Branden Turley

SER 316 Sping18 B

Assignment 7

4/27/18

Task 1

Size

1. Total Lines of Code = 2187 LOC

2. EventsManager = 329 LOC

3. Method 1, it counted each line that had something on it, except for line 21 which was a comment.

Cohesion

1. The LCOM2 stands for Lack of Cohesion Among Method of Class, which measures the extent of intersections of individual method parameter types lists with the parameter type list of all methods in the class.

LCOM2 is in the range [0,2]. LCOM2 >= 1 indicates a very problematic class.

2. TaskListImpl.java because it has the highest avg Lack of Cohesion of Methods. Not sure why, it could be because of dead variables and a lack of encapsulation.

Complexity

1. Mean = 1.746, Std. Dev. 1.547, Max 16

2. EventsManager.java with a mean of 2.5

3. Removed some of the if statements in NoteListImpl, getNotesForPeriod, it reduced complexity by 1

Package-level Coupling

1. **Afferent Couplings (Ca):** The number of classes in other packages that depend upon classes within the package is an indicator of the package's responsibility. Afferent = incoming.

Efferent Couplings (Ce): The number of classes in other packages that the classes in the package depend upon is an indicator of the package's dependence on externalities. Efferent = outgoing.

2. main.java.memoranda.util with 57 Afferent Coupling

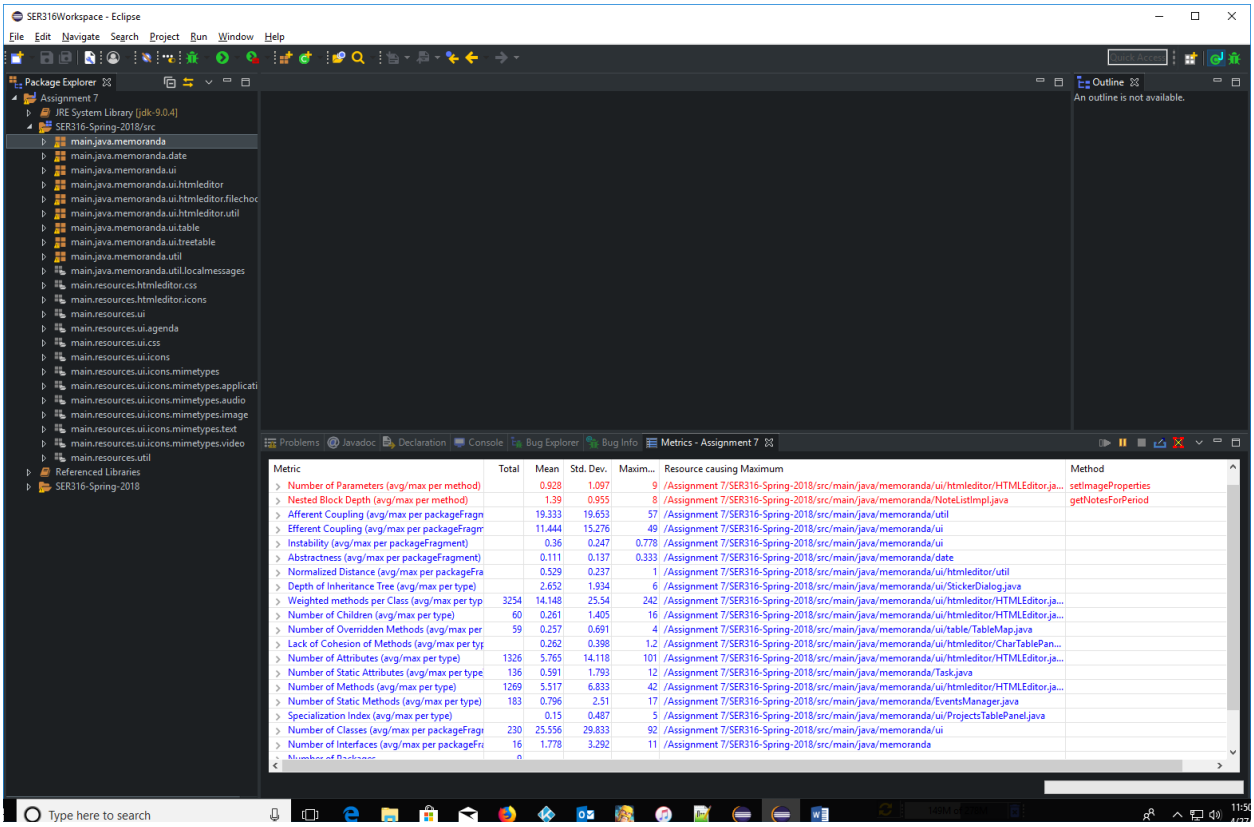
3. main.java.memoranda.ui with 49 Efferent Coupling

Worst Quality

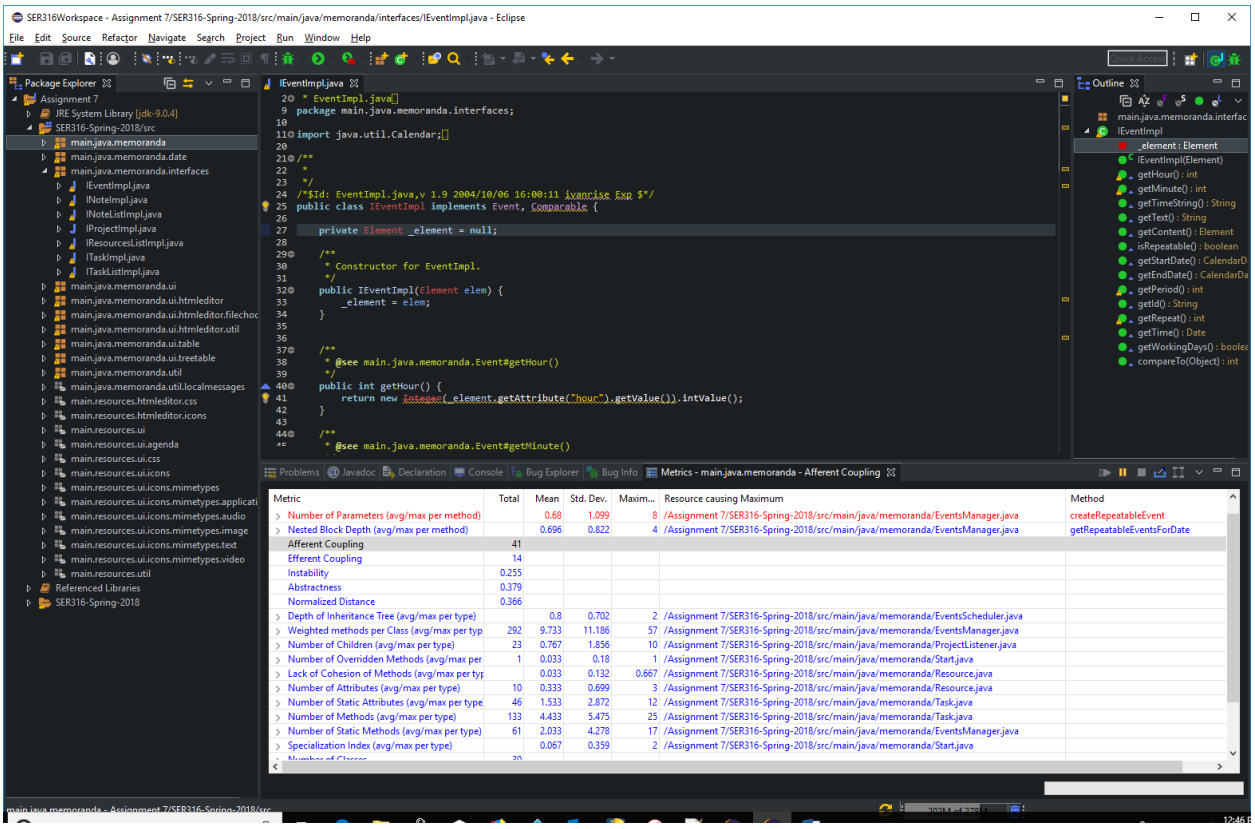
I think based on the metrics that the worst class is the TaskListImpl.java because it has the highest lack of cohesion of methods and has the highest number of methods. I would probably break this class up to increase its simplicity and to create more cohesive classes.

Task 2

1.



After



8. The Weighted methods per Class changed as well as the Nested Block Depth (which is no longer red). These changed because the number of classes increased which lowered the average number of methods. This is an improvement because it makes the program more compartmental and organized.

Task 3

1. CurrentNote.java line 21 – 24, unused / dead code. Fixed by simply commenting it out for reference.

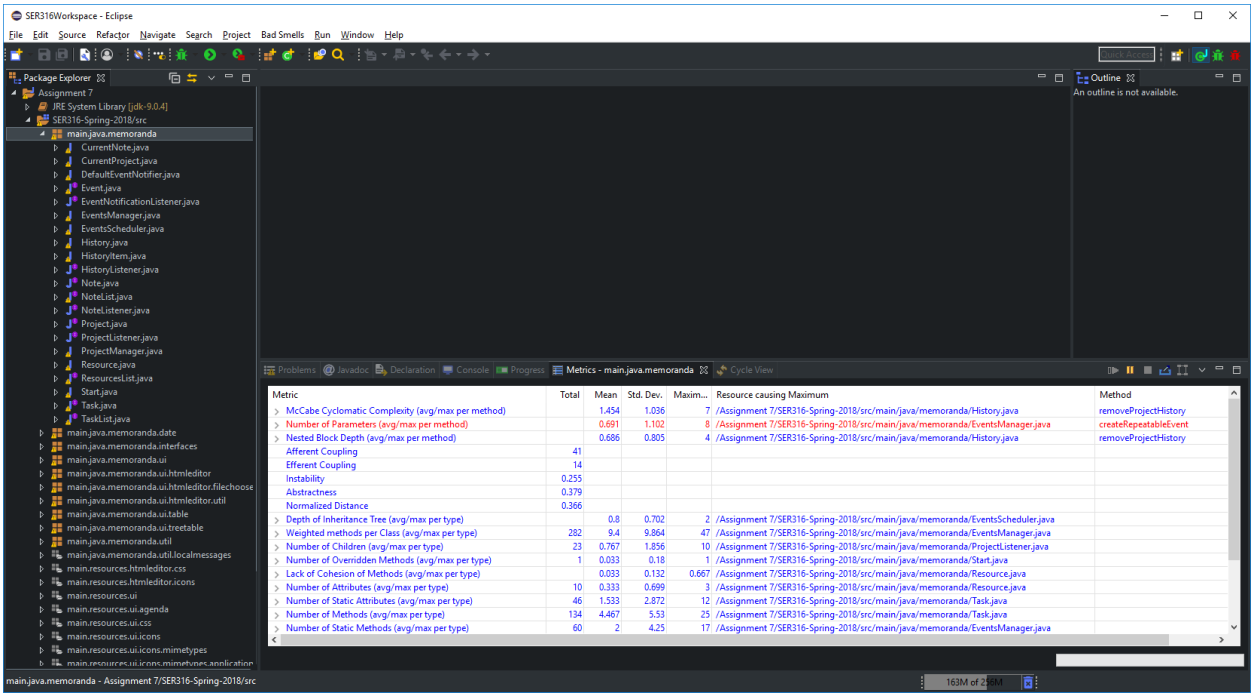
2. Large Class

Replace conditional structure with polymorphic method invocation in EventsManager.java.

Event.java add reference to getRepeatableEventsForDate…

IEventImpl.java, added method to check for repeatable events for date.

3. AFTER



4. After refactoring the Cyclomatic Complexity improved. It changed for the better because by refactoring the class it made it more simple and straightforward by breaking up the complexity.