Major Program 1

COMP 167

Due 10/01/2021 at 11:59 pm

Introduction

This program will require you to create a program that will allow faculty members to post their course and office hour schedules.

Enumeration

First of all, this project utilizes enumerated types which help your program's readability. Here is an example:

```
public enum DaysOfWeek {
  SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY;
}
```

The code above would appear in a separate file named DaysOfWeek.java and would serve as the data type for variables that could only have the values: `SUNDAY`, `MONDAY`, `TUESDAY`, `WEDNESDAY`, `THURSDAY`, `FRIDAY`, `SATURDAY`. Notice, these are not Strings! Internally, `SUNDAY=0`, `MONDAY=1`, `TUESDAY=2`, `WEDNESDAY=3`, `THURSDAY=4`, `FRIDAY=5`, `SATURDAY=6`. However, the enumerated type is more elegant way of representing these values. The following examples should also prove helpful to you:

```
DaysOfWeek firstDayOfWeek = DaysOfWeek.SUNDAY;
```

You can also retrieve the String value of an enumeration. The following example would give the String `value` the value `"SUNDAY"`:

```
String value = firstDayOfWeek.name();
```

UML Class Diagrams

Here are the UML Class Diagrams for the classes you must implement in Java. You are free to add additional private methods if needed.

TimeBlock	
-day:DaysOfWeek	Day of time block
-startTime:int	Start time of block
-endTime:int	End time of block
-comments:String	Comments about the time block.
-location:String	Location where the time block is spent
+TimeBlock()	Set defaults
+//mutator and accessor methods	
+getFormatedTimeBlock() : String	See Comment below.
+toString():String	Use same format as the input file.

The method `getFormatedTimeBlock()` should return a string in the following format:

```
startTime - endTime comments location (e.g. 1200 - 1300 COMP167 ACB 207 )
```

Appointment	
-description:String	Description of the appointment
-timeBlock:TimeBlock	day and times of appointment.
+Appointment()	Set defaults and instantiate the TimeBlock
+//mutator and accessor methods	
+toString():String	Use same format as the input file.

Course	
-courseName:String	Course name
-location:String	Building and room number where course is conducted
-timeBlocks:ArrayList<TimeBlock>	Course days and meeting times.
+Course()	Set defaults and instantiate the ArrayList
+//mutator and accessor methods	
+toString():String	Use same format as the input file.

Faculty	
-firstName:String	Faculty first name
-lastName:String	Faculty last name
-officeLocation:String	Campus building and office number
-courses:ArrayList<Course>	Information on courses taught.
-officeHours:ArrayList<TimeBlock>	Information on office hour sessions.
-appointments:ArrayList<Appointment>	Information on miscellaneous appointments and meetings.
+Faculty()	Set defaults and instantiate the ArrayList
+//mutator and accessor methods	See "Handling ArrayList"
+getCalendar() : String	See note below.
+toString():String	Use same format as the input file.

The method getCalendar() should return all course items, office hour items and appointment items. Each item (i.e. formatted TimeBlock) should be listed under a heading with the day of the week. Within a particular day, the items should be listed in sorted order by time (Hint: Sorting is not necessary--use a loop that goes from 5 to 2400 by 5's.).

Department	
-departmentName:String -unitName:String -universityName:String -faculty:ArrayList<Faculty>	Name of the department (e.g. Computer Science) Name of the unit/college (e.g. Engineering) Name of the University All of the faculty in the department
+Depatment() +//mutator and accessor methods +loadDepartmentData(String inputFileName) : void +saveDepartmentData(String outputFileName): void +atAGlance(int time) : String +toString():String	Default values for properties, instantiate each ArrayList See "Handling ArrayList" section Read in the bank data from a file (See note below on input file) Write bank data to a file (See note below on output file). Returns a string showing what each Faculty member is doing at the given time. Creates a string with all Department that matches the format of the input file (See input file section).

The toString() method

The toString method should return a String formatted as in the input file. Notice that TimeBlock.toString() will not include the location or comment properties. Most classes will have each property on a new line, TimeBlock properties will be separated by a comma.

Handling ArrayLists

Each ArrayList should have five associated methods to perform: getNum, add, get, set and remove. So if you have an ArrayList named widgets that stored items of type Widget, then the associated UML would be:

```
+getNumWidgets() : int //Return the number of items in the ArrayList widgets.
+getWidget(index:int) : Widget //get the Widget at location index in ArrayList widgets
+setWidget(index:int, item:Widget):void //store item at location index in the ArrayList widgets.
+removeWidget(index:int):Widget //remove the Widget stored at the given index and return it
```

```
+addWidget(item:Widget):void //Append the Widget to the ArrayList.
```

Input File

The input file will be read by the user using any file reader and input streams (BufferedReader or Scanner) to your main() method using command-line arguments. The format for the input file is shown below:

```
Department Name
College Name
University Name
Faculty0 First Name
Faculty0 Last Name
Faculty0 Office Location
Faculty0 Number of Courses
Course0 Name
Course0 Location
Course0 Number of meeting days
Course0 Day of the week, Start_Time, End_Time
* repeat for other days
* repeat for other courses
Faculty0 Number of office hour sessions
Office Hours0 Day of the week, Start_Time, End_Time
* Repeat for other office hours.
Faculty0 Number of Appointments
Appointments0 Description
Appointments0 Day of the week, Start_Time, End_Time
* Repeat for other appointments.
* Repeat for other faculty

The file will end when there are no more faculty.
```

You should also reference the input file provided in this repository.

Notes

- When outputting office hour information, use the string "Office Hours" as a description.
- All start and end times will be a multiple of 5.

Output File

The format of the output file is the same as the input file. If you have created the toString() method for the Department class correctly, the produced String should match the input/output format. This should make this method trivial to write.

Grading

Level 1: 25 points

Complete the DaysOfWeek enumerated type, the TimeBlock class, the Course class and the Appointment class. Write a simple driver program that will instantiate an object of each class type, populate the data fields and test the other methods of the classes. Display your output.

Level 2: 10 points

Complete all of the Faculty class except for the getCalendar() and atAGlance() methods.

Level 3: 20 points

Complete all of the Department class except for the loadDepartmentData() and saveDepartmentData() classes.

Level 4: 30 points

Implement the loadDepartmentData() and saveDepartmentData() methods. Your main() method should be able to read from the input file.

Level 5: 15 points

Implement the getCalender() and atAGlance() methods.