

Program Description:

The program consists of four classes: CalendarMain, Appt, CalDay, and TimeTable. CalendarMain has only one public method call and consists of 140 lines of code that show the interaction between the other three classes in creating a calendar of dates. Appt carries 18 functions; one of them is private with 211 lines of code. Appt is used for correctly gathering the time and date to be stored in the calendar. CalDay has 15 functions, all of which are public, and is 190 lines of code long with the intent of linking submitted dates and presenting them together to the user. Lastly, TimeTable has four functions, one of which is private with 146 lines of code. This class's purpose is to show all valid days between two given dates.


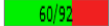


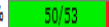





Utility of the code coverage tool:

The test cases achieved a high coverage rating with line coverage being in the 90's for each class and with branch coverage being 72% and 85% covered. Test cases were most easily implemented for simple functions like getters and setters with a notable increase in difficulty when attempting to check functions that carry branching statements and calls to other classes. In total, 15 test cases were made with seven for TestAppt, five for TestCalDay, and three for TestTimeTable. The testing was hardest in CalDay with failures in achieving coverage for branches in a private function. In addition to this, 'lterator' and 'addAppt' had branches that I failed to get coverage on with most testing efforts just going through one of their two branches. This problem was universal as well; handling branches within Appt 'toString' was also a source of several missed branches. As for Cobertura, it was helpful in clearly indicating what is and isn't covered, as well as how many tests pass through a single point in the code. It is a little redundant however, since there are several platforms to code on that already provide this service, but in a more centralized fashion. An example of this easier-to-use testing would be IntelliJ.

Unit Testing Efforts:

I stumbled several times through test coverage with branches in particular being a pain to deal with. I ultimately had to give up on CalDay achieving an 80% branch coverage after I found myself creating meaningless tests and warping existing ones to no avail. The rest was a lot more agreeable for me, meeting the requirements with only a few snags along the way. After the completion of testing, I can say that the code is pretty sturdy in what it needs to do - nearly every function this program provides is covered, albeit in varying degrees. That being said, my inability to hit a few of the branches may indicate an issue in how the code is read - though it is more than likely a fault on my end.

Coverage Report - edu.osu.cs362

Package /	# Classes	Line Coverage	Branch Coverage	Complexity
edu.osu.cs362	4	67%  151/224	65%  60/92	2.447
Classes in this Package /		Line Coverage	Branch Coverage	Complexity
Appt		95%  58/61	85%  24/28	1.833
CalDay		94%  50/53	72%  13/18	1.733
CalendarMain		0%  0/64	0%  0/18	10
TimeTable		93%  43/46	82%  23/28	6