# Activity: Evaluate Metric

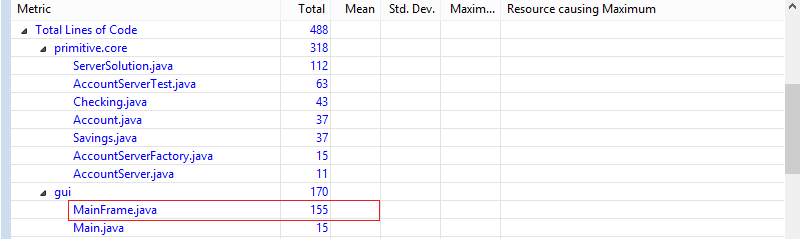
1. **(10) Size**:
   1. The Total Lines of Code (LOC) in the project is 488.



* 1. The largest Java package in the project is primitive.core. Its Total LOC is 318.

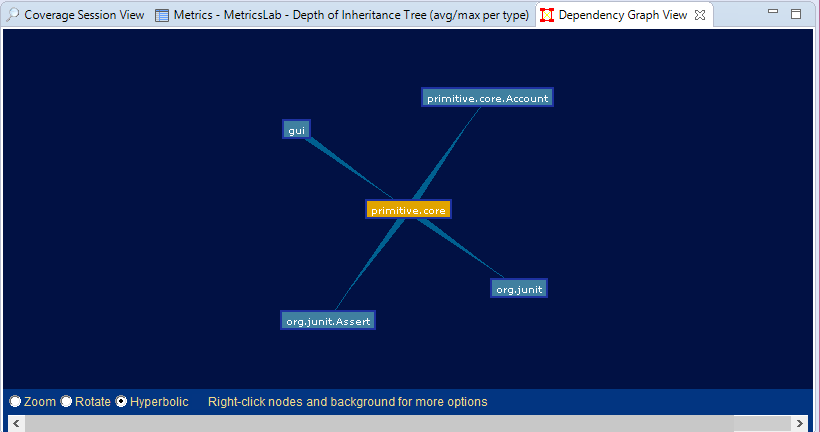


* 1. The largest single code file in the project is MainFrame.java. Its Total LOC is 155.



* 1. The Metrics tool used Logical LOC method without consideration for LOE. There were 15 logical LOC, including curly braces.

1. **(20) Coupling:** 
   1. Package-level Coupling:
      1. Afferent and Efferent are metrics of responsibility (afferent) a particular package, and the classes within, have for other classes in the other packages, and the dependency (efferent) a particular package, and the classes within, have for other classes in the other packages.
      2. Worst Afferent Coupling Measure: primitive.core.
      3. Worst Afferent Coupling Measure: primitive.core.



* 1. Coupling. For the AccountServer interface, please compute the following (the tool will not do these for you!):
     1. Method Fan-in – 3.
     2. Method Fan-out – 1.
     3. Question: The Fan-In and Fan-Out counts are not fair in this code because it is an interface, and interface provides decoupling. So, these metrics are not quite applicable in this code.

1. **(25) 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). Definition of LCOM2 [from [http://www.aivosto.com/project/help/pm-oo-cohesion.html#LCOM4](http://www.aivosto.com/project/help/pm-oo-cohesion.html%23LCOM4)]:

m number of methods in a class

a number of variables in a class

mA number of methods that access a variable

sum(mA) The sum of mA over the variables of a class

Then, LCOM2 = 1 – [sum(mA) / (m\*a)].

LCOM2 equals the percentage of methods that do not access a specific attribute averaged over all attributes in the class. If the number of methods or attributes is zero, LCOM2 is undefined and displayed as zero.

* 1. According to the LCOM, there are 2 classes in the code with > 0 cohesion values. What are they and what are the values?

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Class Name | Account | AccountServerFactory |
| Cohesion value | 0.23 | 0.33 |
|  |  |  |

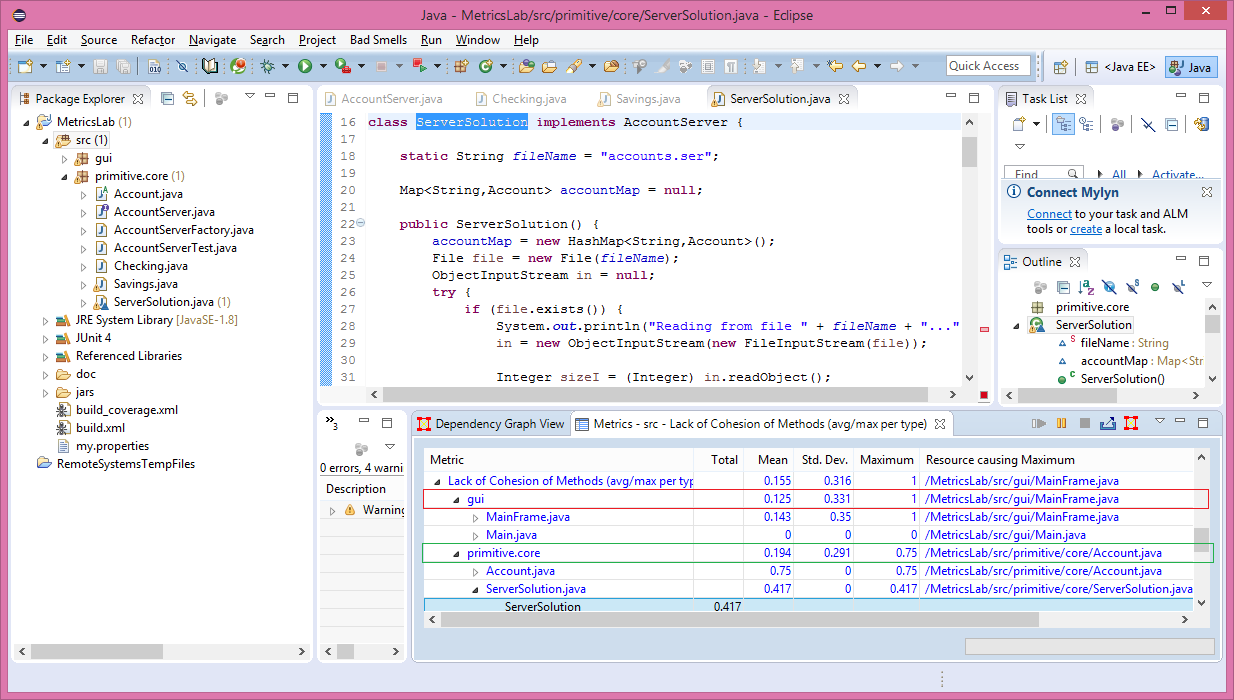
* 1. The reason is that they share the same attributes (variables) with too many methods within these classes.

1. **(25) Complexity**: Make sure you do this after 1-3 so as to not distort those answers!!!
   * 1. The cyclomatic complexity of the primitive.core package is 1.829.
     2. The ServerSolution.java class has, on average, the worst McCabe Cyclomatic Complexity. (2.875).
     3. 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?

I’ve refactored by having a method called *SetState(float Balance)*.

* + 1. The resulting cyclomatic complexity of the primitive.core package after my refactoring in (b) had reduced to 1.744.

1. **(10)** The gui package has lower quality level due to lower LCOM number than the primitive.core package.



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

With respect to the Agile software development process, Measures and Metrics helped code Refactoring by providing an immediate indicator for improvement expectation. This indicator is meaningful because developer can see their result immediately to form a decision for acceptance or rejection of the refactoring effort.