# Task 1

#### Size

- 1. 2187
- 2. EventsManager.java | 329
- 3. Method 1, it counted each statement, but it also counted the ending curly brace that are on their own line, unlike Dr. Gary.

### Cohesion

1. The definition of LCOM2 is the percentage of methods that do not access a specific attribute averaged over all attributes in the class. The range is [0,2], anything >=1 indicates a class with a problem in regards to lack of cohesion. It is calculated by

$$LCOM2 = 1 - \frac{\sum (m*A)}{m*a}$$

Where m – number of methods in the class, a – number of variables in the class, mA – number of methods that access a variable, sum(mA) sum of mA over attributes of a class.

2.

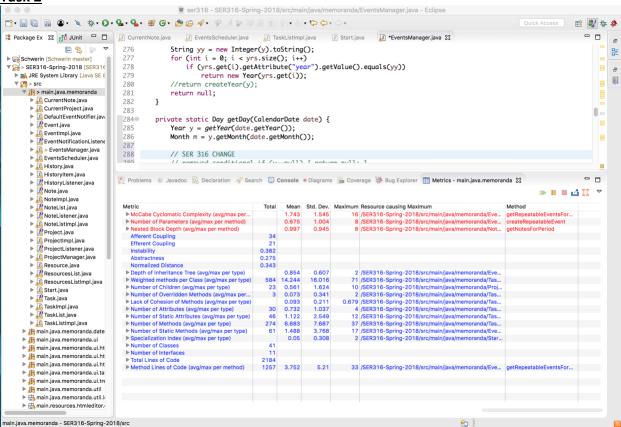
# Complexity

- 1. 1.746
- 2. Start.java | 3.5
- 3. In EventsManager.java on line 288 I removed if (y == null) { return null; } because it isn't totally necessary, if y is null then m should be null so the conditional is not necessary and it reduced the complexity from 3.353 to 3.294 on the class.

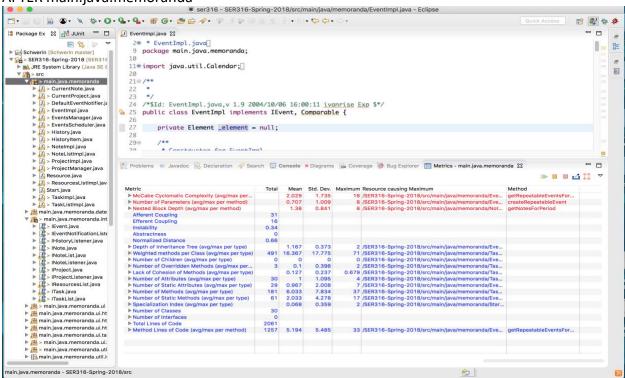
# Package-level Coupling

- Afferent coupling is a metric on the number of class in other packages that depend upon
  the classes within the current package (it's an indicator of a package's responsibility).
  Efferent Coupling is the number of classes in other packages that the classes in a
  package depend on (an indicator of the packages dependence on external packages).
  They are opposites. Afferent coupling is the number of outside packages that depend on
  this certain package, while efferent is the number of outside packages that are
  depended upon for this certain package.
- 2. Main.java.memoranda.util has the worst Afferent Coupling at 57.
- 3. And main.java.memoranda.ui is the package with the highest efferent coupling at 49

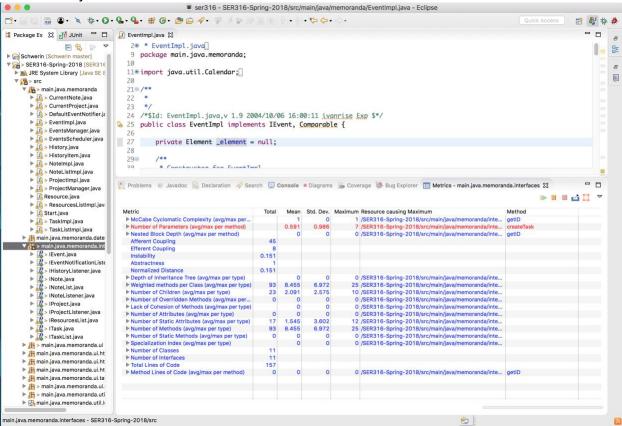
## Task 2



# AFTER main.java.memoranda





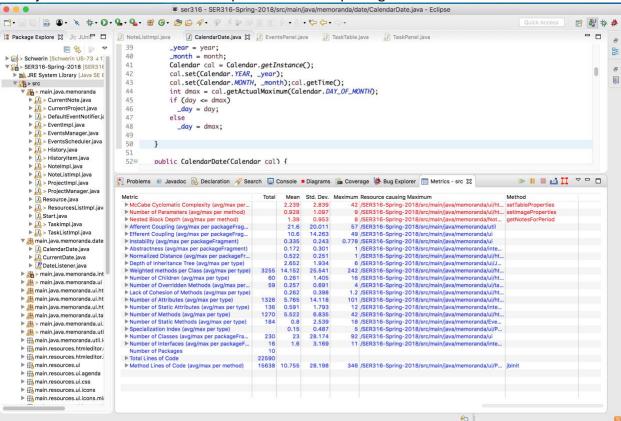


8. Afferent Coupling changed for the better in regards to main.java.memoranda. Because now the Interfaces that are depended upon by other classes is the main.java.memoranda.interfaces responsibility.

### Task 3

- 1. For the Code smell within a class I fixed The EventsManager.java #getRepeatableEventsForDate() method, it was big and nasty. I removed the responsibility in that method of checking if a date belongs into #checkIfEventBelongs() to separate responsibilities and make it easier to read the getRepeatableEventsForDate() method. Other than that, it is naturally complicated to check on dates, so the method is still a bit long.
- 2. I noticed some Feature Envy in TaskPanel.java a long list of chained methods was being called on a taskTable instance in TaskPanel, so I moved it into a method #getParentTaskId in TaskTable.java. The comments explain it in TaskPanel.java:613-625 & TaskTable.java:239-244
- 3. This is of the src package, because the code smells between classes that I refactored were in the main.java.memoranda.ui package. I was having a hard time finding smells in

main.java.memoranda. the After export is of the src package as well.



4. After my refactoring the TaskPanel's efferent coupling measure should have gone down because I removed it's dependency on the TaskTable. For TaskTable it's Afferent coupling should have gone down due to the exact opposite.