Logan Kling
CS362 Aburas
May 1, 2017
Assignment 2 Report

Assignment 2 Report

Program Description:

This program has 3 classes, Appt.java, CalDay.java, & TimeTable.java.

Appt.java is just a class that stores data for an appointment, day, month, year, title, & description. It also allows you to stringify the data involved with any appointment.

Public Methods: 18, Private methods: 1, SLOC: 113

CalDay.java is just a class that stores all the appointments in any given day. It's a linked list of appointments, with merely the day, month, year, & a Boolean denoting whether or not each calendar day is valid. With this you can find an appointment by day, month, or year, & add an appointment to any existing day. It also allows you to create new days & stringify the data involved with any given day.

Public Methods: 11, Private methods: 4, SLOC: 90

TimeTable.java is what really puts the other two classes together. This is a linked list of calendar days that can be treated sort of like a calendar—though it can contain invalid days that won't be on any calendar. With this you can get a range of dates with every appointment in between, or you can delete any appointment.

Public Methods: 3, Private methods: 1, SLOC: 71

Utility:

Cobertura worked very well for showing the level of test coverage, it was just *incredibly* time consuming to write tests with the goal of getting a hundred percent test coverage. I do think it could be more clear what the difference between line coverage & branch coverage is—I'm still struggling to grasp what exactly branch coverage is.

I ended up creating twenty-two test cases, not including the one already in ApptTest.java & I still couldn't get a hundred percent test coverage. I found it downright impossible to get 100% test coverage on private functions. Both if statements within setAppts() in CalDay.java are red, despite the fact that I know I've tested the first clause, & despite the fact that

```
appts != null && appts.size() == 0
```

is downright impossible. If something is null, its size must be greater than zero.

I had the greatest issues with TimeTable.java, however. I couldn't even achieve 100% line coverage on that program. That class honestly made me wonder if there were faults within Cobertura that I was running into. I had two assertions that threw errors for reasons beyond my comprehension:

1. I said the size of a TimeTable with two appointments in it should be two. Cobertura thought it should be one.
2. I tried checking the same appointment against itself in two very different ways. They should be equivalent, right? Cobertura said they were different, even though its output showed they were very clearly the same.

Cobertura can supposedly find easy to miss bugs, but I think its cost in time greatly outweighs its gain in test coverage. I struggled to write test cases for this program which is minute in terms of programs developed by even small companies. However, if one had unlimited time & money, they could supposedly create a large program with a hundred percent test coverage.

Unit Testing:

I found it surprisingly easy to develop ApptTest.java, & while CalDayTest.java took a while to create, my unresolved issue with `private void setAppts()` was the only issue I had with that & I explained that above. I have detailed most of my issues unit testing above, but there was one test based on testing an exception that took me quite a while to fix. My `exception.expect(IllegalArgumentException.class);` catch took quite a while to create do to my own stubbornness. You do not use any assert method in the case of an exception, & the line of code above must come before such an exception is triggered. Not only did it take me almost an hour to figure that out, but it took me almost another hour to realize that I needed this before my test in the class for that to work at all:

```
@Rule
 public final ExpectedException exception = ExpectedException.none();
```

As for the reliability of the Calendar application in itself, I can only say that it appears reliable despite being very cumbersome to use. The reason I don’t know is simply because I don’t know if the issue I had with checking the same appointment against itself in two very different ways was due to Cobertura, or due to the Calendar application.