1. (10) Size:

**a. What is the Total Lines of Code (LOC) in the project?**

The Total Line of Code in the project is 488

**b. What is the the largest Java package in the project and its Total LOC?**

The largest Java package in the project is primitive core and it has 388 lines of code.

**c. What is the the largest single code file in the project and its Total LOC?**

The largest single code file inthe project is the server solution file and it has 112 lines of code.

**d. Inspect Main.java - what method did the Metrics tool use to determine Total LOC? Describe the method.**

The Method seem to be empty so i can't determine the method at the moment.

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**

Efferent: Measure of how many classes are being use by a class.

Afferent: Measure of how many classes are use by the same class

**ii. What package has the worse Afferent Coupling measure?**

If any of the classes breaks, the packages fails

**iii. What package has the worse Efferent Coupling measure?**

If the package breaks, all classes breaks.

**b. Coupling. For the AccountServer interface, please compute the following by hand (the tool will not do these for you!):**

**i. Method Fan-in – The number of other elements in the code depending on this type.**

11

**ii. Method Fan-out – The number of other unique (non-primitive) types this interface depends on.**

3

**iii. Question: Do you think the fan-in and fan-out counts are good or bad for this code? Explain.**

I believe this is a good thing for the code. Show that the code can be reuse many times. I believe reuse is a great idea.

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?**

LCOM2 is define as a directed edge connects a method node to an attribute node if the method references the attribute.

Calculation is LCOM2 is equal to the ratio of method that not accessing a specific class divided by the number of procedures and variable in class.

LCOM2= 1 - sum(mA)/(a\*m)

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

MainFrame with 0.143

Account with 0.6

**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).**

To my best understanding, I believe it it because these both file have many attributes and method that they have a nonzero value.

**4. (35) 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)**

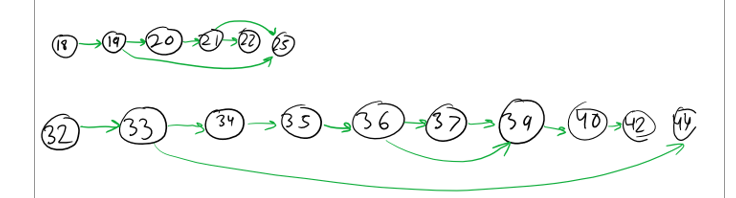
To the best of my understanding, the cyclomatic Complexity mean is 1.829. the Std.Dev is 1.637 and the Maximum is 7.

**b. What class has, on average, the worst McCabe Cyclomatic Complexity (CC)?**

I believe the ServerSolution has the worst Cyclomatic complexity since if it has the highest average, it

The worse buggiest and the most complexity.

**c. In your Unit Testing lab you were asked to create the control flow graphs for Savings.deposit() and Savings.withdraw(). Revisit (or recreate) your control flow graph from that lab for these 2 methods, and calculate McCabe’s Cyclomatic Complexity for each by hand. Now compare what you got to what the Eclipse plugin calculates. Why do you get a different value?**



E – N + p

6 – 6 + 3 = 3

10 – 10 + 3 = 3

I believe it may be of a human error, it might be using the tools completely wrong.

**d. 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 modify the code to reduce the need for these checks everywhere?**

**e. What is the resulting cyclomatic complexity of the banking.primitive.core package after your modification in (b)?**

The Cyclomatic Complexity of the Banking.primitive.core package after the modification, Mean has reduce to 1.707 and the STD.Dev is reduce to 1.384. The maximum is still seven

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

I believe the banking.gui has the worse quality for its average is 2 while the banking.primitive.core average is 1.707.

The ServerSolution Class has the worse quality for its mean is higher the others in the banking gui.