



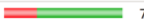
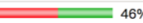










CS362

Assignment 2

Donghao Lin(lindo)

The Utility of code coverage tool

calendar

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
CalendarMain		0%		0%	4 4	76 76	2 2	1 1
DataHandler		71%		46%	35 57	63 260	0 16	0 1
XmlParserErrorHandler		3%		0%	5 6	13 14	4 5	0 1
Appt		94%		84%	8 58	4 99	1 36	0 1
CalendarUtil		78%		30%	6 9	6 13	1 4	0 1
CalDay		96%		82%	5 30	1 76	0 16	0 1
DateOutOfRangeException		0%		n/a	2 2	4 4	2 2	1 1
Total	770 of 2,244	65%	68 of 168	59%	65 166	167 542	10 81	2 7

I had 80% line coverage and branch coverage for Appt and CalDay but less than 80% for DataHandler. Because I was not able to find out more cases for the private class in the DataHandler, and once I put some elements into the function, it returned error.

The test cases that I create almost cover all source code of the Calendar application, But still some of them are not covered, such as constructor, isOn(), and hasTimeset() in Appt class, if(minute<10) in getFullInformationApp() in CalDay class and lots of cases in DataHandler class.

The advantage of using code coverage tool like JaCoCo is that it help us to find of what cases we did not cover directly and spend less time to fix them.

Unit Testing Efforts

Unit testing saves our time to find out how the functions work for the whole application, and we can make big changes to the code in a short time. Also unit testing help us understand the design of code and how the code are working.

I did not find an bug due to I did not cover all cases and I was not fully understand how every functions works for and what should the outputs be.