CS362

Assignment 1

DONGHAO LIN(lindo)

Program Description

The calendar application is supposed to represent appointments that might be stored in xml file with specific time, title, description and email address.

Appt class represents a single appointment that might be stored in an xml file. The appointment consists of startHour, startMinute, startDay, startMonth, startYear, title, description, and emailAddress. The source line of code is 237, and there are 29 public methods and 6 private method in this class.

CalDay class stores all of the appointments of a single calendar day. It also has some useful calendar-related abilities, such as the ability to create a new calendar day that is incremented by a day. The source line of code is 173, and there are 12 public methods and 4 private methods in this class.

CalendarUtil class is containing utility methods that a calendar might need to function. The source line of code is 35, and there are 2 public methods in this class.

DataHandler class handles all data that is read and written from disk. The source line of code is 447, and there are 7 public methods and 8 private methods in this class.

DateOutOfRangeException class checks whether dates are out of range and sent message. The source line of code is 8 and there are 2 public methods in this class.

XmlParserErrorHandler class is to report warnings and errors with the XML Parser.

It is modeled after Java's sample code. The source line of code is 27 and there are 3 public methods and 1 private method in this class.

Appt class:

public Appt(…) # the constructor of the class Appt

pre-condition: no

post-condition: Constructs a new appointment that has no start time on the date specified. The appointment is constructed with no recurrence information by default. To set recurrence information, construct the appointment and then call setRecurrence(...) method. The XmlElement will be set when the appointment is saved to disk. Set parameters.

setValid(....) # the public method

pre-condition: no

post-condition: returns true if appointment valid or false if appointment invalid

setRecurrence(...) # the public method

pre-condition: recurDays >=0, recurBy>=0, recurIncrement >=0, and recurNumber>=0

post-condition: Sets the recurring information with the correct recurDays, recurBy, recurIncrement, recurNumber.

setRecurDays(...) # the private method

pre-condition: recurDays>=0

post-condition: if recurDays is NULL returns a new integer array, else returns recurDays.

CalDay class:

publicCalDay(...) # the constructorof the classCalDay.

pre-condition: no

post-condition: Creates a new CalDay which is ready for appointments to be added to it. Note that months are numbered 0 through 11. This does not check to see if a date is valid.

AddAppt(...) # the public method

pre-condition: appt is valid

post-condition: Adds an appointment to the calendar day object. The appointments are kept in order by start time.

Class CalendarUtil:

IsLeapYear((...) # the public method

pre-condition: year>0

post-condition: if year can be divided by 100 and then 400 reminds 0 and can be divided by 4 reminds 0, then returns true, else returns false. Set parameter year, and

return true if the year is a Leap Year, false otherwise.

Determines if the specified year is a Leap Year

Class DataHandler:

getApptRange(...)# the public method

pre-condition: firstday < lastday

post-condition: return a list of all of the CalDays between firstDate (inclusive) and lastDate (exclusive) with their respective appointments. If the data handler has not been initialized correctly, null is returned. throws DateOutOfRangeException If any of the days constructed by the given values are invalid, or if date 2 is not after date 1.

Retrieves a range of appointments between two dates**.**

getApptOccurences(...) # the private method

pre-condition: Appt, firstday and lastday are valid.

post-condition: constructs a linked list of GregorianCalendar's, each of which represent a day when the appointment occurs. The days are guaranteed to be between firstDay (inclusive) and lastDay (exclusive). They are guaranteed to be in order.

getNextApptOccurrence(...) # the private method

pre-condition: Appt is valid and day>=0

post-condition: Calculates the next recurring day in the given appointment. If the appointment does not recur it returns null. If the date cannot be calculated for some reason it returns null.

saveAppt(...) #the public method

pre-condition: Appt is valid

post-condition: Saves an appointment's information to the XML data tree. Does not write a new XML file to disk. And return true if the appointment was saved correctly.

Bugs Description

setValid(..) in the Appt.java:

I changed startDay < 1 with startDay < 2 in this method in line 180. There is nothing changed and incorrect in the calendar.

setRecurDays(…)in the Appt.java:

I changed this.recurDays = new int[0] to this.recurDays = new int[1] in this method in line 307. Then there is nothing changed and incorrect in the calendar.

represntationApp(…) in the Appt.java:

I changed printableHour > 11 to printableHour >=12 in this method in line 362, then there is nothing changed and incorrect in the calendar.

addAppt(…)in the CalDay.java:

I changed int i = 0 to int i = 1 in this method in line 78, then the time order become wrong in the calendar.

getNextApptOccurrence(...) in the DataHandler.java:

I changed int i = 0 to int i = 1 in this method in line 301, then there is no event on the calendar.