Nicholas Skinner

CS362

5/1/2017

Assignment 2

**Program Description**

This program contains 4 total classes, those classes being Appt, CalDay, TimeTable, and CalendarMain.

Appt contains 18 methods, and most of them are simple get/set methods that put the data in place. A list of these methods is: setStartHour, setStatMinitue, setStartDay, set StartMonth, setStartYear, setTitle, setDescription, and all of the get equivalents of those methods. Aside from all of those we have the following:

Appt - creates new appointment with the received parameters

isValid - checks that date time variables are all proper.

toString - strings all of the data pertaining to the specific appointment in a nice printed statement.

CalDay contains 16 methods, and the majority of them appear to be very basic methods to get and set information in a column associated with the event, these simple methods include: setDay, setMonth, setYear, getDay, getMonth, and getYear. The methods aside from those are:

CalDay - which is a linked list that appends additional appointments, but does not validate them.

addAppt - adds appointments to the calendar day. Appointments are ordered by start time.

isValid - returns true

Iterator - iterates through to ensure appointments are organized by time.

getAppts - receives and returns appointments

getSizeAppts - returns the size of the linkedlist of appts

toString - returns representation of date as a string format.

Time Table is intended to retrieve a number of appointments between two dates, and it contains 3 methods. These methods include:

TimeTable - receives and validates input to find a list of appointments between a number of dates given to the method. It checks to see if the two inputs are actually different dates.

getApptOccurences - receives and validates input pertaining to an appointment, and the date range to scan for the frequency of the appointments.

deleteAppt - Deletes the appointment specified

CalendarMain contains one big long method, with many subroutines to interact with the linked list. The big idea behind this method is being able to view a day at a time with all of its associated appointments, the user is able to delete appointments, and create new ones.

**Utility coverage of the Code Tool**

* How many test cases were generated?

I managed to create 14 different test cases that covered (outside of the main method) the appt, gregorian, and timetable classes. The tests I created managed to cover 73% of the Appt class with a total of 5 tests, 92% of the calday class with 4 tests, and 88% of the TimeTable class with the remaining 5 tests.

* Point out some code your test cases do not cover.

My test cases do not cover a lot of the smaller more menial methods like setDesctiption, setMonth, etc. and checking some of the string printing, as I was encountering a number of logic based difficulties in doing these, and found it easier to implement other methods. A number of cases I had also not gotten to were most of the null scenario situations, as they mostly require a modified input that is nulled out.

.

* What is the advantage of using a code coverage tool? (how it helped/didnt help…)

Cobertura was an interesting tool to work with, it would create an easy to view graphical display of not only how covered your code was by test cases, but also what specific segments were not covered. I do not believe I would have written as many, nor as specific of test cases without the tool, and I believed it assisted in the code coverage of my tests. My test cases would likely not have covered as much of the code required for the assignment, and it also gave some valuable insight into how to design a program to be much easier to design tests for.

**Unit testing efforts**

* Describe unit testing efforts, and experience testing the calendar application.

Creating unit tests was a mild hurdle to learn by itself, but it was very challenging to create larger test cases to test larger segments of code, as there is a lot more information needed in the test method. It is also difficult in some situations to ensure that all the expected true or expected false cases were correct without running a manual test to discover it beforehand.

* Document the status and your view of the reliability of the calendar application.

The calendar application appears to be functional for its designed purpose for the most part, perhaps some additional input checking for classes like CalDay, or some methods to interact with the data in a slightly more meaningful way could be implemented. I also believe that the main method, and the timetable method could be implemented in smaller chunks to be much more approachable to creating test methods for.