and statements that are executed) when running automated unit tests and black box tests.
- **Subversive**: Subversive provides connectivity to a Subversion repository and allows teams to collaborate on a common code base and tracks changes to the code base over time.

2.0 Install Java Development Kit

Note for Mac Users: Java is pre-installed. You can skip this step.

Java libraries are required to run the Eclipse platform. The Java Development Kit (JDK) provides the Java libraries in addition to the `javac` and `java` commands for compiling and executing Java programs. The Java Runtime Environment (JRE) provides the Java libraries and the `java` command for running Java programs. While Eclipse only requires the JRE to run; most Java developers prefer to install the JDK so that they have access to the compilation functionality from the command line or when using other IDEs.

Java also comes in several editions: standard edition (SE), enterprise edition (EE), and micro edition (ME). EE is used for enterprise level development, typically at large corporations, while ME is used for mobile and embedded development. SE provides enough functionality for most medium to large development activities. You want to download a **Java SE**.

The JDK can be downloaded from Oracle. Download the executable (SE either JDK or JRE) for your operating system and follow the installation instructions. Remember which directory you installed the JDK in. If you don't like the provided directory, you can select another directory.

After installing, we want to add the `javac` and `java` command to our path so we can run them from the command line.

***Windows**:*
To do this, open your **Control Panel**, and select **System**. On Windows XP, select the **Advanced** tab. On Windows Vista, select the **Advanced System Settings** task. In the resulting window, select the Environment Variables button. On Windows 7, search for "System" and select the link to **Edit the system environment variables**.

Select the **Path** variable and select **Edit**. Move your cursor to the end of the text field for entering the value. Add a semi-colon (;) and add the full path to the `bin` directory of your Java installation. Select **OK**.

Select the **CLASSPATH** variable and select **Edit**. Move your cursor to the end of the text field for entering the value. Add a semi-colon (;) and put a period (.). Select **OK**. The period represents your current working directory, and means that all files in your current working directory will be on the classpath when compiling and running Java applications. If you do not have a CLASSPATH variable, select the **New** button, and create one. The name of the variable should be CLASSPATH, in all capital letters.

See http://courses.ncsu.edu/csc116/common/java_install/ for a video on how to install the Java SE Development Kit.

3.0 Download Eclipse

Eclipse is available in several distributions with plug-in configurations geared towards specific development activities. Download the Eclipse IDE for Java Developers. If you do not see an option for Eclipse IDE for

Java developers, you can try a more direct link or download Eclipse Classic. **Figure 1: Eclipse download selection**

Eclipse does not require installation. Instead you will extract the zipped files to the directory of your choosing (we recommend that **Windows** users install under your top drive, like **C:** and **Mac** users install in **Applications**). You can then start Eclipse by entering the **<eclipse-home-dir>/eclipse** and double clicking the purple Eclipse icon.

Start **Eclipse**.

4.0 Install FindBugs and EclEmma through Eclipse Marketplace

Eclipse Marketplace is a standard location where plug-in authors can provide their plug-ins for Eclipse users. Access the Marketplace through **Help > Eclipse Marketplace...**. For each plug-in (FindBugs and EclEmma), complete the following steps:

1. Enter the plug-in name in the search bar and search for the plug-in
2. After you find the plug-in, click the **Install** button.
3. The menu will list the plug-in features that will be installed. You do not have to install optional

   features. Select the features you want to install and click **Next>**.

4. Accept the license agreement and click **Finish**.

The plug-in will install, and you will be prompted to restart Eclipse. You may restart Eclipse after each installation, or you can wait until you have installed all of the plug-ins and then restart Eclipse.

5.0 Install Subversive through Eclipse Marketplace (includes an additional extra step)

To install Subversive, complete the following steps:

1. Access the Marketplace through **Help > Eclipse Marketplace...**
2. Enter Subversive in the search bar and search for the plug-in
3. After you find the plug-in, click the **Install** button.
4. The menu will list the plug-in features that will be installed. You do not have to install optional

   features. Select the features you want to install and click **Next>**.

5. Accept the license agreement and click **Finish**.
6. Restart Eclipse.
7. You will be prompted to install an SVN connector. Select the **1.7.X SVN Kit Connector (or the**

   latest version of the SVN Kit Connector). If you are not automatically prompted to install the SVN connector, do the following:

   1. Click

   - **Windows users: Window > Open Perspective > Other > SVN Repository Exploring**
   - **Mac users: Eclipse > Open Perspective > Other > SVN Repository Exploring**

2. You will be prompted to install an SVN connector. Select the **1.7.X Kit Connector (or the**

latest version of the SVN Kit Connector). 8. Restart Eclipse.

6.0 Installing PMD and CheckStyle through Update Sites

Not all plug-ins are available through Eclipse Marketplace. Instead, some plug-in authors host their own plug-in update sites that users can connect to when installing their plug-ins. The URLs to the appropriate update sites are listed in Step 3, below.

Follow these steps for installing a plug-in from an update site:

1. Select **Help > Install New Software...**.
2. Click the **Add** button to add a new update site.
3. Enter the plug-in name and the plug-in update site URL. Click **OK**.

- PMD Update Site: http://pmd.sf.net/eclipse
- CheckStyle Update Site: http://eclipse-cs.sf.net/update/

4. Select the checkbox next to the plug-in name and click **Next >**.
5. Verify your install details and click **Next >**.
6. Accept the license agreement and click **Finish.**

You may be prompted with a warning about installing software that contains unsigned content when installing from non-Eclipse Marketplace sources. Please select **OK** to continue with the install. The plug-in will install, and you will be prompted to restart Eclipse. You may restart Eclipse after each installation, or you can wait until you have installed all of the plug-ins and then restart Eclipse.

7.0 PMD and CheckStyle Settings

PMD and CheckStyle can notify you about many potential anomalies in your code. With all notifications turned on, you could receive hundreds of reports and some of those reports would be duplicate between PMD and CheckStyle. Instead, we will use custom configurations for PMD and CheckStyle.

Download the PMD and CheckStyle configuration files for CSC216. These are the configuration files that we use for automated grading. By having a copy of these configuration files locally in Eclipse and running PMD and CheckStyle on your code, there should be no surprises during grading.

PMD Configuration File

Follow these steps to install the PMD configuration file:

1. Select

- **Windows users: Window > Preferences > PMD > Rules Configuration.**

- **Mac users: Eclipse > Preferences > PMD > Rules Configuration.**

2. Click the **Clear all** button. This will remove all of the current rules.
3. Click the **Import rule set...** button and **Browse** for the csc216_pmd.xml file. You can leave the **Import by Reference** check box selected. Click **OK**.
4. The rules will be listed. Click **OK** to apply the rules. If prompted to complete a full build of your workspace, click **OK**.

CheckStyle Configuration File

Follow these steps to install the CheckStyle configuration file:

1. Select

- **Windows users: Window > Preferences > CheckStyle.**

- **Mac users: Eclipse > Preferences > CheckStyle.**

2. Click the **New...** button.
3. Select the **Type** to be **External Configuration File**. Give the configuration a name, like **CSC216**. **Browse** to the location ofcsc216_checkstyle.xml. Click the **OK** button.
4. Select the configuration you just created in the **Global Check Configuration** table. Click the **Set as Default** button.
5. Click the **OK** button. If prompted to complete a full build of your workspace, click **OK**.

NOTE: There is a known inconsistency between the CheckStyle alerts in Eclipse and Web-CAT. When using CheckStyle in Eclipse, you will NOT receive an alert on a method lacking Javadoc if the method has an @override tag. When using CheckStyle in Web-CAT, you WILL receive an alert on a method lacking Javadoc even if there is an @override tag. We expect that you will comment EVERY method even if there is an @override tag.

8.0 Java 1.5 Source Compatibility

The tool we're using for automated grading, Web-CAT, expects that submitted code will conform to the Java 1.5 API. That means you should not use classes or methods that were introduced to Java in versions 1.6 or later. We can enforce this requirement locally by specifying the source compatibility of our Eclipse workspace. Follow these steps to set up the 1.5 source compatibility for your Eclipse workspace:

1. Select

- **Windows users: Window > Preferences > Java > Compiler.**

- **Mac users: Eclipse > Preferences > Java > Compiler.**

2. In the **Compiler compliance level:** drop down, select 1.5.
3. Click **OK.** If prompted to complete a full build of your workspace, click **OK.**

The source compatibility can also be set at the project level, rather than for all projects in the workspace. Follow these steps to set the source compatibility for a single project:

1. Right click on the project name and select **Properties**.
2. Select **Java Compiler**.
3. Check the box labeled **Enable project specific settings.** 4. In the **Compiler compliance level:** drop down, select 1.5.