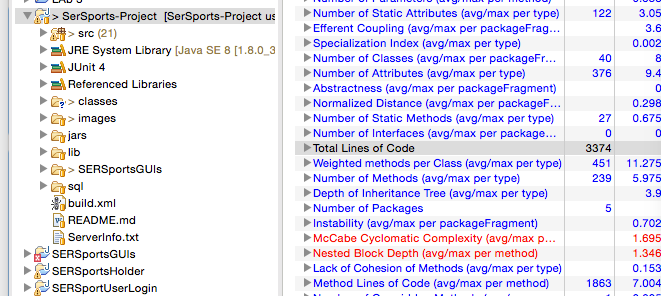
1. **Size**:
   1. Total Lines of Code in Full project is 3374

****

* 1. GUI package had 1170 lines of code

1. **Complexity**:
   1. What package has, on average, the worst McCabe Cyclomatic Complexity (CC)?
   2. What class has, on average, the worst McCabe Cyclomatic Complexity (CC)?
   3. The method banking.gui.MainFrame.BalanceChecker.focusLost() has a high CC. Inspect this method, determine what it does (run the application), and rewrite the method to have equivalent functionality but a lower CC.
2. **Coupling:**
   1. Package-level Coupling:
      1. What do Afferent and Efferent coupling mean?
      2. What packages are the “worst” according to the Afferent/Efferent Coupling metrics?
   2. Method-level Coupling. For the methods banking.server.ServerSolution.update() and banking.primitive.core.Account.display(), calculate the following metric values **by hand** (the plugin will not do these for you, however you can get Fan-In by using Eclipse to find the References to the methods in your project):
      1. Method Fan-in – This is the number of other methods that depend on this method.
      2. Method Fan-out – The number of other types/methods (excluding constructors) this method depends on.
      3. Question: Which of these 2 methods do you think would be more problematic to change? Why?
3. **Cohesion:**
   1. 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? How is it calculated?
   2. According to the LCOM, there are 2 classes in the code with > 0 cohesion values. What are they and what are the values?
   3. 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).

Finally, please discuss the following:

1. Which metric or combination of metrics are the most intuitive? Most useful?
2. Which package and which class have the worst quality and why?

**Submission:** A jar[zip]file named labmetrics.<asurite>.jar[zip] with the following:

1. An export XML of the metrics tool on the given lab code.
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 focusLost method from 2.c. Don’t give us your entire project code, just give the new MainFrame.java file.

*This is an individual submission!*

**Going forward with your projects:**

Metrics are yet another way to indicate potential quality problems in your code, but their nature is much different than static analysis, unit tests, and code reviews. First, they try to boil things to numbers, and it can be hard to define meaning in those numbers. Second, they can often be non-specific; that is, not specific to a line or small lines of code.

As always, in your projects I expect you to utilize metrics at as indicator of *potential* problems. In your quality, define the complexity, coupling, and cohesion numbers that raise red flags, and then decide what your course of action is (fix? code review? correlate to static analysis?) and provide evidence you are executing that strategy. We will not be integrating metrics with Jenkins, so you will have to also state how often you will run metrics manually.