Task 1:

Size

1. In the total project there are 22539 LOC in main.memoranda.java
2. HTMLEditor.java is the single largest with 2144
3. Measured by Physical Line of Code method

Cohesion

1. LCOM2 is the percentage of methods that do not access a specific attribute averaged over all attributes in a class. LCOM2 = 1-sum(mA)/(m\*a) where sum(mA)= sum of methods that access a variable over all variables of a class. m = number of methods in class. a = number of variables in class.
2. CharTablePanel.java has the highest LCOM2 because it is used by every panel in the application. All characters are pulled against this class that are in a panel.

Complexity

1. 16 is the maximum and the mean is 1.7
2. ProjectPackager.java at 5.667
3. In EventsManager.java I eliminated a null check on the date because it will check for null within the for loop. I reduced the complexity from 5 to 4 because it was one less branch. It already returns null if it doesn’t find a value in the for loop so didn’t need to return null in the beginning.

Package-level Coupling

1. Efferent couplings- The number of classes in other packages that the classes in a package depend upon is an indicator of the package’s dependence on externalities.

Afferent couplings- The number of classes in other packages that depend upon classes within the package is an indicator of the package’s responsibility.

The difference would be efferent is the number of classes that your class is dependent on to run. Afferent is the number of classes that are dependent upon you class to run.

1. Main.java.memoranda.util is 57
2. Main.java.memoranda.ui is 49

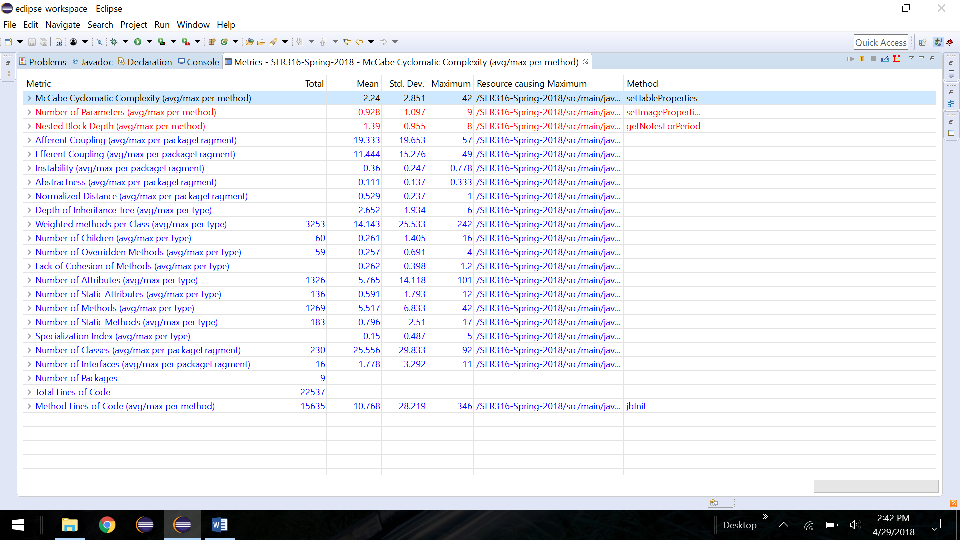
Worst Quality

1. HTMLEditor.java would be the lowest quality class in the project in my opinion. It has the highest cyclonic complexity by far so it is very easy for it to be wrong and difficult to troubleshoot the issues. The package it is in has the highest Afferent coupling meaning the rest of the project is more dependent on this package being correct than any other package.

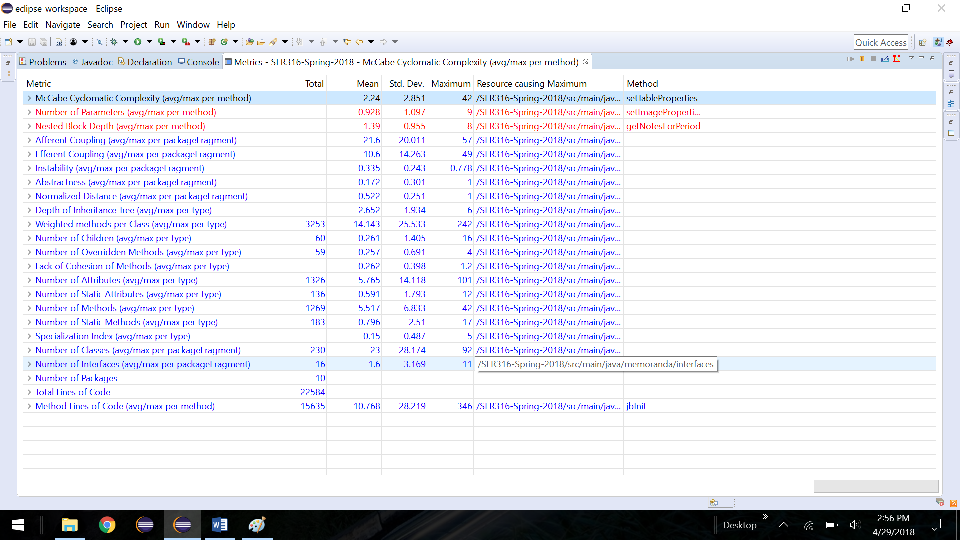
Its methods have a high average LOC at 18.899 meaning that the methods are fairly complex and it has one of the largest methods in the project at 277 LOC.

Task 2

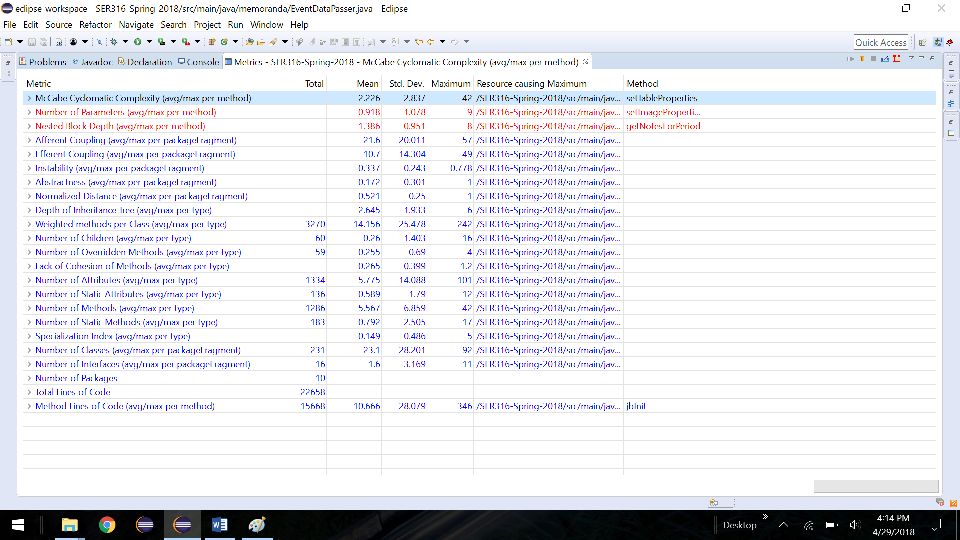
Step 1

1. 

After

1. 
2. Efferent on the project got better. Efferent got better because it is a measure of averages and since we split up the amount of dependencies between different packages it makes the overall look better.

Task 3

1. Eliminated duplicate code in the createProject method of the ProjectManager.java class. there were two methods to create a project one with an ID and one without so I eliminated the one without and checked for empty ID.
2. I added a class to pass data between the eventspanel and the event creation to eliminate the need for 8 parameters in the creation method and eliminate Primitive Obsession. I combined the relative info into one object to be able to maintain how it is handled
3. 
4. The average number of parameters went from 3.928 to 3.918.

The depth of inheritance went down from 2.652 to 2.645

LCOM went from 3.262 to 3.265

Average number of parameters went down due to the fact that I created an intermediate object to handle the parameters for event creation and eliminated the long method parameter.