

Harold Bowers (hrbowers)

SER316 – Mehlhase

Assignment 7

## Task 1

### SIZE:

- 1.) Total LOC is 2187
- 2.) The largest single code file is EventsManager.java with a total LOC of 329
- 3.) Used noteChanged, a method that iterates through note listeners and saves current note

### COHESION:

- 1.) Definitions of LCOM2 :

m – number of procedures (methods) in a class

a – number of variables (attributes) in a class

mA – number of methods that access a variable (attribute)

sum(mA) – sum of mA over attributes of a class

LCOM2 can be calculated as:

$$LCOM2 = 1 - \text{sum}(mA)/(m*a)$$

Where LCOM2 equals the percentage of methods that do not access a specific attribute

averaged over all attributes in the class. If the number of attributes or methods is zero,

LCOM2 is undefined and displayed as zero.

- 2.) The class with the highest cohesion is a 10-way tie of classes with a score of zero. One of them is ProjectImpl.java, and I believe this is because it has methods that mostly operate on the same attributes of the class.

### COMPLEXITY:

- 1.) The mean cyclomatic complexity for the main package is 1.746.
- 2.) The class with the highest McCabe score is Start.java with a score of 3.5.
- 3.) In EventsManager.java I commented out the first if statement located in public static Collection getEventsForDate(CalendarDate date), specifically line 114. I decided to make this change because this specific statement did not seem necessary to the program's functionality, and since complexity measures the possible paths of a class, I knew removing an if statement would reduce the number of paths. I was able to reduce the score (for EventsManager.java specifically) from 2.5 to 2.469. This lowered the overall package score from 1.746 to 1.743.

## PACKAGE-LEVEL COUPLING:

- 1.) Coupling is the degree of interdependence between software modules. **Afferent coupling** is the number of classes in other packages that depend upon classes within the package is an indicator of the package's responsibility. **Efferent coupling** is 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. Efferent couplings signal outward. The key difference between the two is that afferent measures responsibility to other classes while efferent measures dependence on other classes.
- 2.) main.java.memoranda.util has the highest afferent coupling with a score of 57.
- 3.) main.java.memoranda.ui has the highest efferent coupling with a score of 49.

## WORST QUALITY:

The class with the worst quality in my opinion is TaskListImpl.java. To arrive at this conclusion, I searched to find a class that ranked high in both cyclomatic complexity and lack of cohesion. TaskListImpl.java, which had the highest lack of cohesion score (0.679) as well as being above a score of 2 in complexity (2.273). While other methods would score worse in complexity, TaskListImpl.java was the only one that also scored poorly in cohesion.

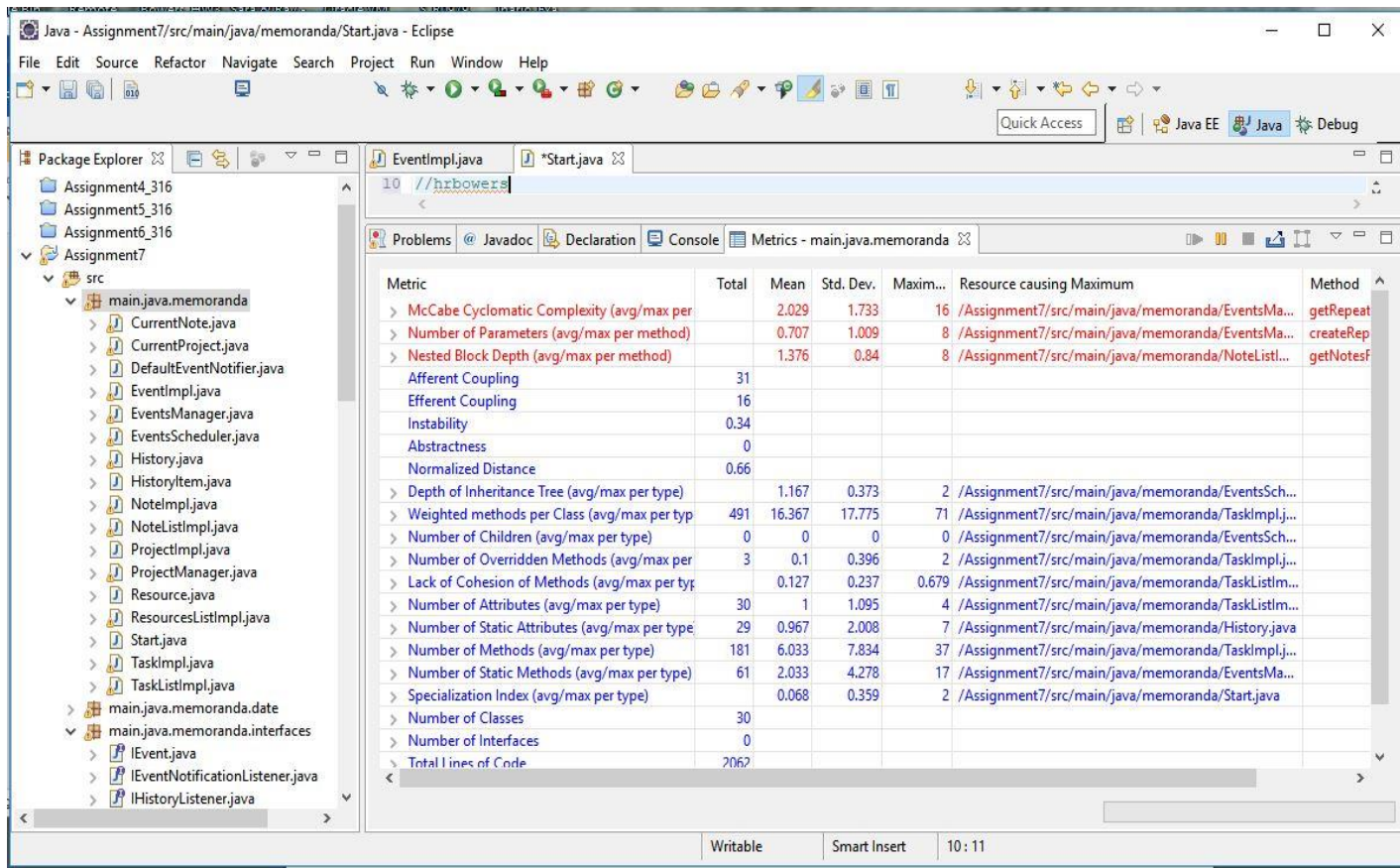
## Task 2

### 1.) Rerun of metrics plugin

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, with 'main.java.memoranda' expanded. The Metrics plugin window is open, showing a table of metrics for the 'main.java.memoranda' package. The table includes columns for Metric, Total, Mean, Std. Dev., Maxim..., and Resource causing Maximum. The metrics are sorted by Total value, with 'Total Lines of Code' at the top (2185) and 'Lack of Cohesion of Methods (avg/max per type)' at the bottom (0.679).

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	
> McCabe Cyclomatic Complexity (avg/max per	1.743	1.543	16	/Assignment7/src/main/java/memoranda/EventsMa...	getf...	
> Number of Parameters (avg/max per method)	0.675	1.004	8	/Assignment7/src/main/java/memoranda/EventsMa...	crea...	
> Nested Block Depth (avg/max per method)	0.994	0.943	8	/Assignment7/src/main/java/memoranda/NoteList...	getf...	
Afferent Coupling	34					
Efferent Coupling	21					
Instability	0.382					
Abstractness	0.275					
Normalized Distance	0.343					
> Depth of Inheritance Tree (avg/max per type)	0.854	0.607	2	/Assignment7/src/main/java/memoranda/EventsSch...		
> Weighted methods per Class (avg/max per typ	584	14.244	16.016	71	/Assignment7/src/main/java/memoranda/TaskImpl...	
> Number of Children (avg/max per type)	23	0.561	1.624	10	/Assignment7/src/main/java/memoranda/ProjectList...	
> Number of Overridden Methods (avg/max per	3	0.073	0.341	2	/Assignment7/src/main/java/memoranda/TaskImpl...	
> Lack of Cohesion of Methods (avg/max per typ	0.093	0.211	0.679	/Assignment7/src/main/java/memoranda/TaskListm...		
> Number of Attributes (avg/max per type)	30	0.732	1.037	4	/Assignment7/src/main/java/memoranda/TaskListm...	
> Number of Static Attributes (avg/max per type	46	1.122	2.549	12	/Assignment7/src/main/java/memoranda/Task.java	
> Number of Methods (avg/max per type)	274	6.683	7.687	37	/Assignment7/src/main/java/memoranda/TaskImpl...	
> Number of Static Methods (avg/max per type)	61	1.488	3.768	17	/Assignment7/src/main/java/memoranda/EventsMa...	
> Specialization Index (avg/max per type)	0.05	0.308	2	/Assignment7/src/main/java/memoranda/Start.java		
> Number of Classes	41					
> Number of Interfaces	11					
> Total Lines of Code	2185					

## 7.) Rerun of metrics plugin after moving interfaces to new package and refactoring



Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
> McCabe Cyclomatic Complexity (avg/max per		2.029	1.733	16	/Assignment7/src/main/java/memoranda/EventsMa...	getRepeat
> Number of Parameters (avg/max per method)		0.707	1.009	8	/Assignment7/src/main/java/memoranda/EventsMa...	createRep
> Nested Block Depth (avg/max per method)		1.376	0.84	8	/Assignment7/src/main/java/memoranda/NotelIsl...	getNotesF
Afferent Coupling	31					
Efferent Coupling	16					
Instability	0.34					
Abstractness	0					
Normalized Distance	0.66					
> Depth of Inheritance Tree (avg/max per type)		1.167	0.373	2	/Assignment7/src/main/java/memoranda/EventsSch...	
> Weighted methods per Class (avg/max per typ	491	16.367	17.775	71	/Assignment7/src/main/java/memoranda/TaskImplj...	
> Number of Children (avg/max per type)	0	0	0	0	/Assignment7/src/main/java/memoranda/EventsSch...	
> Number of Overridden Methods (avg/max per	3	0.1	0.396	2	/Assignment7/src/main/java/memoranda/TaskImplj...	
> Lack of Cohesion of Methods (avg/max per typ		0.127	0.237	0.679	/Assignment7/src/main/java/memoranda/TaskListm...	
> Number of Attributes (avg/max per type)	30	1	1.095	4	/Assignment7/src/main/java/memoranda/TaskListm...	
> Number of Static Attributes (avg/max per type	29	0.967	2.008	7	/Assignment7/src/main/java/memoranda/History.java	
> Number of Methods (avg/max per type)	181	6.033	7.834	37	/Assignment7/src/main/java/memoranda/TaskImplj...	
> Number of Static Methods (avg/max per type)	61	2.033	4.278	17	/Assignment7/src/main/java/memoranda/EventsMa...	
> Specialization Index (avg/max per type)		0.068	0.359	2	/Assignment7/src/main/java/memoranda/Start.java	
> Number of Classes	30					
> Number of Interfaces	0					
> Total Lines of Code	2062					

8.) Abstractness dropped from 0.275 to 0 and Instability dropped from 0.382 to 0.34. Efferent coupling dropped from 21 to 16. Lines of code dropped from 2185 to 2062. These drops occurred because of the movement of interfaces to the new package, while stats like complexity and cohesion actually got worse because the interfaces helped to lower the score.

## Task 3

- 1.) For smell within a class I found duplicate code in NoteListImpl.java, specifically the nested for loops used in public Collection getAllNotes() (line 77) and public Collection getMarkedNotes() (line 99) used to iterate through dates. To fix this, I created a private method inside the same class called private Vector v() (line 54) which handles the for loop iteration and is then used by both of the methods.
- 2.) Smells between classes was a bit more difficult as I wasn't sure if changes would enhance anything, but I decided that class envy between EventsScheduler.java and EventsManager.java was a valid smell. To alter this, I moved all code from EventsScheduler into EventsManager and deleted the EventsScheduler class. References to EventsScheduler were changed to reference

EventManager in AgendaGenerator.java of package main.java.memoranda.util. In package main.java.memoranda.ui changes occurred in AgendaPanel.java, App.java, DailyItemsPanel.java, and EventsPanel.java. The end result is removal of the EventsScheduler class.

### 3.) After (code smell fixes)

The screenshot shows the Eclipse IDE with the 'Metrics' view open, displaying the McCabe Cyclomatic Complexity (avg/max per method) for the project 'main.java.memoranda'. The metrics table is as follows:

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
> McCabe Cyclomatic Complexity (avg/max per	2.012	1.706	16	/Assignment7/src/main/java/memoranda/EventsMa...	getRepeat	
> Number of Parameters (avg/max per method)	0.704	1.008	8	/Assignment7/src/main/java/memoranda/EventsMa...	createRep	
> Nested Block Depth (avg/max per method)	1.366	0.792	8	/Assignment7/src/main/java/memoranda/NoteListl...	getNotesF	
Afferent Coupling	31					
Efferent Coupling	15					
Instability	0.326					
Abstractness	0					
Normalized Distance	0.674					
> Depth of Inheritance Tree (avg/max per type)	1.172	0.378	2	/Assignment7/src/main/java/memoranda/History.java		
> Weighted methods per Class (avg/max per typ	489	16.862	19.562	74	/Assignment7/src/main/java/memoranda/EventsMa...	
> Number of Children (avg/max per type)	0	0	0	0	/Assignment7/src/main/java/memoranda/ProjectIm...	
> Number of Overridden Methods (avg/max per	3	0.103	0.402	2	/Assignment7/src/main/java/memoranda/TaskImpl.j...	
> Lack of Cohesion of Methods (avg/max per typ	0.134	0.246	0.679	/Assignment7/src/main/java/memoranda/TaskListm...		
> Number of Attributes (avg/max per type)	30	1.034	1.098	4	/Assignment7/src/main/java/memoranda/TaskListm...	
> Number of Static Attributes (avg/max per typ	29	1	2.364	10	/Assignment7/src/main/java/memoranda/EventsMa...	
> Number of Methods (avg/max per type)	182	6.276	7.917	37	/Assignment7/src/main/java/memoranda/TaskImpl.j...	
> Number of Static Methods (avg/max per type)	61	2.103	5.416	26	/Assignment7/src/main/java/memoranda/EventsMa...	
> Specialization Index (avg/max per type)	0.071	0.365	2	/Assignment7/src/main/java/memoranda/Start.java		
> Number of Classes	29					
> Number of Interfaces	0					

4.) Numerous metrics show a change for the better after fixing the code smells. Cyclomatic complexity has dropped from 2.029 to 2.012 because of the duplicate code fix, moving all the if and for loop statements into a single method shared by others. Efferent coupling dropped from 15 to 16 as a class was removed due to class envy, reducing dependency. Instability dropped from 0.34 to 0.326, and obviously the number of class has dropped from 30 to 29. While all these were positive changes, I did notice that LCOM actually rose from 0.127 to 0.134, which I believe is due to the EventsManager class now taking on all the variables of the EventsScheduler class. All in all, I believe the smell fixes were beneficial.

