Software Metrics Assignment

Student Name:

Submit typed solutions with questions followed by your answers along with your "Metrics.java" implementation. You may discuss the assignment with your peers, but the assignment should be completed individually.

Overview

In this assignment you will be implementing a few software package metrics for evaluating software code quality. You will be implementing method stubs outlined for you in the *Metrics.java* class. The *SoftwareMetrics* project provided to you is an Eclipse plugin project that adds a "Software Package Metrics" view to Eclipse. Once you have installed the Eclipse plugin (see section below) you can open the view by navigation to *Window*->*Show View* ->*Other…*->*Software Metrics*->*Software Package Metrics*. To use the view, you must have a project in the workspace and have indexed (mapped) the project. The view listens for selection events on project packages. When you select a package in an indexed application the plugin view will show you the software metrics that you have computed.

Once you have finished implementing the *Metrics.java* methods you should complete the attached questions included in this assignment. Specifically you will be asked to evaluate the quality of code in the Apache Commons IO library (included as an Eclipse project for your convenience).

Submit your *Metrics.java* implementation and your answers to the questions below. It is ok to work with others in a group, but everyone must do their own work.

Installing the Software Metrics Eclipse Plugin

With the Software Metrics project in the workspace, right click on the project and select *Export*. Select *Plug-in Development*->*Deployable plug-ins and fragments*. Select the *Install into host. Repository:* radio box and click *Finish*. Press *OK* for the notice about unsigned software. Eclipse will restart and the plugin will be installed. To install again with new changes to the plugin, repeat this process.

Testing, Debugging, and Other Hints

You may find it helpful to debug the SoftwareMetrics project on the Atlas Shell. To invoke methods in *Metrics.java* from the Atlas Shell, add the *SoftwareMetrics* project as a dependency to any Atlas Shell project by editing the MANIFEST.MF file located in the META-INF folder of the shell project. For your convenience a shell project with a dependency on the *SoftwareMetrics* project is provided for you in this repository.

To open the Atlas Shell navigate to *Window*->*Show View*->*Other…*->*Atlas* ->*Atlas Shell* and select the appropriate project. In the Shell window enter "import metrics.Metrics" to import the Metrics class.

You can now call *Metrics* methods such as *Metrics.countNodes(…)* in the Shell window. If you wish to call Metrics methods directly, enter "import metrics.Metrics._" to import all Metrics methods. You will need to repeat this step when restarting the Shell. Remember that you can use the "selected" keyword on the shell to define a query with the current source text or graph selection.

If you change code in the *SoftwareMetrics* project, you must save your code and restart the Shell (otherwise you may be running a stale version of your implementation)!

If you are unfamiliar with Atlas, refer to the Atlas Help (*Atlas*->*Open Atlas Help*) menu and http://www.ensoftcorp.com/atlas/.

You may find that indexing the Apache Commons IO library and all of the Java runtime libraries require too much memory for your machine. For this homework you should have Atlas set to map only the jars that are used by the application. Navigate to *Window->Preferences->Atlas->Map only used classes in and* press *Apply*. By default *Map only used classes in jars* is enabled.

If you are not receiving selection events in the Software Package Metrics view, make sure you have indexed your workspace and that the projects you are selecting packages in have been indexed by Atlas (see *Atlas*->*Manage Project Settings*). If you are still have problems try closing the Software Package Metrics view and re-opening it again.

If you have compile errors when you import the project, make sure you are running Atlas version 2.7.3 of Atlas (see *Help->Installation Details*). If you do not have version 2.7.3 or later installed upgrade your installation by navigating to *Help->Check for Updates*. If you do not have Atlas installed visit http://www.ensoftcorp.com/atlas/download/. Note that academic licenses are available for free to students.

Questions

Part 1. (10 points) Upload "Metrics.java" as an attachment during submission.

Part 2. Please answer the following questions in your homework submission.

1) **(1 point)** In the Apache Commons IO project what package is the most abstract?

2) **(1 point)** In the Apache Commons IO project what package is the least abstract?

3) **(1 point)** In the Apache Commons IO project what package is the most stable?

4) **(1 point)** In the Apache Commons IO project what package is the least stable?

5) **(1 point)** With regard to the Distance (D) from the Main Sequence is the org.apache.commons.io package well designed? Support your answer.

6) **(1 point)** With regard to the Distance (D) from the Main Sequence is the org.apache.commons.io package better or worse compared to the org.apache.commons.io.filefilter package? Support your answer.

7) **(2 points)** Do you feel these metrics accurately capture the quality of this code? If yes, explain. If no, what other metrics or code properties should be considered to evaluate the quality of the library?

8) **(2 points)** The SetDefinitions.java defines a set "SetDefinitions.app()" that represents all program declarations and their relationships inside the application excluding program artifacts that are found in the Java APIs or other Jar libraries. What effect does this have specifically on the Metrics.getEfferrentCouplings() method? Would the metric still be useful if packages in the Java APIs (such as java.lang or java.math) were included?