CST316 Software Enterprise: Construction and Transition **Spring 2016 online**
Practice: Software Metrics **February 21, 2016**

Objectives
1. Understand the relationship of metrics to quality code.

Preliminaries:
1. You will apply this to code from the JUnit lab given to you earlier in the semester. You may download and expand this code in an empty directory, and then create a new Eclipse project using this source code, or you may already have it in your Eclipse from earlier in the session. Please use the given version of the code, not your modified solution code from that lab, or you will get the wrong values! I understand this code has defects in it but that is not really relevant to this lab!
2. Download and install the Eclipse metrics plugin from http://metrics.sourceforge.net/update. Once installed, right click in your project and select "Properties" and then "Metrics". Check the *enable* option and click "OK". Then bring up the Metrics View using the "Window → Show View →Metrics → Metrics View". Then force a rebuild the project by unchecking "Project → Build Automatically", "Project → Clean", and "Project → Build Project". You should see a table-tree widget in the Metrics View at the bottom with output of various measures and metrics (it takes a minute to calculate and show up). If it doesn't show you may have to click on the green "play" arrow in the lower right in the view to run the metrics. Take an initial export to XML via the export widget upper-right of the pane and save it as cst316_<asurite>_metrics_initial.xml.

Activity: Evaluate metrics
The metrics produced by this plugin include complexity (*McCabe Cyclomatic Complexity*), cohesion (*Henderson-Sellers Lack of Cohesion or LCOM2*), and coupling (*Afferent and Efferent Coupling)*. Unfortunately I have not found a way to configure the plugin only to output the metrics you want, so you have to hunt through the output. Using the tool, please answer (in Word) the following:

1. **(10) Size**:
   a. What is the Total Lines of Code (LOC) in the project?
   b. What is the the largest Java package in the project and its Total LOC?
   c. What is the the largest single code file in the project and its Total LOC?
   d. Inspect Main.java - what method did the Metrics tool use to determine Total LOC? Describe the method.

2. **(20) Coupling:**
   a. Package-level Coupling:
      i. What do Afferent and Efferent coupling mean? Look these terms up on Wikipedia and summarize the distinction.
      ii. What package has the worse Afferent Coupling measure?
      iii. What package has the worse Efferent Coupling measure?
   b. Coupling. For the AccountServer interface, please compute the following (the tool will not do these for you!):
      i. Method Fan-in – The number of other elements in the code depending on this type.
      ii. Method Fan-out – The number of other unique (non-primitive) types this interface depends on.
      iii. Question: Do you think the fan-in and fan-out counts are good or bad for this code? Explain.

3. **(25) Cohesion:**
   a. The tool calculates "Lack of Cohesion of Methods" (LCOM) using the Henderson-Sellers method, or what is commonly referred to as *LCOM2* (there are LCOM1 through LCOM4). What is the definition of LCOM2 and how is it calculated?
   b. According to the LCOM, there are 2 classes in the code with > 0 cohesion values. What are they and what are the values?
   c. Considering how LCOM2 is calculated, identify why these 2 classes have nonzero values (you do not have to try and fix, you just have to indicate why there is a nonzero value for each).

4. **(25) Complexity**: Make sure you do this after 1-3 so as to not distort those answers!!!
   a. What is the cyclomatic complexity of the banking.primitive.core package*)*
   b. What class has, on average, the worst McCabe Cyclomatic Complexity (CC)?
   c. The specification of the current "state" of an Account is tied to its balance. Yet in several places throughout the code there are checks of the balance to set the Account state. Can you refactor the code to reduce the need for these checks everywhere?
   d. What is the resulting cyclomatic complexity of the banking.primitive.core package after your refactoring in (b)?

5. **(10)** Which package and which class have the worst quality and why? Us the metrics to support your answer.

6. **(10)** Describe how Measures and Metrics can support Refactoring in an Agile context.

Submission: A jar[zip]file named labmetrics.<asurite>.jar[zip] submitted to Blackboard by 2/28 11:59:00pm with the following:
1. The initial XML export file named cst316_<asurite>_metrics_initial.xml, and final XML export you should do after completing step 6 named cst316_<asurite>_metrics_final.xml.
2. A Word or text document with your answers to the questions I have asked throughout this lab. Please label them by question number (1-6) and sub-numbers (e.g. a,b,…i,ii…).
3. The new version of the code (src directory) from your refactoring in 4.c.