James Barry
CS362-001-W2018

Assignment 1 - Calendar Bugs

Program Description

The Calendar application given for this assignment is a java-based calendar that can be run on a command line. It gives various tools like creating and updating appointments, as well as viewing all important info currently on the calendar. It is broken up into a number of classes that contain different methods, which will be explored in detail below.

Appt Class

The Appt class is a set of methods designed to let the user add appointments to the calendar. The methods are as follows.

- public Appt(int startHour, int startMinute, int startDay, int startMonth, int startYear,String title, String description)
    - This is the constructor for an appointment. It creates an appointment and sets all the class variables to the values it is given. It also checks if the appointment is valid.
    - Pre-conditions: an appointment is being made and all relevant variables have been received
    - Post-conditions: A valid appointment has been constructed and verified.
- private void isValid()
    - Checks to ensure that a constructed appointment is valid. This method is invoked by the constructor. It ensures that all date values are within valid ranges, such as starting hour being within a 24 hour range, and starting month being within a 12 month range.
    - Pre-conditions: appointment received
    - Post-conditions: return false if appointment is not valid, else return true
- public void setRecurrence(int[] recurDays, int recurBy, int recurIncrement, int recurNumber)
    - Sets recurring information. This is used in the circumstance that an appointment is recurring, meaning that it happens multiple times on a predictable schedule.
    - Pre-conditions: recurring appointment received
    - Post-conditions: recurring details for appointment set.
- private void setRecurDays(int[] recurDays)

    - Sets the recurDays variable within the class. This is invoked by setRecurrence. It checks if recurDays is null, setting it to int[0] if so or the given value if not.
    - Pre-conditions: none
    - Post-conditions: recurDays has been set

CalDay Class

The CalDay class holds appointments for a single calendar day. It also has the ability to create new calendar days for assigning appointments to. The methods are as follows.

- public CalDay(GregorianCalendar cal)
    - Constructs a calendar day, setting the day, month, and year. It also creates a linked list of appointments using setAppts, and sets the day validity to true.
    - Pre-conditions: a calendar day is being made an a calendar has been received.
    - Post-conditions: a valid calendar day with all relevant variables set has been made.
- public void addAppt(Appt appt)
    - Adds an appointment to a calendar day. They are sorted by starting time. It does not check if the given appointment occurs on the given day, allowing recurring appointments to be set. It doesn't check for duplicates either.
    - Pre-conditions: appointment received
    - Post-conditions: valid appointment set, or nothing if the appointment was invalid
- private void setAppts(LinkedList<Appt> appts)
    - Sets appointments for the day. This is invoked by the constructor.
    - Pre-conditions: appointment list received
    - Post-conditions: appointments in object set, or nothing if list was null.

CalendarUtil Class

The CalendarUtil class contains various methods that a calendar needs to function. The methods are as follows.

- public static int NumDaysInMonth(int year, int month)

    - Gets the number of days for a month in a certain year. It references a DaysInMonth array containing the standard number of days for each month, January at 0 index and December at 11 index, and also checks for leap years.
    - Pre-conditions: month and year received
    - Post-conditions: number of days returned
- public static boolean IsLeapYear(int year)
    - Checks if a year is a leap year. It follows standard Gregorian Calendar rules, checking if the year is divisible 100 but not 400, or if the year is a multiple if 4. This is called by NumDaysInMonth to check if the given year is a leap year.
    - Pre-conditions: year received
    - Post-conditions: return true if leap year, else return false

TimeTable Class

This class retrieves all appointments between two dates. The methods are as follows.

- public getApptRange()
    - Generates a list of appointments between two days as a linked list of calendar days. It uses a linked list of appointments and two given calendar days. It will throw an out of range exception if any of the days are invalid, or if the second day is not after the first day.
    - Pre-conditions: relevant info received
    - Post-conditions: a valid linked list of appointments has been created, or an appropriate error has been thrown otherwise.
- private static getApptOccurences()
    - Takes an appointment and generates a list of all recurrences of it. The appointments will be in chronological order and between a first day (inclusive) and a second day (exclusive).
    - Pre-conditions: relevant info received
    - Post-conditions: valid list of appointments returned, or returns early if the date range is invalid.
- private static getNextApptOccurrence()
    - Gets the next recurring day for an appointment. This is invoked by getApptOccurences(). It will return null if the given appointment is not recurring. It will also return null if the date fails to be calculated.
    - Pre-conditions:appointment and day received

- Post-conditions: next occurence returned, or error returned if needed

Notes

Note that there are two other classes, but they do not need to be covered in this report. This includes CalendarMain, which contains functions for displaying the calendar, and DateOutOfRangeException, which helps other classes when errors occur.

Some metadata for the library was also grabbed from a JavaNCSS report. The details are as follows.

- Number of classes: 6
- Number of functions/methods: 56
- Non-commenting source statements: 411

Bugs Description

1. In isValid() in Appt.java, I created an off-by-one bug by changing “startMonth-1” on line 113 to just “startMonth.” This bug would likely be noticed in testing, but it’s an easy error to make and pinpointing the cause could be difficult. This would cause the desired month to get screwed up, as inputting January would give February and inputting December would cause an out of range error, since the months are 0-indexed.
2. In getApptRange() in TimeTable.java, I removed lines 38-41, meaning that the method will no longer be able to check if the first date given is before the second. A good testing suite would likely check this, but it’s also an exception that’s easy to forget about. This causes the following code in the method to go out of whack, as it calculates based on the assumption that the first day given comes first chronologically.
3. In CalDay.java, I removed line 64, which set “valid” in the object to true after initializing it. This means that checking if a CalDay object is valid will be impossible, causing a lot of error checking to go awry throughout the method. I included this bug because it’s an error I’ve made before when programming constructors.
4. In CalendarUtil, I removed the math that checks if the given year is divisible by 100 but not 400. I honestly did not know that this was a leap year rule, so I would not have included it if I programmed this myself. This will simply cause a few very

specific years to not be calculated as leap years, so no February 29th will exist for those years.

5. In TimeTable.java at line 55, I changed daysDifference from 0 to 1. This will cause some off-by-1 errors in what days appointments happen on.