Timothy Reser (treser)

Assignment 7- Refactoring/Metrics

Task 1:

Size:

1. 22539 total LOC
2. HTMLEditor.java has the most with 2144 LOC
3. CurrentNote.java has 28 LOC
   It counts all lines that are not blank or comments, which includes lines that contain only brackets as well ({}).

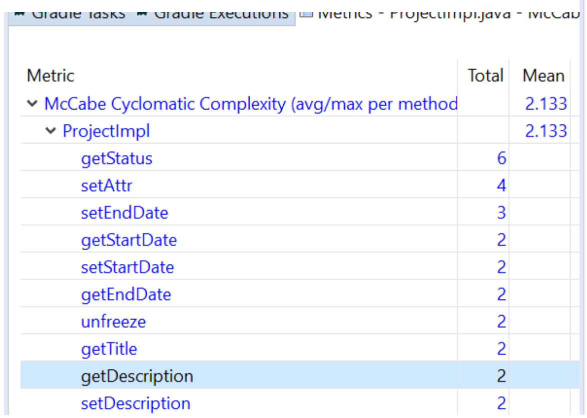
Cohesion:

1. Henderson-Sellers method of LCOM is a measure of how well designed a class is in terms of encapsulation, subdivision, and simplicity. LCOM2 is the percentage of methods that do not access a specific attribute averaged over all attributes in the class. If the number is 0, LCOM is undefined and determined to be 0.
   LCOM2 is defined by the equation $1 - \frac{sum(mA)}{m*a}$ where:
   $$m = number\ of\ methods\ in\ the\ class$$
   $$a = number\ of\ variables\ in\ a\ class$$
   $$mA = number\ of\ methods\ that\ access\ a\ variable$$
   $$sum(mA) = the\ sum\ of\ mA\ over\ variables\ in\ a\ class$$
2. HistoryItem.java has the highest cohesion at LCOM2 = 0.333 (unless you count the several classes that have LCOM2 = 0).
   This is because there are only 2 variables/attributes in the class and all the methods access at least one of them.

Complexity:

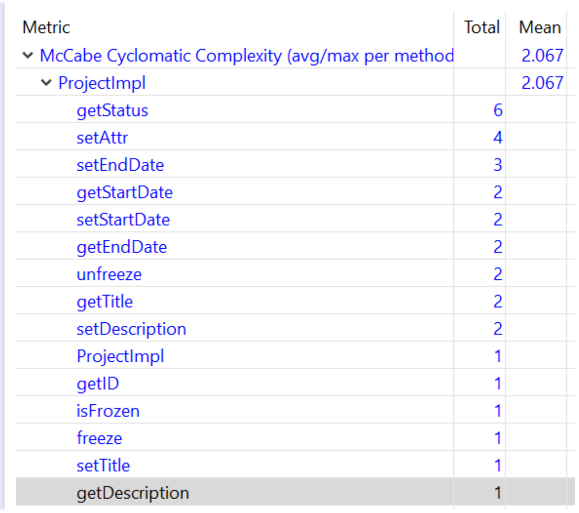
1. The cyclomatic complexity of the main package is 1.749 on average (mean).
2. The worst McCabe Cyclomatic Complexity belongs to Start.java with an average of 3.5.
3. Initially the complexity of the method getDescription in ProjectImpl.java was 2. After making some modifications to remove unnecessary conditionals the complexity has been reduced to 1, lowering the average for the file by 0.07.

Timothy Reser (treser)



Metric	Total	Mean
McCabe Cyclomatic Complexity (avg/max per method		2.133
ProjectImpl		2.133
getStatus	6	
setAttr	4	
setEndDate	3	
getStartDate	2	
setStartDate	2	
getEndDate	2	
unfreeze	2	
getTitle	2	
getDescription	2	
setDescription	2	

Before



Metric	Total	Mean
McCabe Cyclomatic Complexity (avg/max per method		2.067
ProjectImpl		2.067
getStatus	6	
setAttr	4	
setEndDate	3	
getStartDate	2	
setStartDate	2	
getEndDate	2	
unfreeze	2	
getTitle	2	
setDescription	2	
ProjectImpl	1	
getID	1	
isFrozen	1	
freeze	1	
setTitle	1	
getDescription	1	

After

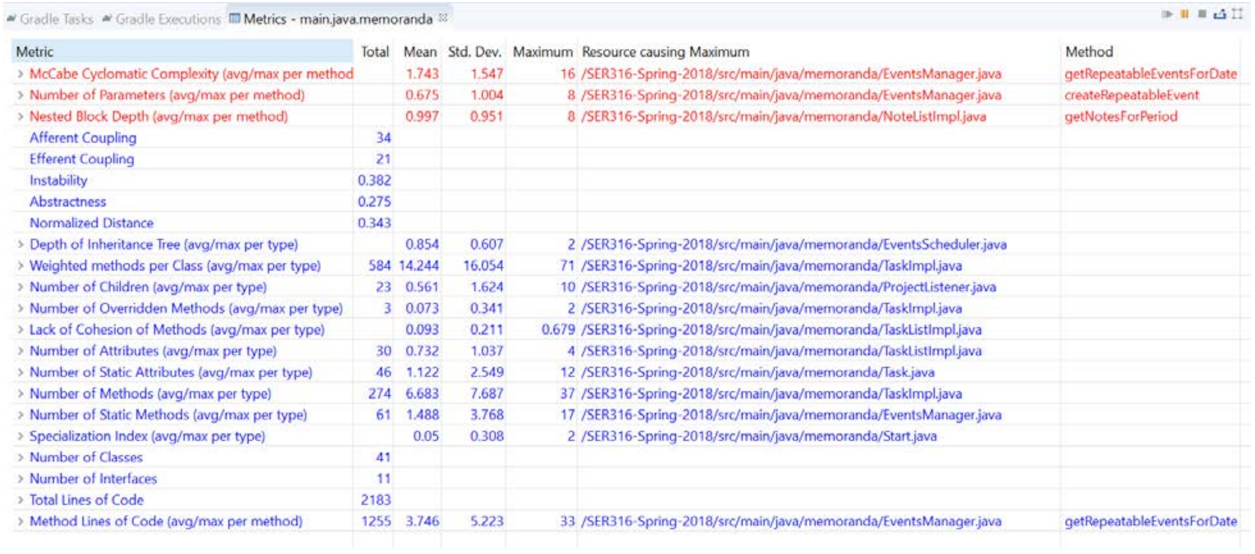
Package-Level Coupling:

1. Afferent means that other classes depend on the class that is being referenced. Efferent means that the class being reference depends on other classes.
2. The package with the worst Afferent Coupling is main.java.memoranda.util with a total of 57.
3. The package with the worst Efferent Coupling is main.java.memoranda.ui with a total of 49.

Worst Quality:

NoteListImpl is the worst class. It has an average of 3.692 for cyclomatic complexity and 1 of the 13 methods (getNotesForPeriod) has a CC of 15. Additionally, NoteListImpl has an above average LOCM2 value of 0.50.

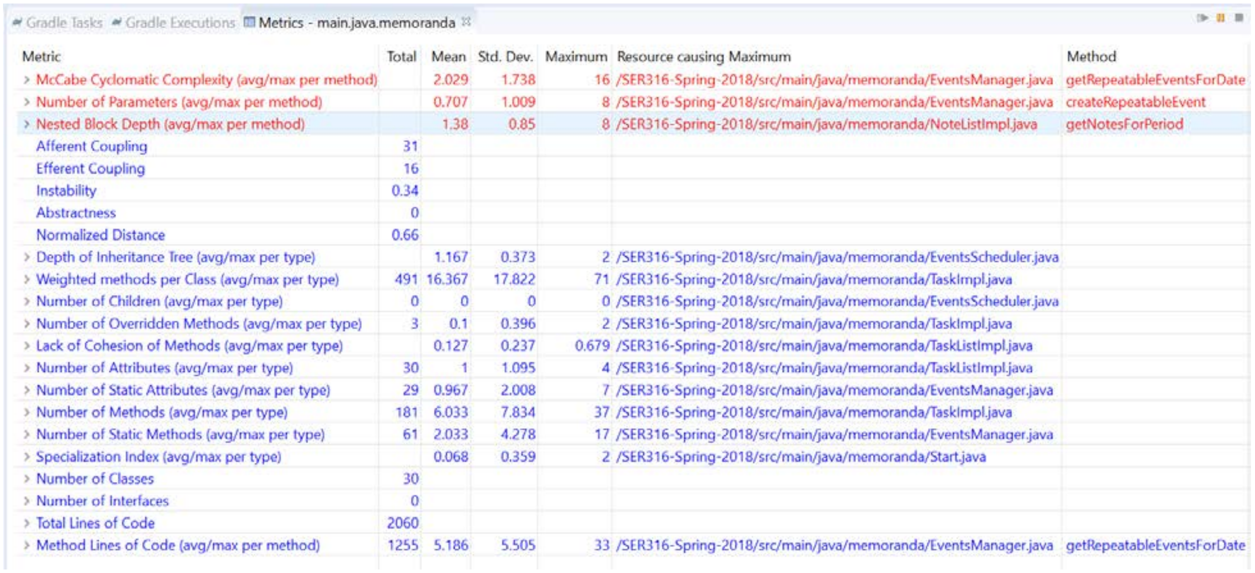
Task 2:



Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per method		1.743	1.547	16	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate
Number of Parameters (avg/max per method)		0.675	1.004	8	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	createRepeatableEvent
Nested Block Depth (avg/max per method)		0.997	0.951	8	/SER316-Spring-2018/src/main/java/memoranda/NoteListImpl.java	getNotesForPeriod
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	/SER316-Spring-2018/src/main/java/memoranda/EventsScheduler.java	
Weighted methods per Class (avg/max per type)	584	14.244	16.054	71	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Children (avg/max per type)	23	0.561	1.624	10	/SER316-Spring-2018/src/main/java/memoranda/ProjectListener.java	
Number of Overridden Methods (avg/max per type)	3	0.073	0.341	2	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Lack of Cohesion of Methods (avg/max per type)		0.093	0.211	0.679	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Attributes (avg/max per type)	30	0.732	1.037	4	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Static Attributes (avg/max per type)	46	1.122	2.549	12	/SER316-Spring-2018/src/main/java/memoranda/Task.java	
Number of Methods (avg/max per type)	274	6.683	7.687	37	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Static Methods (avg/max per type)	61	1.488	3.768	17	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	
Specialization Index (avg/max per type)		0.05	0.308	2	/SER316-Spring-2018/src/main/java/memoranda/Start.java	
Number of Classes	41					
Number of Interfaces	11					
Total Lines of Code	2183					
Method Lines of Code (avg/max per method)	1255	3.746	5.223	33	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate

Before

Timothy Reser (treser)



Gradle Tasks | Gradle Executions | Metrics - main.java.memoranda

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per method)		2.029	1.738	16	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate
Number of Parameters (avg/max per method)		0.707	1.009	8	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	createRepeatableEvent
Nested Block Depth (avg/max per method)		1.38	0.85	8	/SER316-Spring-2018/src/main/java/memoranda/NoteListImpl.java	getNotesForPeriod
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	/SER316-Spring-2018/src/main/java/memoranda/EventsScheduler.java	
Weighted methods per Class (avg/max per type)	491	16.367	17.822	71	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Children (avg/max per type)	0	0	0	0	/SER316-Spring-2018/src/main/java/memoranda/EventsScheduler.java	
Number of Overridden Methods (avg/max per type)	3	0.1	0.396	2	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Lack of Cohesion of Methods (avg/max per type)		0.127	0.237	0.679	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Attributes (avg/max per type)	30	1	1.095	4	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Static Attributes (avg/max per type)	29	0.967	2.008	7	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	
Number of Methods (avg/max per type)	181	6.033	7.834	37	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Static Methods (avg/max per type)	61	2.033	4.278	17	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	
Specialization Index (avg/max per type)		0.068	0.359	2	/SER316-Spring-2018/src/main/java/memoranda/Start.java	
Number of Classes	30					
Number of Interfaces	0					
Total Lines of Code	2060					
Method Lines of Code (avg/max per method)	1255	5.186	5.505	33	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate

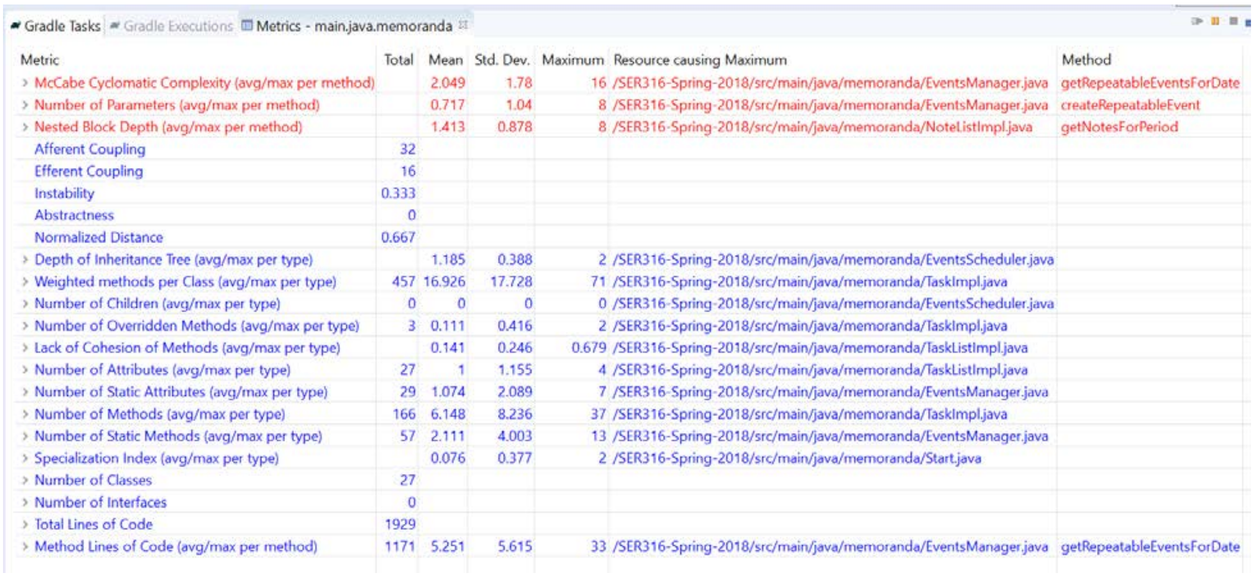
After

8. The Afferent and Efferent Coupling decreased, which is an improvement. This change is due to there no longer being as many dependencies within the package. This is a little misleading, however, as those values were just moved to another package. Other metrics have changed for the worse, such as the increase in average cyclomatic complexity and LCOM2. This change is not necessarily representative either since interfaces can inflate the LOC without changing complexity which makes the averages lower than they probably should be.

Task 3:

1. EventsManager.java had the code smell of divergent change. It had several methods that had nothing to do with event objects but rather belonged to a date/calendar class. It could easily be fixed by moving/refactoring the methods unrelated to events to the appropriate class. This is the approach I took and moved all date type objects were moved to the CalendarDate class in the main.java.memoranda.date package.

Timothy Reser (treser)


Gradle Tasks | Gradle Executions | Metrics - main.java.memoranda

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per method)		2.049	1.78	16	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate
Number of Parameters (avg/max per method)		0.717	1.04	8	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	createRepeatableEvent
Nested Block Depth (avg/max per method)		1.413	0.878	8	/SER316-Spring-2018/src/main/java/memoranda/NoteListImpl.java	getNotesForPeriod
Afferent Coupling	32					
Efferent Coupling	16					
Instability	0.333					
Abstractness	0					
Normalized Distance	0.667					
Depth of Inheritance Tree (avg/max per type)		1.185	0.388	2	/SER316-Spring-2018/src/main/java/memoranda/EventsScheduler.java	
Weighted methods per Class (avg/max per type)	457	16.926	17.728	71	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Children (avg/max per type)	0	0	0	0	/SER316-Spring-2018/src/main/java/memoranda/EventsScheduler.java	
Number of Overridden Methods (avg/max per type)	3	0.111	0.416	2	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Lack of Cohesion of Methods (avg/max per type)		0.141	0.246	0.679	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Attributes (avg/max per type)	27	1	1.155	4	/SER316-Spring-2018/src/main/java/memoranda/TaskListImpl.java	
Number of Static Attributes (avg/max per type)	29	1.074	2.089	7	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	
Number of Methods (avg/max per type)	166	6.148	8.236	37	/SER316-Spring-2018/src/main/java/memoranda/TaskImpl.java	
Number of Static Methods (avg/max per type)	57	2.111	4.003	13	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	
Specialization Index (avg/max per type)		0.076	0.377	2	/SER316-Spring-2018/src/main/java/memoranda/Start.java	
Number of Classes	27					
Number of Interfaces	0					
Total Lines of Code	1929					
Method Lines of Code (avg/max per method)	1171	5.251	5.615	33	/SER316-Spring-2018/src/main/java/memoranda/EventsManager.java	getRepeatableEventsForDate

After

4. Both CC and LCOM2 averages increased, by 0.02 0.013 respectively, which a change for the worse. Much like earlier steps this isn't due to a reduction of code quality. This change is due to the fact that methods with above average CC and LCOM2 properties were moved out of the package leaving a higher percentage of below average methods.