## Massey University

## 159.251 Software Design and Construction

## Tutorial 3 - Software metrics and code analysis tools

### Prerequisites (what you are expected to do before you start the tutorial)

1. Study the lecture material on software metrics
2. Get the projects required for this tutorial from Git
3. Before you start a task, install the tools required to complete the tasks (see the instructions below each task about the tool)

**Tools Required**

* + Metric Plugin (Task 1)
  + PMD (Task 2)
  + Spotbugs (Task 3)

### Objectives

1. Get familiar with using software metrics tools to measure several aspects of a program.
2. Learn about other aspects of static analysis tools such finding potential source code bugs in a program.

**Projects Required to complete the tasks**

Clone the following working projects:

**1- Apache commons lang:**

<https://github.com/apache/commons-lang.git>

**UPDATE: if you have problems compiling common-lang, you can download all dependencies from** <https://aatahir@bitbucket.org/aatahir/common-lang-updated.git> as a zip file. Unzip this file and add the new folder under common-lang/lib

**2- JDepend**

<https://github.com/clarkware/jdepend.git>

then import those projects into Eclipse.

**Note**: make sure that you correctly build those projects (e.g., fix paths and dependencies) by running the Maven and Ant files, as you have learned in the previous tutorial. For JDepend, you may also need to add Junit to the Build Path! Have a look at the ***readme***  files of each project for more details.

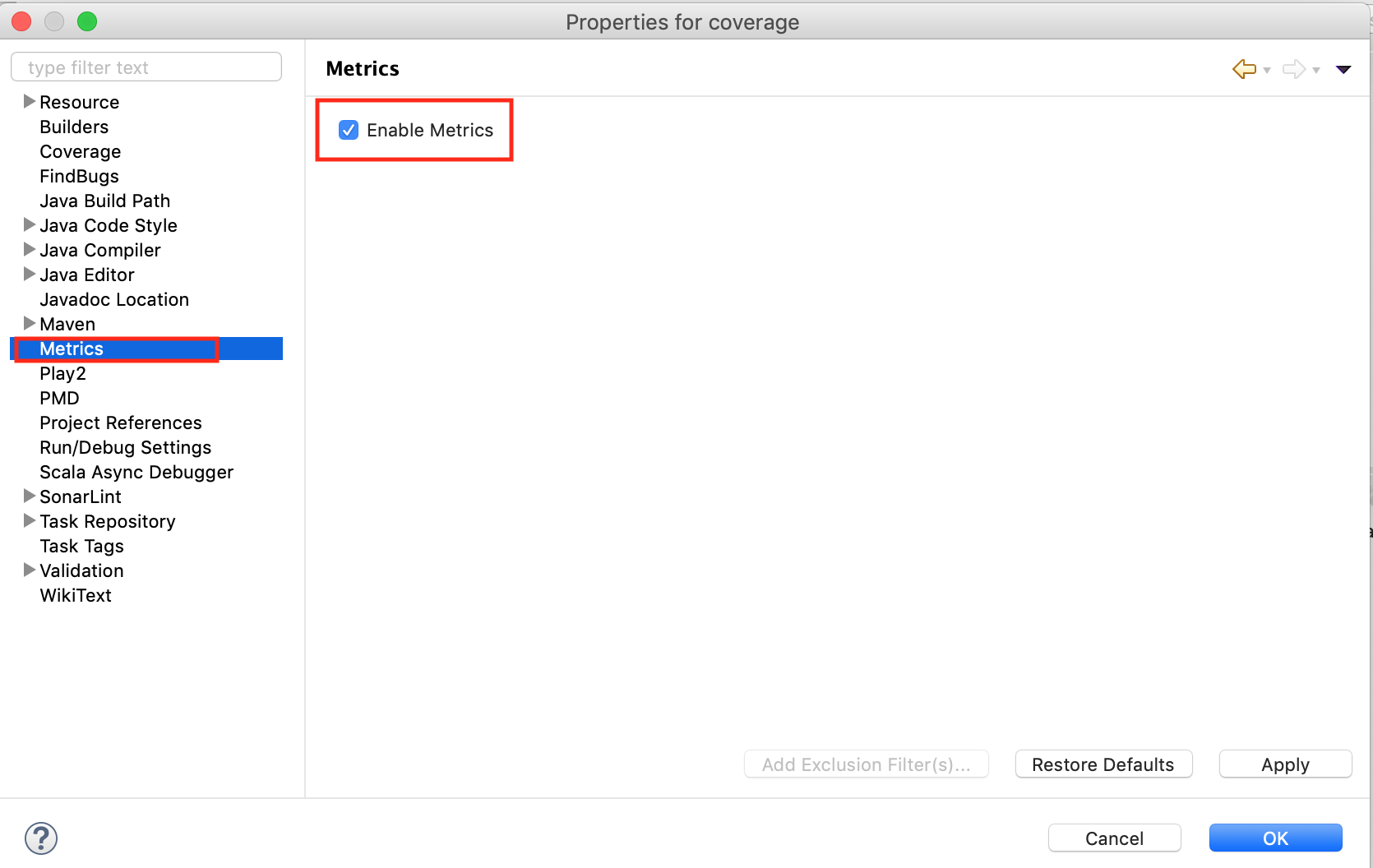
**Task 1: Measure your code size and complexity using Eclipse Metrics Plugin**

**Get the Tool First!**

[**Metrics Plugin**](http://metrics2.sourceforge.net/)can collect various source code size and complexity metrics including Line of Code (LOC), Number of Methods, and Cyclomatic Complexity ([CC] or [VG]).

You need to install the plugin into your Eclipse environment, and then activate the plugin so it reads your project..

1. Follow the installation instructions in the following page: <http://metrics2.sourceforge.net/>
2. After installation, make sure that you activate the plugin for your project by **project name-> right-click ->** **Properties -> Metrics -> Enable Metrics**

****

1. Make sure to open Java perspective by **Window ->Open Perspective->Java**
2. Make sure that you have the correct Metrics View from **Windows -> Show view -> Metrics view**

**Task**

1. Generate the metrics reports for Apache Commons Lang and JDepend in XML format. You can click on export icon  in Metrics window to save it in XML format.
2. In a seperate .txt file , please note the following values:

number of packages

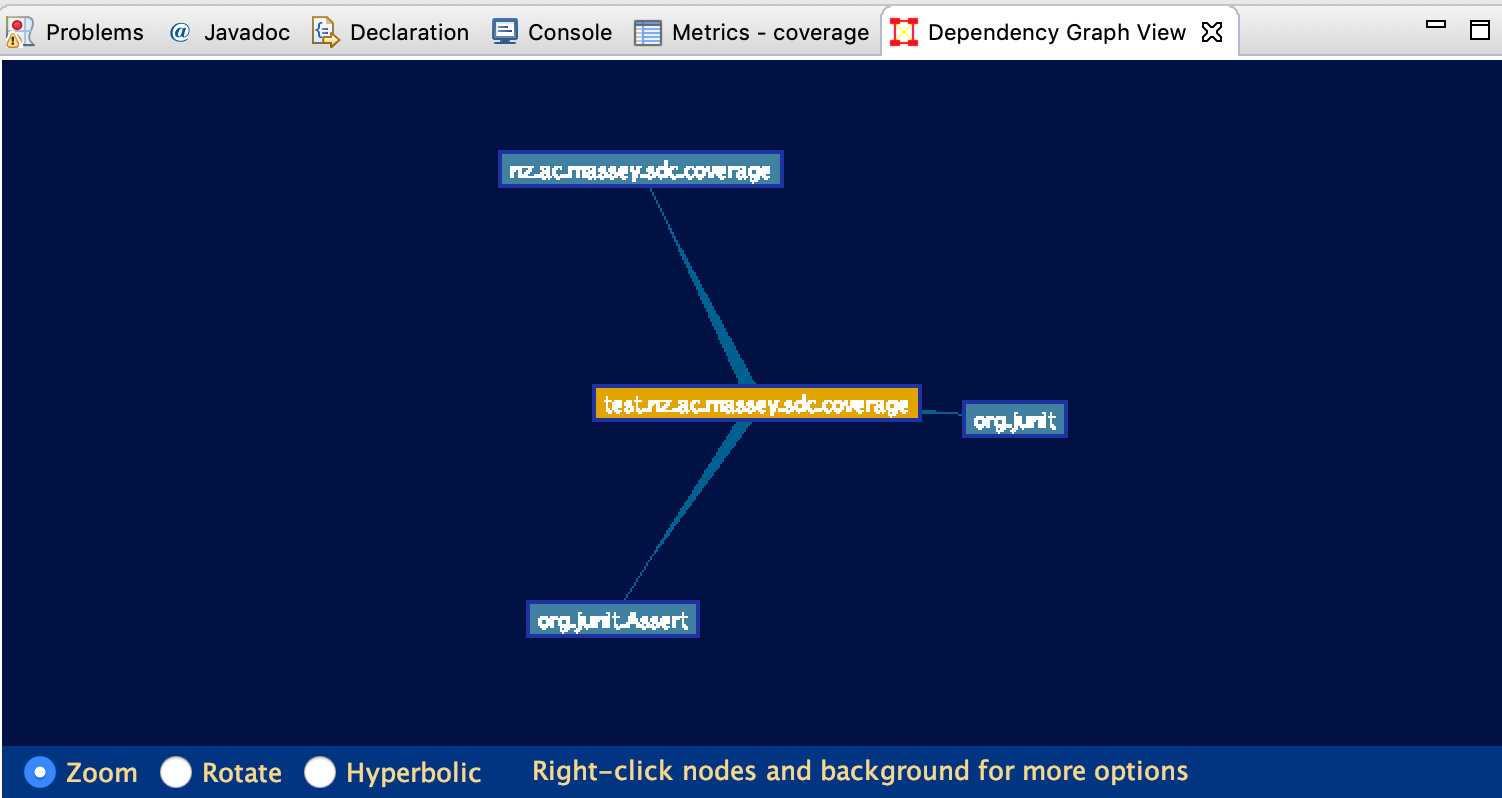
number of classes (average and max).

number of methods (average and max).

name of largest class file (.java) in each project based on LOC value.

**What to submit**

* Create a folder named Metrics and save the documents mentioned below in this folder.
* Save your results as a .txt file named as Task1.txt . This file should be submitted together with the XML reports that you generated for both projects.
* Take a screenshot of the dependency graph by clicking on  in Metrics view and include this in your submission folder. An example of a dependency graph is shown below.

****

**Task 2: Measuring code quality using PMD**

**Get the Tool First!**

[PMD](https://marketplace.eclipse.org/content/eclipse-pmd) is a popular source code analyser that statistically checks for potential bugs and flaws in your code. It finds problems and issues in your code caused by programming behaviour and coding style. You will need to install PMD Eclipse plugin to analyse the source code of the two projects above. Go to [PMD](https://marketplace.eclipse.org/content/eclipse-pmd) webpage to study more about this plugin and install it.

**Task**

Run this by only using the Java setup as shown in PMD options (PMD support other languages as well such JavaScript).

Generate two reports (one for each project) that contain the results of your analysis on both projects. Once the reports are generated, the output files can be obtained from <Project\_Home>/reports.

**What to submit**

* Create a folder named PMD and save the documents mentioned below in this folder.
* Submit your HTML report and highlight the most useful values from the report.
* Add this to a separate .txt file named Task2.txt and explain why you picked up those values.

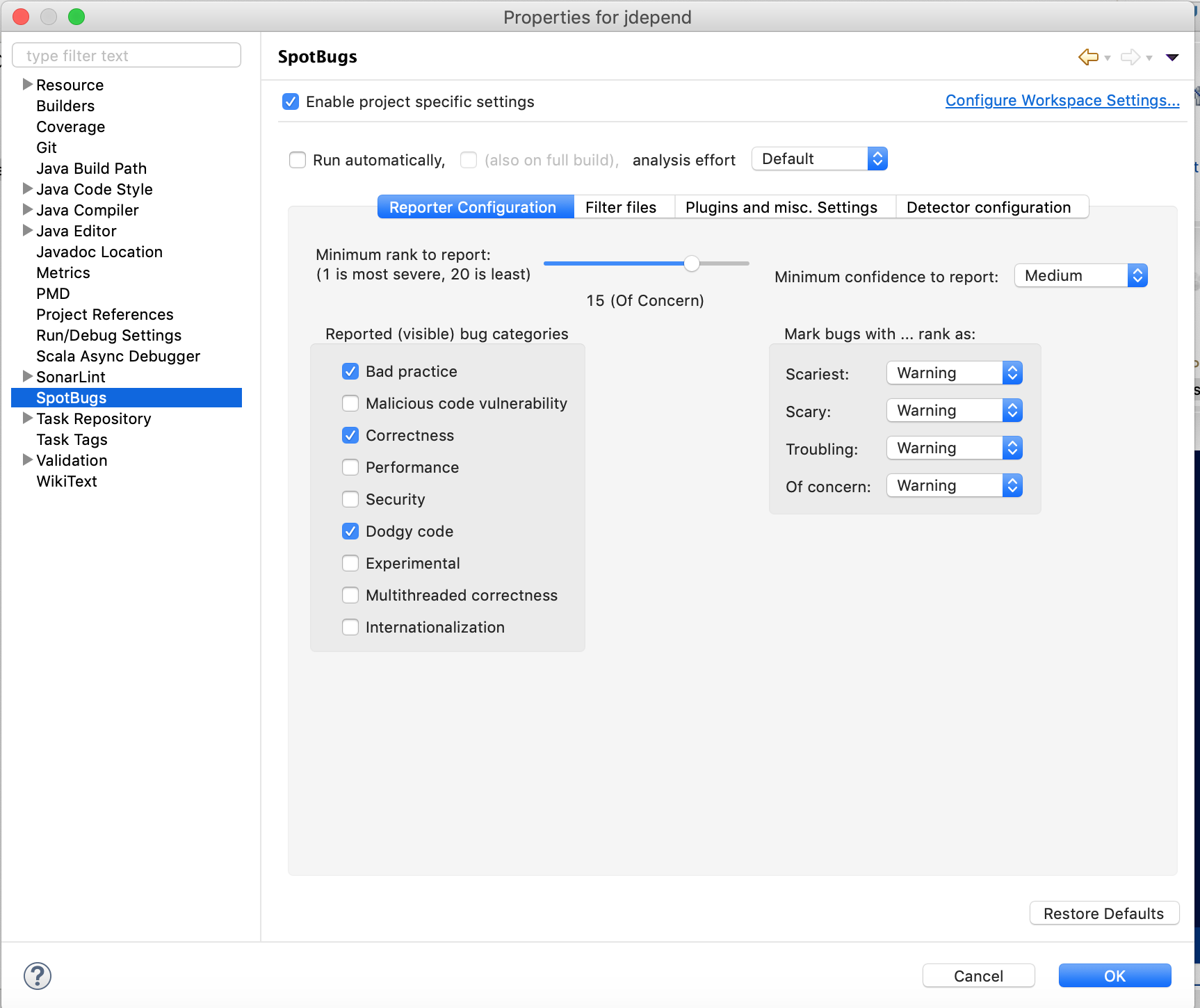
**Part 3: Use spotbugs to detect possible bugs in your code**

**Get the Tool First!**

[Spotbugs](https://spotbugs.github.io/) (previously Findbugs) is a powerful static code analyser that detects possible bugs and issues in your Java code. This includes issues such as Dead Code and complex code. You need to install Spotbugs Eclipse plugin following the same steps that you followed for the previous two plugins.

**Task**

Your task is to learn how to do the analysis yourself! Read the manual here ([https://spotbugs.github.io](https://spotbugs.github.io/)). FInd issues in the following categories: Bad Practice, Correctness and Dodgy Code only (see below)



**What to submit**

* Create a folder named Spotbugs and save the documents mentioned below in this folder.
* You need to produce an XML report of the violations in the three categories that you found in both projects. Please save all the reports in Spotbugs folder

**Tutorial 3 Submission Format**

**Once done with analysis, create a folder Tutorial3 and keep all three folders Metrics, PMD and Spotbugs in this folder (don't forget to copy files within the sub-folders too ;) )**

**Zip the folder ‘Tutorial 3’ and submit the file Tutorial3.zip on stream.**