Danh Nguyen

04/07/18

CS362

Program Description

The purpose of the Appt class is to create a single appointment that might be store in the xml file depending on if the appointment information is invalid or valid. If valid, it will store information about the appointment start hour, minute, day, month, year, title, description, and email address. The purpose of the CalDay class is to store all of the appointments of a single calendar day. The purpose of the CalendarUtil class is to contain methods that a calendar might use to function. These methods are use to calculate the number of days in a particular month, and check for leap year. DataHandler class acts as a database of the calendar, it is there to allow data to be read and written from memory. The SOLC of Appt class is 387, 261 for the CalDay class, 64 for the CalendarUtil class, and 595 for the DataHandler class. In terms of public and private methods, the Appt class has 30 public and 6 private methods. The CalDay class has 13 public and 4 private methods, CalendarUtil class has 3 public and 0 private methods, and the DataHandler class has 8 public and 8 private methods.

ApptClass

1) Public Appt()
   Description: Constructs a new appointment that starts at a specific time on the date specified. The appointment constructed by default has no recurrence. To set recurrence, setRecurrence() must be call separately.
   Pre-Condition: The package calendar must be imported.
   Post-Condition: Set the hour, minute, day, year, month, title, description, and email of the appointment. By default, the recurring information will be set to never repeat.
2) setValid()
   Description: The purpose of this method is to check whether the provided value of the hour, minute, month, and year provided is valid for the appointment to be created and save.
   Pre-Condition: none
   Post-Condition: set the valid status of the appointment base on the month, hour, minute, and start day provided.
3) setRecurrence()
   Description: This method will set the recurrence information for the appointment just created.
   Pre-Condition: recurs by week/month/year is an integer, how often it recurs every week/month/year is an integer, and the number of recurrences is also an integer.
   Post-Condition: set the days the appointment recurs on, whether it will recur by week/month/year, how often it recurs every week/month/year, and the recurring number which refers to how many recurrences so far.
4) setRecurDays()

Description: set the day of the week that the appointment will recur. If the day of the week provided is empty, then the appoint days of recur is empty.
Pre-Condition: recurDays is within range of 0-6 both in size and value.
Post-Condition: set recurDays of appt.

CalDay

1) public CalDay()
   Description: form by a default constructor and constructor. The default set the calday object to invalid meaning it's not yet a calendar day. While the constructor will set it to valid. The constructor will also set the date of the calendar day which is the day, month, and year as well as the linked list that will contain all appointments within the calendar day.
   Pre-Condition: for default there is none, but for the post-condition calendar day must be one of the GregorianCalendar.
   Post-Condition: Set the date of the calendar day which is the day, month, and year as well as the linked list that will contain all appointments within the calendar day.

2) AddAppt()
   Description: Adds an appointment to the calendar day object. The appointments are kept in order by start time and allow appointment to be added twice.
   Pre-Condition: calDay object must be valid
   Post-Condition: checks if the appointment is valid meaning does it meet the requirement of start time in the ApptClass. Then it reorders the appointments within the calendar day by start time.

CalendarUtil

1) IsLeapYear()
   Description: The purpose of this method is to check whether the year provided is a leap year. A leap year is a year that occurs every 4 years where February has 29 days.
   Pre-Condition: month must be February for the method to function
   Post-Condition: The function returns a Boolean value of true or false that represent whether the year is a leap year.

DataHandler

1) getApptRange()
   Description: Retrieves a range of appointments between two dates. The dates between the two dates as well as their respective appointments will be store in a list.
   Pre-Condition: if date 2 and date 1 must be within range.
   Post-Condition: Throws out the given dates if day number 2 is before day number 1.
2) getApptOccurences()
   Description: Figure out which days the appointment occurs on
   Pre-Condition: date 2 is not after date 1

Post-Condition: make sure that the first day of the appointment is before the last day of the appointment.

3) getNextApptOccurences()
   Description: Calculates the next recurrence day of the given appointment
   Pre-Condition: appt given must be within the range of the date1 and date2.
   Post-Condition: If appointment doesn't reoccur, or if the date is out of range then it returns null. Else it returns the day of the week of the recurrence specified by recur variable.
4) saveAppt()
   Description: Saves an appointment's information to the XML data tree.
   Pre-Condition: getValid function must be valid for the appointment to be store.
   Post-Condition: The information of the appointment such as the StartHour, StartMinute, StartDay, StartMonth, StartYear, title, description, emailaddress, recurdays, recurdaysstrings, recurby, recurIncrement, and recurNumber will all be store in a node element.

Bugs Description

Method #1: setValid() inside the Appt class

Line #172

What I changed: I change "startHour < 0" to "startHour > 0"

Unexpected/Incorrect Result: The logical bug I introduced was that any appointment with which the start hour of anything greater than 0, then that appointment is invalid.

Method #2: getNextApptOccurrence() inside the DataHandler class

Line #331

What I changed: I change "(!appt.isRecurring())" to (appt.isRecurring())

Unexpected/Incorrect Result: This means that any appointment that was set up to have recurring dates, now always have no reoccurrence notice.

Method#3: NumDaysInMonth() inside the CalendarUtil class

Line #32

What I changed: I changed "(IsLeapYear(year) && (month == FEBRUARY))" to " "(IsLeapYear(year) || (month == FEBRUARY))

Unexpected/Incorrect Result: This means that the month February of every calendar will be a leap month.

Method #4: setAppts() inside the CalDay class

Line #117

What I changed: I changed “if(appts != null)” to “if(appts == null)”

Unexpected/Incorrect Result: This means that an appointment in the linked list of the some calendar day can be an appointment that in invalid. Meaning the appointment has incorrect information.

Method #5: toString() inside the Appt class

Line#377

What I changed: I changed “if(!getValid())” to “if(getValid())

Unexpected/Incorrect Result: This means that any appointment which are valid is printed out as an appointment that is not valid