CST8132 Assignment 01 – Event Planner

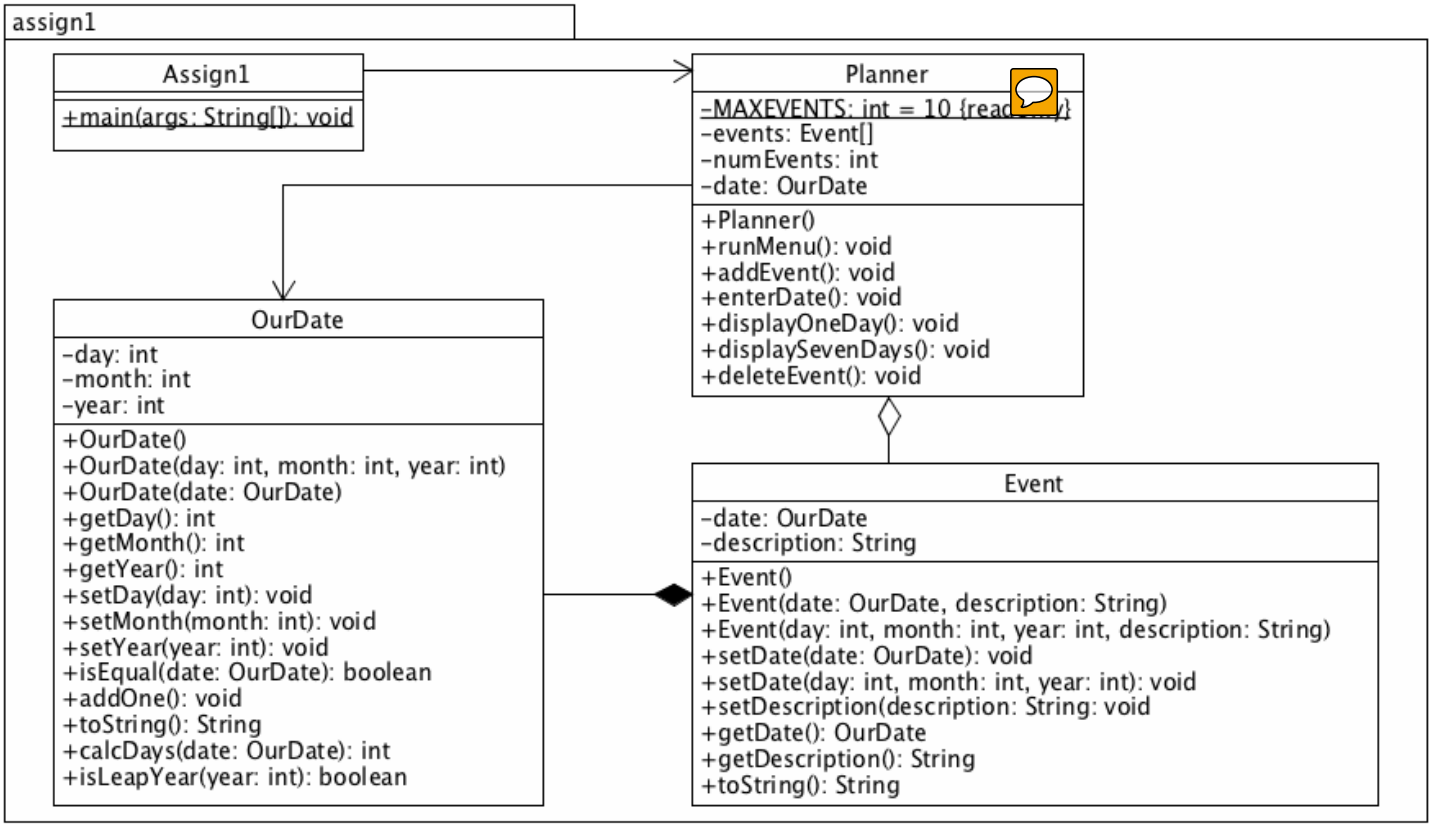
Due: Friday February 12, 2016 by 11:59pm as upload to Blackboard

Note: The upload link will disappear immediately after the due date/time.

Problem Description

Develop a tool to keep track of events – a basic electronic planner of sorts. The Planner will be able to add events, delete events and view events either for a specific date or for seven days starting at a specific date. The planner is able to calculate how many days exist between events. An Event will be stored in an array (set maximum size for now to be 10). There can only be one Event for a given date. Sorting, deleting and retrieving elements (in this case, the elements are references to Event objects) *efficiently* in an array will be a major topic in your third semester Data Structures course; for now, events will be stored in an array and do not need to be sorted. This means we will just search sequentially through the array. Every Event will contain a date (day, month, year) and an activity description (String). All input entered from the keyboard must be fully edited for valid data. In the case of this assignment, if a user attempts to enter data that is outside of the appropriate range, by default the field will be set to the first day, month or year of the acceptable range of values. For example if a user attempts to set a day to 34, it will automatically be set to 1. NOTE: Only Planner interacts with the user, no other classes should use Scanner or println (etc.)

Program Design

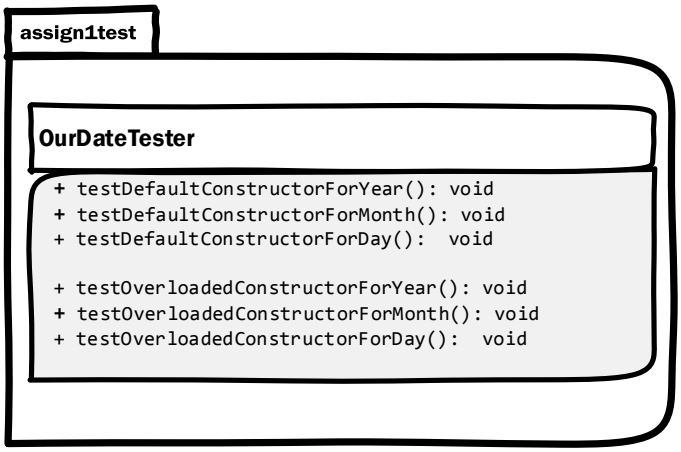


In order to bind the scope of the assignment, you are required to build a system that:

- **Assign1** contains only a main method, which instantiates one Planner and calls runMenu().
- **Planner** has a private int constant which is used for the maximum amount of events, an array for Event references (10 references), an int and an OurDate reference, one constructor and six methods:
  - runMenu() - shows the menu, interacts with the user for menu selection, loops till they exit.
  - addEvent().
    - verifies that the maximum number of events is already in the Planner
    - loops through the array to confirm that an event for the date specified is not already taken.
    - If there is not already an event on the date specified, add the event to the array and add one to the numEvents counter.
  - enterDate() – fetches the date from the user and assigns values to the fields of each reference by invoking methods of the OurDate class.
  - displayOneDay() - displays the event for the date specified.
  - displaySevenDays() – displays seven days of events by invoking displayOneDay() seven times, starting with the event with the date specified.
  - deleteEvent()
    - loops through the events array, searching for event by the date.
    - If the event is found, delete it and move all the event references up one, so as not to leave a null element in the middle of the array. Remove one element from the numEvents counter.
- **Event** has an OurDate reference, a String reference, three constructors and six methods:
  - Two ***overloaded*** setDate() methods.
  - setDescription().
  - getDate().
  - getDescription().
  - toString() - returns a String representation of the Event reference.
- **OurDate** has three fields, day, month and year, three constructors, and eleven methods:
  - setDay(), setMonth(), setYear(), each must verify that the day, month or year respectively is valid.
  - getDay(), getMonth(), getYear(), returns day, month or year, respectively.
  - isEqual() - compares two dates to see if they are equal..
  - toString() - returns a String representation of the OurDate reference.
  - addOne() – adds one day to the current date, checks for boundaries such as last day month, year, etc..
  - isLeapYear() - determines if the year specified is a leap year or not – you may use an existing algorithm (found online ?) for this, BUT…be certain to reference the source in a comment.
  - calculateDays() - calculates the amount of days from a base year (in this case January 1, 2000) to the date specified. Invokes the isLeapYear() method.

Testing Design

Sufficiently testing all relevant parameters in this assignment would require a large amount of unit tests. Consequently, this assignment is going to confine the tests required, to those specified in the OurDateTester UML. Only the default (i.e. no formal parameters) constructor, and the overloaded, three input constructor require testing. As indicated, there should be one test for each field of the class OurDate. Each unit test should have an appropriate message if a test fails.



Discussion Questions

- How did the use of an array to manage references to Event objects help in creating this program? (Would things have been easier with 10 separate variables, what about with 1000 Event objects?)
- What is the difference between Aggregation, and Composition? What is the relationship between Planner and Event and Why?
- What are the benefits of separating the test classes from the program classes using packages?
- How would automated testing help when changes are made to the software in future?
- List a few other features to test in the OurDate class with JUnit.

Comments Note

• At the top of each source code file include the following comment header, adding the needed information:	/* File Name: * Course Name: * Lab Section: * Student Name: * Date: */
• Classes and class members (class level fields, constructors, methods) should have a brief description as a comment immediately above in the code listing. Example:	/* * Represents a, electronic Planner to keep track of * Events. */ **public class** Planner{

(Javadoc comments are not required for Assignment 1)

Tasks

☐ Build the program, do not overlook the package name used for the program classes.
☐ Build the JUnit test classes, use a separate package as indicated in the UML.
☐ Write one-page essay, using MS Word, that addresses the discussion questions (below) as a guide to discuss what was accomplished in the lab, cite and reference any sources used online to help you answer questions.

Submission Requirements

- Place source code files and essay document into a zip archive and upload it to Blackboard by the due date.
- The essay should be an electronic document, Microsoft Word format, include your name inside the document.
- Your lab professor will indicate any additional submission requirements to you in the lab

Grading Rubric

Criteria* (Equal Weight)	Needs Work (0)	Poor (1)	Intermediate (2)	Excellent (3)	Value Scored
Naming	Classes and class members follow no Java naming conventions.	Java naming conventions are not well followed.	Classes and class members follow Java naming conventions with tiny inconsistencies.	All classes, class members are named following Java naming conventions, consistently and perfectly.	
Comments (Javadoc not required Assign 1)	Comments missing or incorrect.	Many classes and / or class members missing meaningful comments	Very few classes and / or members missing comments, comments are meaningful.	Nearly everything is commented, comments are meaningful, brief, well written.	
Discussion	Missing or only repeats the questions	Provides brief answers to questions	Questions are answered and examples are used from the written code to illustrate	Questions are answered with examples from code, student discusses how concepts in this program will be beneficial in future programing.	
Citations	No referenced work cited when it needed to be. See Algonquin College Policy AA20 on Plagiarism.	References and citations present but not APA style, e.g. only the URL provided.	References and citations loosely follow APA style	References and citations closely follow APA style	
Compiles	Program does not compile, too many syntax mistakes for the professor to track or debug without major re-write.	Program does not compile, has several syntax mistakes	Program does not compile, has a few small syntax mistakes	Program compiles	
Execution	Program is missing much functionality. For example program starts to run but does not work correctly.	Program is missing much code, and much of the required concepts.	Program demonstrates most of the concepts, some parts left out.	Program demonstrates understanding and application of concepts notably manipulation of a 1 dimensional array, random numbers, and composition.	
Unit Tests	Unit tests omitted or largely incomplete	Unit tests work but code is not well organized (within each @Test method)	Unit tests work and each @Test method has well organized code	Unit tests work, each @Test method has well organized code, and meaningful variable names used to improve readability.	
				Max(21):	
				Total:	

Notes on citations and references

- Please do not cite or reference other students, ask for the original sources from them and cite and reference those instead
- You will not get credit for an assignment or project if large portions are copied directly from other sources, even if you cite and reference the source(s) correctly. Assignments are to be your own original work; other works can be used for help in solving problems or as small pieces of your larger program. Determinations on this are up to the discretion of the professor, if in doubt check with your professor.
- Note: One exception to this is in the case of lecture and lab handouts, and code samples posted to Blackboard. You are free to use these as a starting point just cite + reference them as a personal communication from your professor e.g. Stanley Pieda (2016) personal communication.

Appendix – Additional Notes

JUnit Testing – Learning Resources (See Blackboard for larger list)

McProgramming. (2014). Java - JUnit testing in Eclipse. [Video] Retrieved from https://www.youtube.com/watch?v=I8XXfgF9GSc

Lars Vogel. (2015). Unit Testing with JUnit – Tutorial. Retrieved from http://www.vogella.com/tutorials/JUnit/article.html

UML Class Diagram resource:

Donald Bell. (2004). UML basics: The class diagram. Retrieved from http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/#N100A5

Sample Program Run (user inputs are bold and highlighted):

```
Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 1
Enter event description: rock concert
Enter event date:
Enter year: 2015
Enter month: 3
Enter day: 4

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 1
Enter event description: work
Enter event date:
Enter year: 2015
Enter month: 3
```

```
Enter day: 3

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 1
Enter event description: dinner at folks
Enter event date:
Enter year: 2015
Enter month: 3
Enter day: 5

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 1
Enter event description: movie night
Enter event date:
Enter year: 2015
Enter month: 3
Enter day: 4
You already have an activity for that date and time...cannot be entered

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 2
Enter date to display:
Enter year: 2015
Enter month: 3
Enter day: 5
Your events for 2015/3/5 are:
     2015/3/5 - dinner at folks

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 3
Enter starting date to display:
Enter year: 2015
Enter month: 3
Enter day: 1
Your events for the week starting 2015/3/1 are:
Your events for 2015/3/1 are:

Your events for 2015/3/2 are:
```

```
Your events for 2015/3/3 are:
     2015/3/3 - work

Your events for 2015/3/4 are:
     2015/3/4 - rock concert

Your events for 2015/3/5 are:
     2015/3/5 - dinner at folks

Your events for 2015/3/6 are:

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 4
Enter date to delete event from:
Enter year: 2015
Enter month: 3
Enter day: 4

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 3
Enter starting date to display:
Enter year: 2015
Enter month: 3
Enter day: 1
Your events for the week starting 2015/3/1 are:
Your events for 2015/3/1 are:

Your events for 2015/3/2 are:

Your events for 2015/3/3 are:
     2015/3/3 - work

Your events for 2015/3/4 are:

Your events for 2015/3/5 are:
     2015/3/5 - dinner at folks

Your events for 2015/3/6 are:

Make a selection:
1.  Add event to planner
2.  Display event for a day
3.  Display events for a week
4.  Delete an event
0 to quit: 0
Goodbye
```