**Task 1**

Size

1. 2187
2. EventsManager
3. It used the get() method because it is the first method.

Cohesion

1. It’s the number of pairs of methods that do not share attributes minus the number of pairs that do. If that difference is greater than 0 then the answer is LCOM2. If the answer is negative, the answer is 0.
2. There are about many classes that have 0 for lack of cohesion wich means they have high cohesion, right? Sure! I don’t know why the are so cohesive. So I’ll pick task.java

Complexity

1. The complexity is 1.746
2. EventsManager. It is 2.5
3. EventImpl Line 139 getWorkingDays. It had an if statement that chose wether the Boolean return statement was true of false. I deleted the “if” and just returned the answer of the former if statement (the part in the parenthesis). The method dropped from a 3 on complexity to a 1.

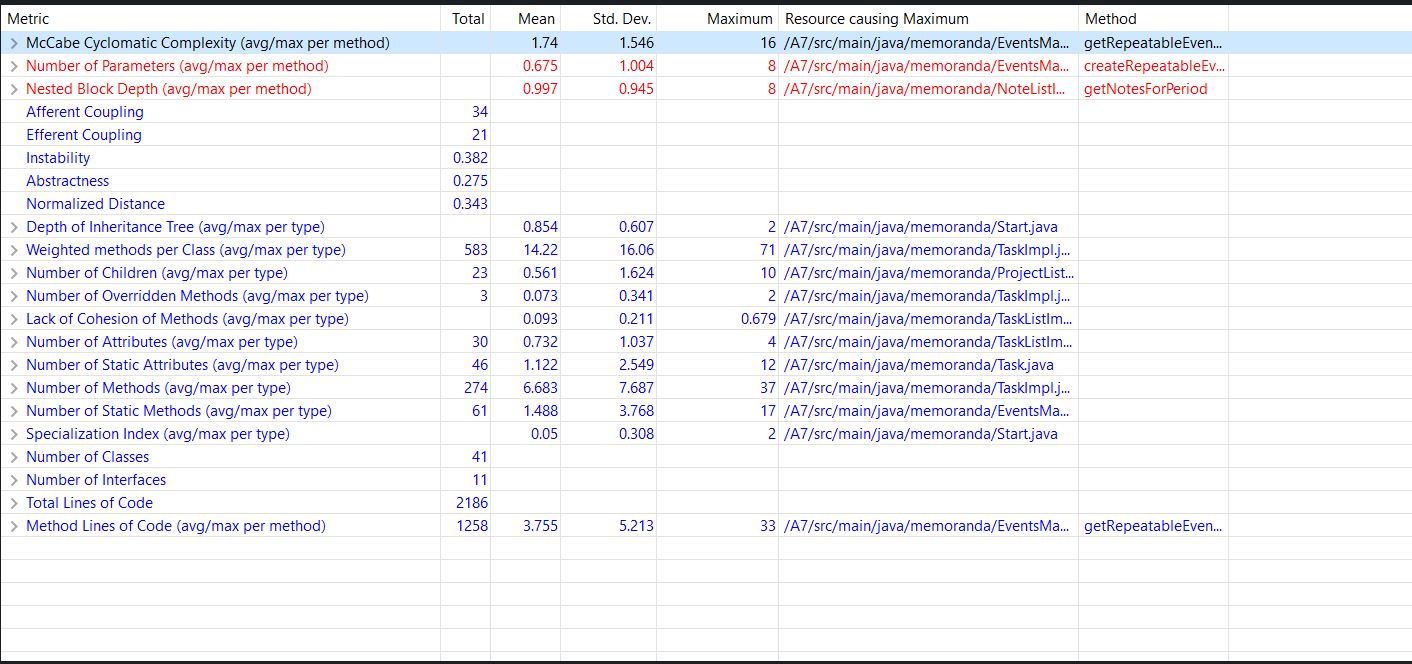
Package-level Coupling

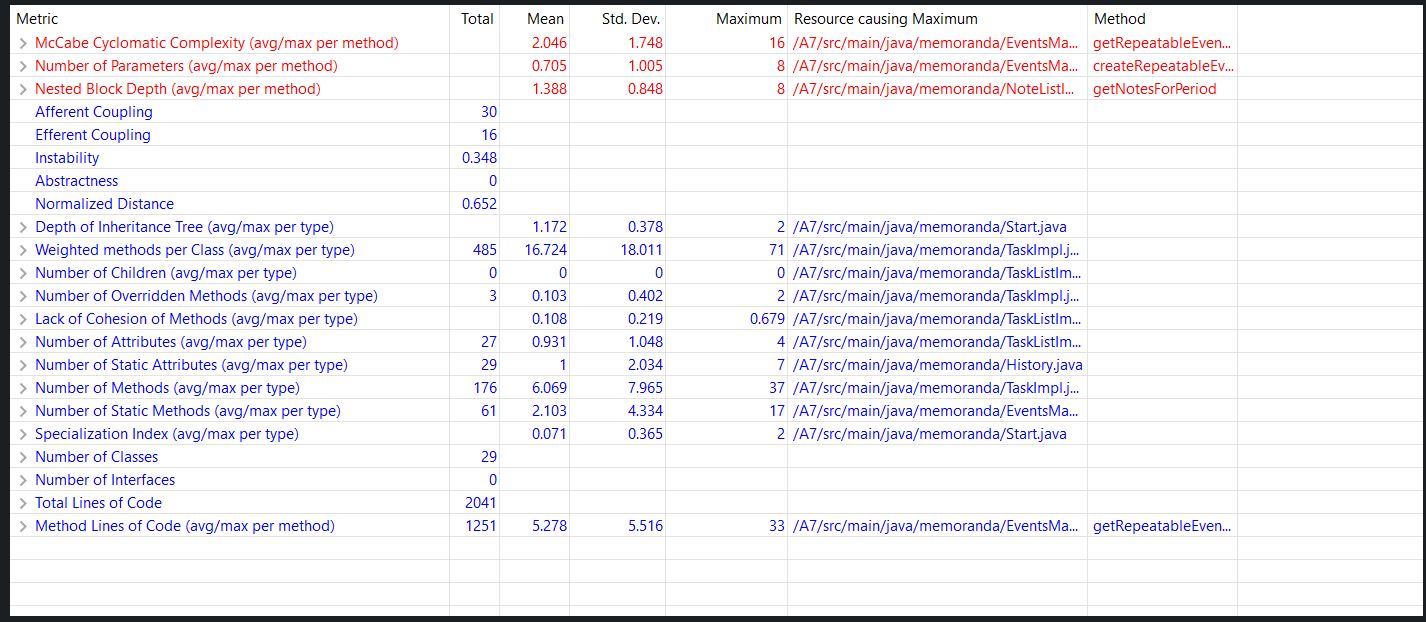
1. Afferent coupling are coupling where classes outside of the package need the classes within a packages. Efferent is when the classes inside the class need classes outside the class.
2. Main.java.memoranda.util. its at 57
3. Main.java.memoranda.ui its at 49

Worst Quality

1. EventManager because it has the highest complexity at a mean of 2.5. It also has a high number of parameters at 8. It had the most red (bad) lines in the Metrics window.

**TASK 2**

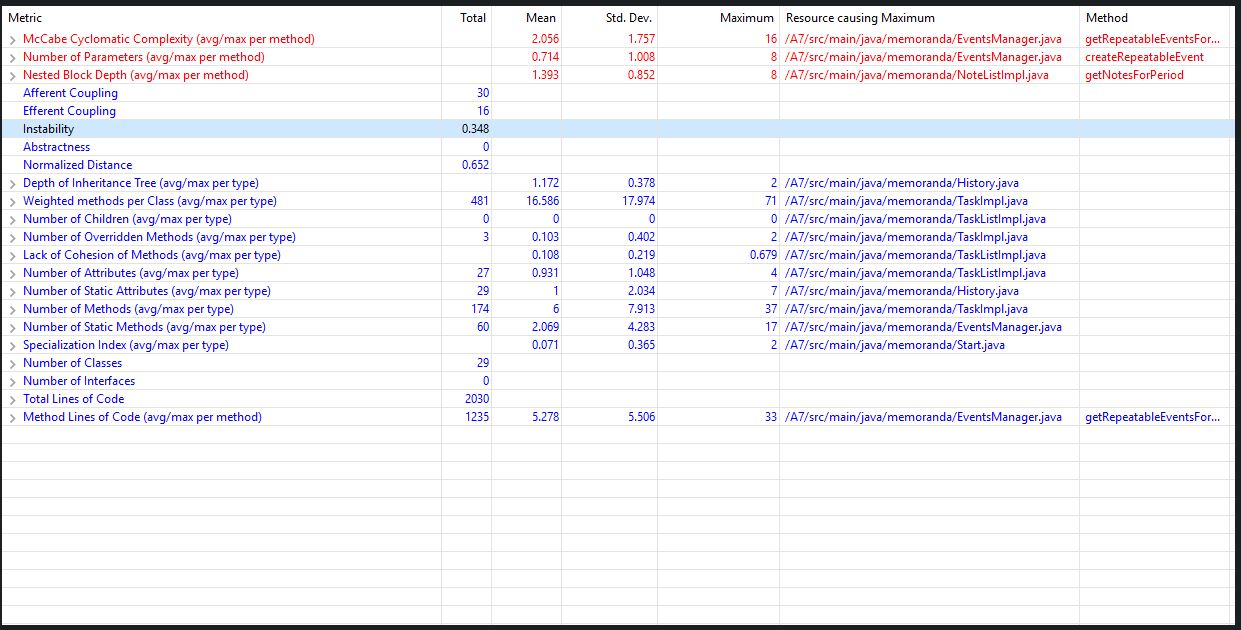




Efferent couplings decreased and I believe that is a good thing. I think the more external dependencies a package has, the more opportunity for things to go wrong…like machinery with moving parts.

**PART 3**

1. Code Smell Within a Class: Assign7\SER316-Spring-2018\src\main\java\memoranda\ TaskListImpl.java Line 341. This was an example of duplicate code. The interface requires a getTopLevelTasks method. In the class, the getTopLevelTasks only returned the private method getAllRootTasks. That method was not required and there was no benefit to having it private. I renamed getAllRoottasks to getTopLevelTasks and made it public and deleted the former getTopLevelTasks. This provided the same functionality and didnt have any ripple effects/
2. Code Smell Between Classes: IEventNotificationListener line 20, DefaultEventNotifier line 39, EventsScheduler line 62, 107, 138. This was an example SPECULATIVE GENERALITY. This looks like something they wanted to implement in the future but the method was never actually implemented. There was just a comment in the body. So I had to remove it which caused ripples in other classes. I LEFT THE “DELETED” CODE IN AS COMMENTS SO YOU COULD SEE WHAT I WAS “DELETING”.



1. Lines of Code dropped from 2041 to 2030. This is because I deleted a method from an interface which caused a ripple effect. I was say less lines of code is an improvement.