DUE DATE: OCTOBER 27, 2021 AT 4:59PM

Instructions:

- You must complete the "**Blanket Honesty Declaration**" checklist on the course website before you can submit any assignment.
- Only submit the **java files**. Do ***not*** submit any other files, unless otherwise instructed.
- To submit the assignment, upload the specified files to the **Assignment 2** folder on the course website.
- Assignments must follow the **programming standards** document published on UMLearn.
- After the due date and time, assignments may be submitted but will be subject to a late penalty. Please see the ROASS document published on UMLearn for the course policy for late submissions.
- If you make multiple submissions, only the **most recent version** will be marked.
- These assignments are your chance to learn the material for the exams. Code your assignments independently. We use software to compare all submitted assignments to each other, and **pursue academic dishonesty vigorously**.
- Your Java programs must compile and run upon download, without requiring any modifications.
- Automated tests will be used to grade the output of your assignment. If you do not follow precisely the guidelines described below, these automated tests can fail and you will lose marks. In particular, **make sure that your methods are spelled exactly as described below**. Also**, make sure that your code produces exactly the example outputs (including correct spacing)** shown in these guidelines.

Assignment overview

In this assignment, you will build a simplified calendar application. You will also practice reading from a file and handling exceptions.

Testing Your Code

We have provided sample test files for each phase to help you verify that your code is working to the assignment specifications (e.g., correct method headers). **These files are only *starting* points for testing your code.** Part of developing your skills as a programmer is to think through additional important test cases and to write your own code to test these cases. The test files should give you a sense what this additional code could consist of.

Phase 1: A Simplified Calendar

In this Phase, you will create a Calendar class that uses a 3-dimensional array to store several Entries. To keep this assignment tractable, we are going to make a lot of simplifications as compared to calendar programs that are available in the real world. For example, we will assume that each entry in the calendar begins on the hour, and that leap years do not exist. We will also only allow 1 entry in the calendar to begin at a given time on a given day (however, we will allow entries to overlap).

First, implement a simple Entry class to represent an entry in the calendar.

The **`Entry`** class should have:

- Three instance variables: the entry's name (a **`String`**), the start time (an **`int`**), and a duration in minutes (an **`int`**). In our simplified calendar, an entry will always start on the hour and will use a 24-hr clock (as opposed to representing time using AM/PM).
- A constructor that has three parameters, in the above order, which are used to initialize the three instance variables.
- Three standard accessor methods for each of the three instance variables above: **`getName()`**, **`getStartTime()`** and **`getDuration()`.**
- A standard **`toString`** method which returns a **`String`** containing the name of the entry, the start time, and the duration, formatted as shown in the sample output.

- A **boolean overlaps(Entry)** method that returns true if the two Entries overlap (i.e., one will run past the start time of the other) and false of they do not. An entry that finishes exactly at the start time of the other entry would not be considered to overlap.
- A **boolean equals(Entry)** method that checks if two Entries are equal. Two entries are equal if they have the same name, start time and duration.
- A standard **clone()** method that returns a new Entry instance with a deep copy of the original values for its instance variables.

Next, implement a Calendar class that will store and provide operations on Entries.

The **Calendar** class should have the variables and methods described below. All methods listed below should be public.

- Three instance variables: the name (**String**), a year (**int**) and a 3D array to store Entries according to [month][day][startTime]. Hint: the Temperature example from Week 5 has a very similar structure. You can add additional class constants as needed to improve the readability of your code.
- A constructor that accepts two parameters: a name (**String**) and a year (**int**). The constructor should initialize the name and year instance variables and allocate space for the 3D array. Hint: the constructor in the Temperature example will be useful here as well.
- Two static **boolean** methods: **isValidTime(int)** and **isValidDate(int month, int day)** that returns true if the parameters are valid and false otherwise. A valid time is between 0 and 23 inclusive. A valid month is between 1 and 12 inclusive and a valid day depends on the month. For example, month 1 (i.e., January) has 31 days whereas month 2 (i.e., February) has 28 days.
- A method **void addEntry(int month, int day, Entry entry)** that adds the entry passed as a parameter to the 3D array at that month, day and the entry's start time. If there is already an entry at that position in the array, it should be overwritten by this new entry.
- A method **Entry getEntry(int month, int day, int time)** that returns a reference to the Entry object at that month, day and time in the calendar. The method should return **null** if no such Entry object exists.
- A **boolean isConflicting(int month, int day, Entry entry)** method that returns true if the entry passed as a parameter would conflict with another calendar entry on that day. An entry conflicts with another entry on that day if they start at the same time or overlap. You do not need to check other calendar dates for entries that would overlap.
- A **void displayEntries(int month, int day)** that displays all entries in the calendar on that date. If there are no entries on that date, the method should display a message indicating so. The method should also display a message if the date is not valid. See the sample output for wording and formatting.
- An **int getDurationOfEntriesOnDay (int month, int day)** method that returns the total duration (in minutes) of all entries in the calendar on that date. The method should return 0 if the date is invalid.
- An i**nt getNumEntries(int month, int day)** that returns the number of entries on that date. The method should return 0 if the date is invalid.
- An **int getNumEntries()** that returns the number of entries in the calendar.
- A **boolean deleteEntry(int month, int day, Entry entry)** method that deletes the specified entry on that day in the calendar (if it exists). The method should return true if the operation was successful and false if there was no such entry.
- A **boolean deleteAllOccurences(Entry entry)** that deletes all occurrences of the entry in the calendar. The method should return true if at least one such entry was deleted.
- A **toString()** method that returns a String containing the calendar's name, year and number of entries (formatted as shown in the sample output).

You can begin testing your class with **TestPhase1.java**. You should get the output shown below.

```
Created 5 entries:
e1: 9:00: Dentist Appointment, 180 mins
e2: 11:00: COMP 1020 class, 60 mins
```

```
e3: 12:00: Graduate student meeting, 50 mins
e4: 11:00: COMP 1020 class, 60 mins
e5: 19:00: WSO concert, 60 mins
Calling accessor methods for e1:  Dentist Appointment 9 180
e1 equals e2? false
e1 overlaps with e2? true
e2 overlaps with e3? false

Checking some times and dates:
Is 0 a valid time?: true
Is 25 a valid time?: false
Is 1/15 a valid date?: true
Is 12/31 a valid date?: true
Is 14/15 a valid date?: false
Is 2/30 a valid date?: false

Creating a new calendar:
Sophie's Calendar, 2021, Number of Entries: 0

Adding three entries on two different days
Sophie's Calendar, 2021, Number of Entries: 3
Entries on 2/14:
9:00: Dentist Appointment, 180 mins
12:00: Graduate student meeting, 50 mins
Entries on 1/29:
11:00: COMP 1020 class, 60 mins
Entries on 3/17:
No entries on this date.
Entries on 2/30:
2/30 is invalid.
Event on 2/14 at 9:00: 9:00: Dentist Appointment, 180 mins
Event on 2/14 at 10:00: null
Number of entries on 2/14: 2 for a total duration of 230 mins.
Number of entries in the calendar: 3

Checking conflicts
Does e2 conflict with any entries on 2/14? true
Does e5 conflict with any entries on 2/4? false
Does e1 conflict with any entries on 2/15? false

Deleting e1 from 2/14
As expected, entry was deleted
Entries on 2/14:
12:00: Graduate student meeting, 50 mins
Sophie's Calendar, 2021, Number of Entries: 2
Trying to Delete e2 from 2/14
As expected, entry was not deleted
Sophie's Calendar, 2021, Number of Entries: 2

Adding e1 to multiple dates
Sophie's Calendar, 2021, Number of Entries: 6

Deleting all occurrences of e1
Sophie's Calendar, 2021, Number of Entries: 2
Entries on 9/4:
No entries on this date.
```

Phase 2: Reading from a file and Exception handling

In this phase, you will modify your Calendar class so that it can read in Entry information from a file. The file will have the following format: The first line contains the name of the calendar. The second line contains the year. Each line after corresponds to an entry to add to the calendar. Each of these lines will consist of the following information separated by commas: entry name,start time,duration,month,day.

Example file:

```
Bob’s Calendar
2019
Coffee with mom,11,60,2,11
Ballet recital,2,120,3,3
….
```

Make the following modifications to your Calendar class

- Add a **private void** method **processEntryLine**(**String** line) that accepts a calendar entry line as formatted above and adds the entry information to the calendar. The method should throw an **Exception** if the date or start time are invalid. The method should also throw an Exception if parts of the line cannot be converted to integers as required (e.g., using **Integer.parseInt(String)**). Add an appropriate message to your Exception (before throwing it) as shown in the sample output below.
- Create a second constructor that accepts a filename (**String**) as a parameter and uses the information in the file to initialize the calendar. Your constructor should call your **processEntryLine** method, and handle any exceptions that this method throws by outputting the Exception’s message to the console.

You can begin testing your class with `TestPhase2.java`. You should get the output shown below. Start with the calendar1.txt file, which does not contain any errors, and move onto trying calendar2 file once you have your code working with the “error-free” file.

```
Creating a new calendar by reading from a file
Celeste's Calendar, 2019, Number of Entries: 23
Entries on 1/29:
12:00: Lunch with Mom, 90 mins
Entries on 2/11:
8:00: Running group, 120 mins
10:00: Coffee with Al, 60 mins
11:00: Pysch Midterm, 90 mins
18:00: Dinner with friends, 150 mins

Creating a second calendar by reading from a file
Problem with line format: For input string: "a"
Problem with entry: Invalid date: 29/29
Problem with entry: Invalid start time: 30
Bob's Calendar, 2021, Number of Entries: 5
Entries on 1/29:
No entries on this date.
Entries on 2/11:
10:00: Coffee with Al, 60 mins
11:00: Pysch Midterm, 90 mins
```

Hand in

Submit your two Java files (`Entry.java`, `Calendar.java`,). ***Do not submit .class or .java~ files!*** You do ***not*** need to submit the `TestPhaseN.java` files that were given to you or the .txt files. If you did not complete both phases of the assignment, use the Comments field when you hand in the assignment to tell the marker which phases were completed, so that only the appropriate tests can be run. For example, if you say that you completed Phase 1, then the marker will compile your files and also TestPhase1.java, and then run the main method in TestPhase1. If it fails to compile and run, you will lose ***all*** of the marks for the test runs. The marker will ***not*** try to run anything else, and will ***not*** edit your files in any way. (***Make sure none of your files specify a package at the top)***