**PMD For Eclipse**

**Introduction**

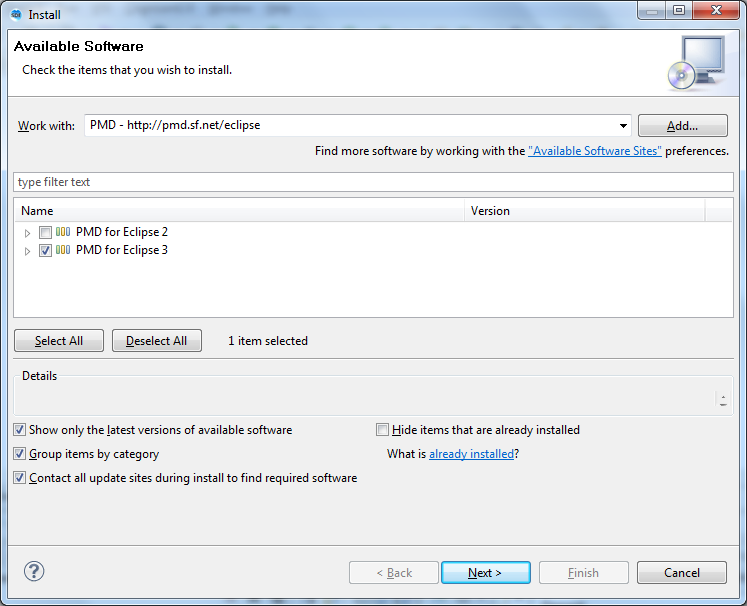
PMD scans Java source code and looks for potential problems like:

* Possible bugs - empty try/catch/finally/switch statements
* Dead code - unused local variables, parameters and private methods
* Suboptimal code - wasteful String/StringBuffer usage
* Overcomplicated expressions - unnecessary if statements, for loops that could be while loops
* Duplicate code - copied/pasted code means copied/pasted bugs

PMD is integrated with JDeveloper, Eclipse, JEdit, JBuilder, BlueJ, CodeGuide, NetBeans/Sun Java Studio Enterprise/Creator, IntelliJ IDEA, TextPad, Maven, Ant, Gel, JCreator, and Emacs.

**Installing the PMD plugin for Eclipse from update site**

1. Start Eclipse and open a project
2. Select "Help"->"Software Updates"->"Find and Install"
3. Click "Next", then click "New remote site"
4. Enter "PMD" into the Name field and "http://pmd.sf.net/eclipse" into the URL field



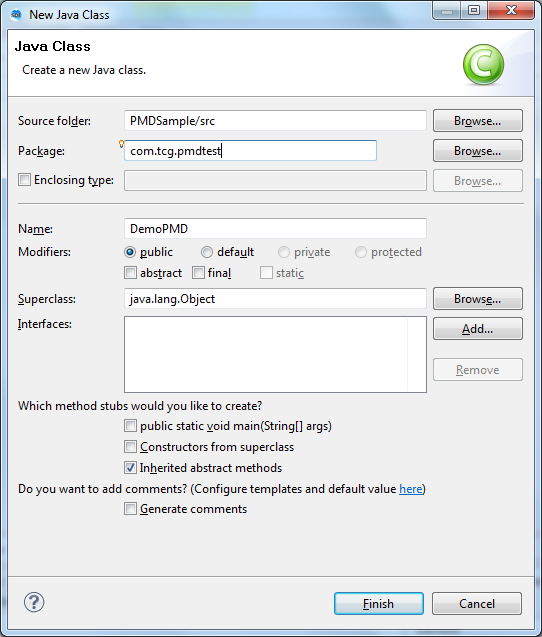
1. Click through the rest of the dialog boxes to install the plugin

**Creating the Sample Project to Analyze**

1. Launch Eclipse
2. Create a new project called “PMDSample”

File 🡪 New 🡪 Java Project

1. In the Package Explorer right-click on PMDSample project and select New | Class
2. In the dialog enter the class name as DemoPMD, package structure and click Finish



1. A new class DemoPMD is created in project’s default package. Paste the following code into the new class.

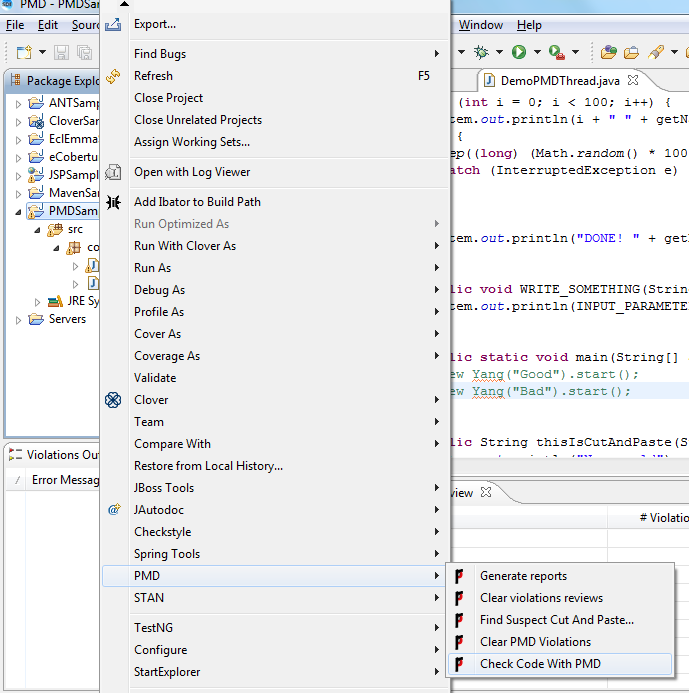
|  |
| --- |
| package com.tcg.pmdtest;  public class DemoPMD  {  private static final String bAD = "BAD";  public void BadMethodName() {  System.out.println("Hello PMD World!");  }  public String thisIsCutAndPaste(String first, int second) {  System.out.println("New world");  return "New world";  }  } |

1. Similarly, create a second class **DemoPMDThread** and paste the following code

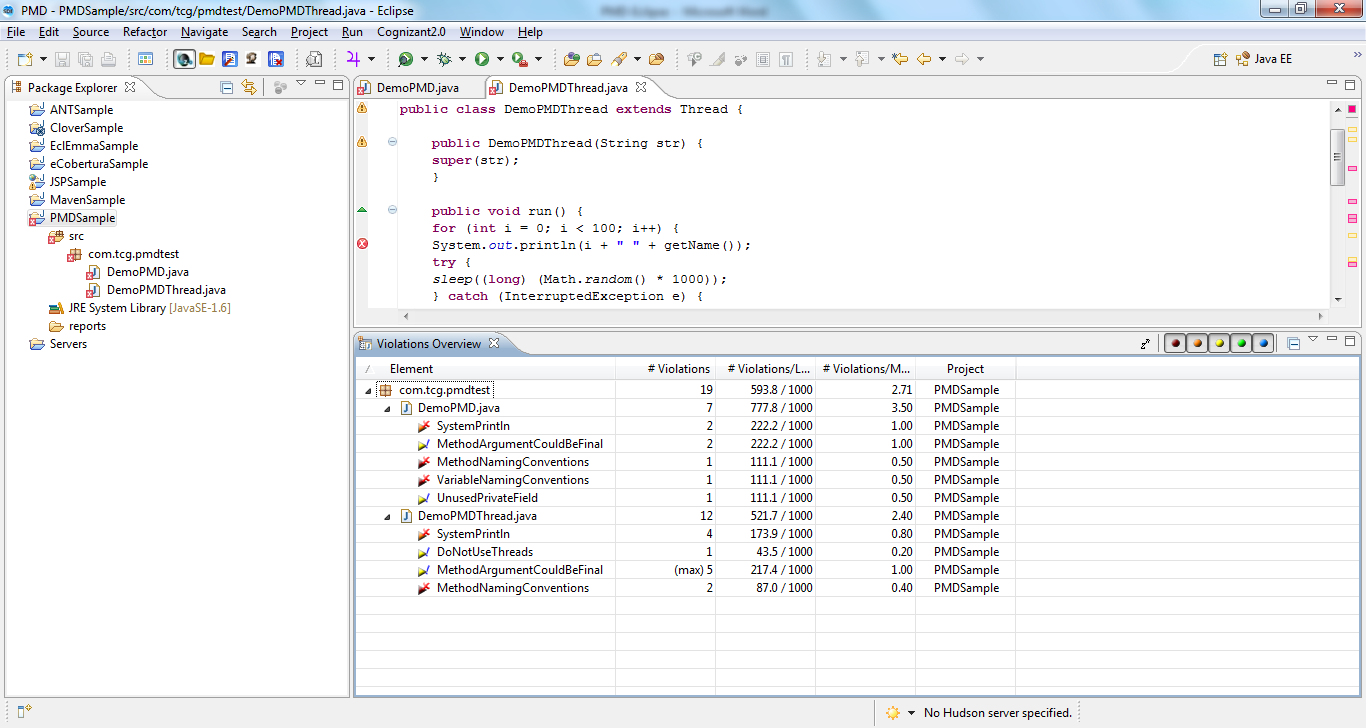
|  |
| --- |
| package com.tcg.pmdtest;  public class DemoPMDThread extends Thread {  public DemoPMDThread(String str) {  super(str);  }  public void run() {  for (int i = 0; i < 100; i++) {  System.out.println(i + " " + getName());  try {  sleep((long) (Math.random() \* 1000));  } catch (InterruptedException e) {  }  }  System.out.println("DONE! " + getName());  }  public void WRITE\_SOMETHING(String INPUT\_PARAMETER) {  System.out.println(INPUT\_PARAMETER);  }  public static void main(String[] args) {  new DemoPMDThread("Good").start();  new DemoPMDThread("Bad").start();  }  public String thisIsCutAndPaste(String pFirst, int pSecond) {  System.out.println("New world");  return "New world";  }  } |

**Running the PMD project**

1. In the Package Explorer right-click on QA Project and select PMD | Check Code With PMD

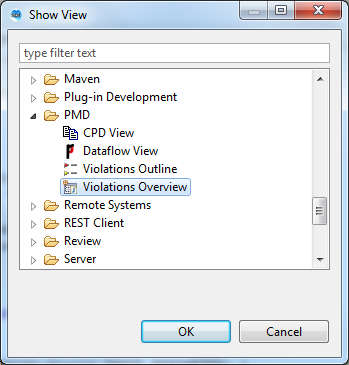


1. Wait for PMD to scan the project.
2. You can see the violation details in “Violation Overview”

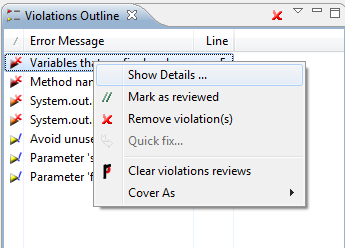


1. If you are unable to locate PMD " Violations overview" then

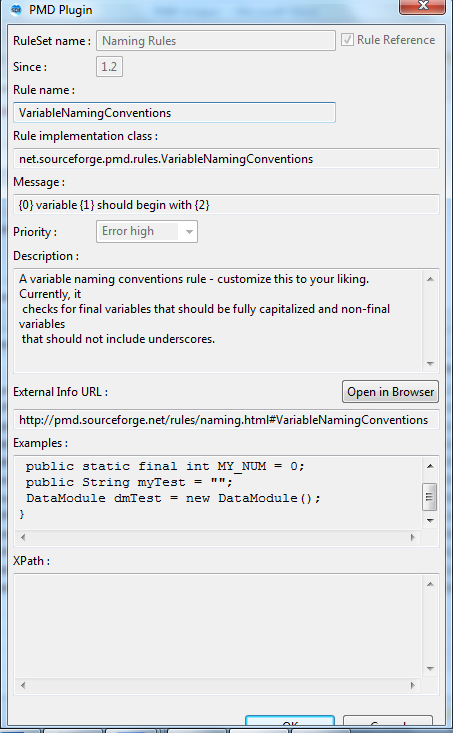
Navigate to Window 🡪 Show View 🡪Other 🡪select PMD 🡪Violations overview



1. It is also possible to get more detail on a violation by simply right-clicking it and selecting Show Details. PMD pops-up a dialog with information such as rule name, implementation class, message, description and an example. This feature can be helpful when trying to makes sense of a new rule or letting inhouse developers know about a particular company rule or coding convention.



1. Clicking on show details will get the below pop-up

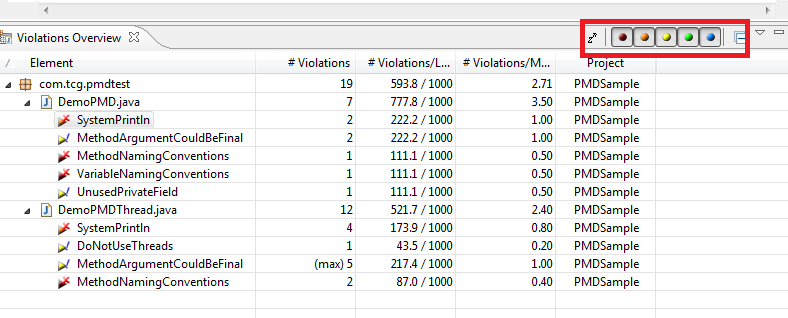


**PMD Features**

1. In the PMD Violations table it is possible to add review annotations to the source where the violation occurred. This can be an effective way to mark target files for further review. Just right-click on any violation in the list and select Mark review. PMD will insert a review annotation to the Java source right above the violation line itself. The review annotation should look like this:

// @PMD:REVIEWED:MethodNamingConventions: by Sanjay Singh on 5/08/10 5:04 PM

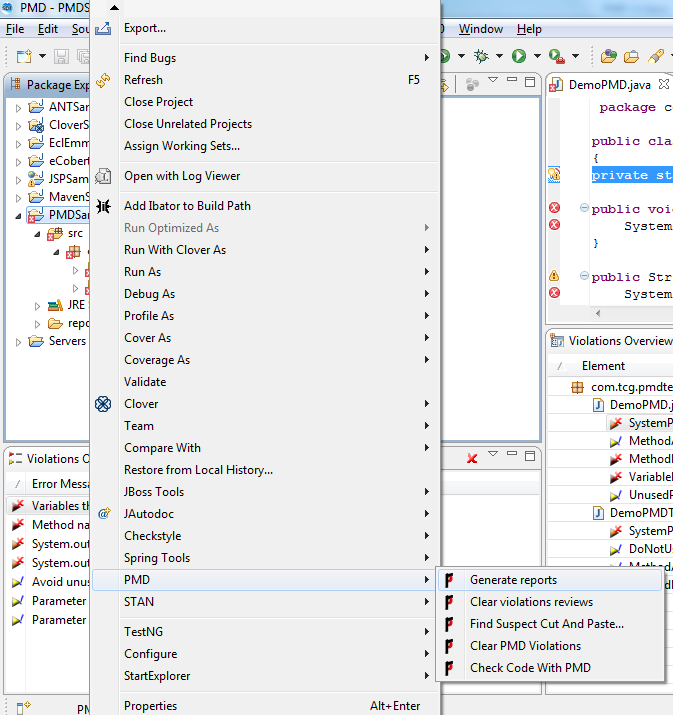
1. Review annotations can be removed anytime by right-clicking on the Project and selecting PMD 🡪 Clear violations reviews. Similarly, PMD Violations can be cleaned-up by right-clicking on the Project and selecting PMD 🡪 Clear PMD Violations.
2. In large projects this list could easily grow up to several thousand. This is one of the reasons PMD allows violations be filtered by priority. Priority is a configurable attribute of a PMD rule. PMD assigns priorities from 1 to 5 and each priority is represented by a colored square at the top-right corner of the view. These little squares are actually clickable on-off switches used to control the visibility of the violations they represent.



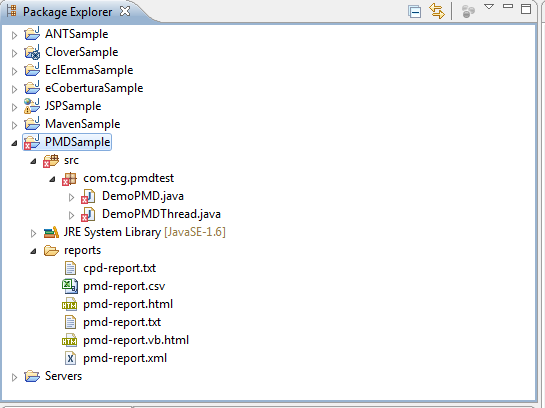
**Generating PMD Reports from Eclipse**

PMD provides a handy reporting tool it is able to generate reports in multiple formats. Currently PMD can generate reports in HTML, XML, plain text as well as Comma Separated Value (CSV) formats. You can do this without using any other tool.

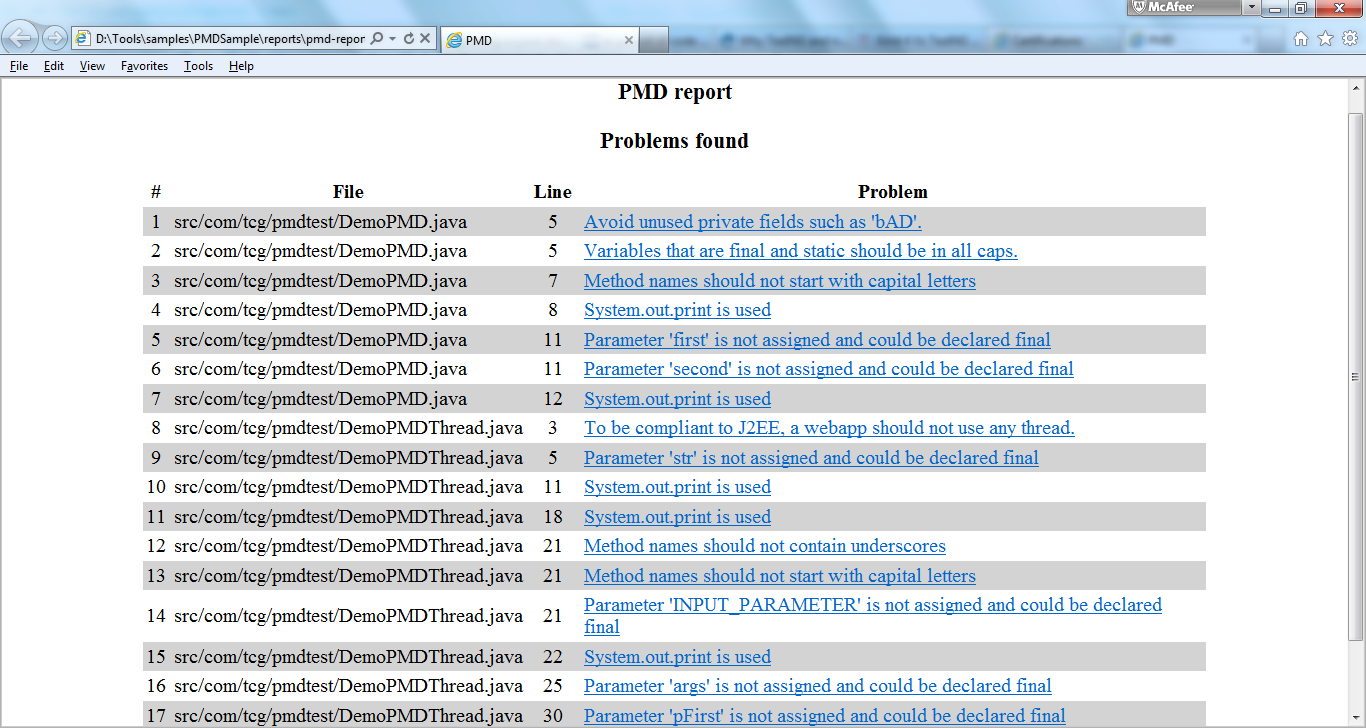
1. Right Click on the java project in the project explorer and select PMD 🡪 Generate reports



1. It creates a report folder in your application and creates violation report.
2. In Navigator view or package explorer view click on Project and then on the reports folder
3. That folder contains same report in different formats ( text, csv, html, xml).



1. Html report will looks like as shown below.



**Customizing PMD**

The easiest way to begin customizing PMD is by playing with existing rules. Adding new rules is also possible as well as removing unnecessary ones, however these require more knowledge. Since experimenting with existing rules is the easiest, it makes sense to start with them.

Each PMD rule has six attributes:

1. Rule name - immutable
2. Rule implementation class - immutable
3. Priority
4. Message
5. Description
6. Example

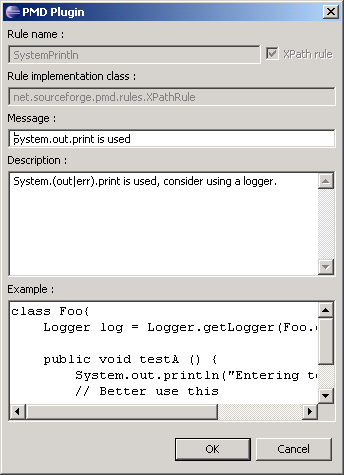
Of these six attributes the first two are immutable - they cannot be customized by users. While Message, Description and Example are text-based properties and can accept any String data, Priority is an integer field ranging from 1 to 5.

PMD stores rule configuration in a special repository referred to as the Ruleset XML file. This configuration file carries information about currently installed rules and their attributes. Changes made through the Eclipse Preferences page are also stored in this file. In addition, the PMD Preferences page allows exporting and importing Rulesets which makes them a convenient vehicle for sharing rules and coding conventions across the enterprise.

Before starting the customization it may be a good idea to back-up existing configuration.

**Modifying the Rule set**

1. Navigate to Window 🡪 Preferences 🡪 PMD 🡪 Rules configuration
2. Click Export rule set...
3. Enter a file name e.g. pmd-rules.xml and click Save
4. Back in the Rules configuration page scroll-down and click on SystemPrintln rule
5. With the rule selected, click the Priority column and change the priority from Error (2) to Information (5)
6. With the rule still selected, click the Edit rule... button on the right



1. Change the Message, Description and Example. Click OK when done
2. Click Export rule set...
3. Enter a new file name e.g. pmd-rules-customized.xml and click Save
4. Compare the two XML files - they are different
5. In Package Explorer right-click on QA Project and select PMD 🡪 Clear PMD Violations
6. In Package Explorer right-click on QA Project and select PMD 🡪 Check Code With PMD
7. Wait for PMD to finish the scan
8. If PMD Violations view is not open navigate to Window 🡪 Show View 🡪 Other... and select PMD 🡪 PMD Violations
9. Now the system.out.println will be priority 5