# Software Metrics Assignment

Solutions

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

**Part 2.** Please answer the following questions in your homework 5 submission.

1) **(1 point)** In the Apache Commons IO project what package is the most abstract?
   Indexing Jars: org.apache.commons.io.monitor
   No Jar Indexing: org.apache.commons.io.monitor

2) **(1 point)** In the Apache Commons IO project what package is the least abstract?
   Indexing Jars: org.apache.commons.io
   No Jar Indexing: org.apache.commons.io

3) **(1 point)** In the Apache Commons IO project what package is the most stable?
   Indexing Jars: org.apache.commons.io.input
   No Jar Indexing: org.apache.commons.io.output

4) **(1 point)** In the Apache Commons IO project what package is the least stable?
   Indexing Jars: org.apache.commons.io.filefilter
   No Jar Indexing: org.apache.commons.io.filefilter and org.apache.commons.io.monitor

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.

   Indexing Jars: D = |A+I-1| = |.04+.55-1| = |.59-1| = |-.41| = .41
   No Jar Indexing: D = |A+I-1| = |.04+.55-1| = |.59-1| = |-.41| = .41

   Distance (D) ranges from 0 to 1 where 0 is directly on the main line. This package has a distance of .41 which is closer 0 than not. With respect to distance from the main sequence this package could certainly do better, but a correct answer could lean in either direction (well designed or not well designed) if it is well supported.

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.

   D_io = .41
   Indexing Jars: D_filefilter = |A+I-1| = |.12+.67-1| = |.79-1| = |-.21| = .21
   No Jar Indexing: D_filefilter = |A+I-1| = |.12+.57-1| = |.69-1| = |-.31| = .31

   With respect to the distance D from the main sequence, org.apache.commons.io is worse designed than org.apache.commons.io.filefilter, since the org.apache.commons.io.filefilter package is significantly closer to 0 than the org.apache.commons.io package.

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?

Code metrics are an approximation that captures some properties of software. They are useful for evaluating software, but do not capture everything necessary to evaluate the quality of software. In particular these metrics do not assess whether the algorithms in the software are appropriate or the code is readable.

Other answers are acceptable, but should include support.

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?

Including the Java APIs would be less useful because the Efferent Couplings would include the dependencies inside the standard libraries (which all programs make use of). The result would skew the metrics significantly and would cause difficulty interpreting results. One could argue, however, that the metrics would still be useful for comparing two packages relatively.