

CSCI1250 INTRODUCTION TO COMPUTER SCIENCE 1

PROJECT 4 – SPRING 2018

THE PROJECT

We wish to keep track of student advisees. Each advisee consists of a name, student id, concentration, number of hours completed, name of advisor, whether or not they have completed a major sheet for graduation and whether or not they have filed an intent to graduate. An advisor needs the ability to:

- update as well as access each advisee's information
- display all student advisees in the system including all information formatted as shown in the example
- display a list of student advisees who have been cleared for graduation

ADVISEE CLASS DESCRIPTION

You are to build the Advisee class. Begin by creating the UML diagram that includes the following:

- **The fields necessary to maintain an advisee:**
 - **name, string** (default value ETSU Student) – this is the name of the student advisee
 - **student id, string** (default value E#####) – this is the student id for ex. E0123456
 - **concentration, string** (only available concentrations are CS, IS, IT) (default value XX) – this is the student's major concentration. This is for Computing majors only at this point.
 - number of **hours completed, int** (default value 0) – this is the number of college credit hours the student has completed
 - **advisor, string** (default value No Advisor) – this is the name of the advisor
 - major sheet, **boolean** (default value **false**) – this tells if they have filed their major sheet for graduation
 - **intent to graduate, boolean** (default value **false**) – this tells if they have filed their intent to graduate

- **Constructor methods**

- A default constructor (should set all attributes to default values as shown above)

Note: the constructor methods should call the appropriate set methods rather than assigning values directly.

For example, in the constructor, instead of assigning the default value for the name directly using

```
name="XXX XXXXX"
```

call the setName method instead sending it the values for the default name

```
setName ("XXX XXXXX") ;
```

- A constructor accepting a value for all attributes
- A copy constructor

CSCI1250 INTRODUCTION TO COMPUTER SCIENCE 1

PROJECT 4 – SPRING 2018

- **Accessor and mutator methods for each attribute**
 - The mutator for concentration should only allow the following values for concentration:
 - CS, IS, IT (capitalization matters...it can come in upper or lowercase, but should be stored as uppercase)
 - If something other than these values is sent to the mutator, it should be assigned XX instead.
 - The mutator for the student **id** should **only allow student id's that are 8 characters long and begin with an E**
 - If something other than these values is sent to the mutator, it should assign the default value instead (E#####)
- **Methods to perform the following operations (as mentioned above)**
 - determine what a student advisee's classification is based hours completed
 - should be called **getClassification**
 - returns the **string classification**
(It should return one of the following strings: Freshman, Sophomore, Junior or Senior)
 - The classification can be determined as follows:
 - for an undergraduate: Freshman < 30 hours, Sophomore 30-59 hours, Junior 60-89, Senior 90 and up
 - determine if a student has met the requirements to graduate
 - should be called **metGraduationRequirements**
 - **returns a boolean** – true if they have met all requirements, **false if they** have not
The graduation requirements are as follows:
 - student must have completed at least **120 credit hours**
 - student must have filed an intent to graduate
 - student must have filed a major sheet
 - create a message that will tell whether or not the student has been cleared to graduate
 - should be called **clearedToGraduateMsg**
 - returns the string message stating whether or not they have been cleared to graduate based upon whether or not they have met the graduation requirements
 - If the student has met all graduation requirements, the message should state “Yes – all requirements have been met”
 - If the student has not met all graduation requirements, the message should state “No – “ followed by as many of the following that apply:
 - not enough hours;
 - not completed major sheet;
 - not filed intent to graduate

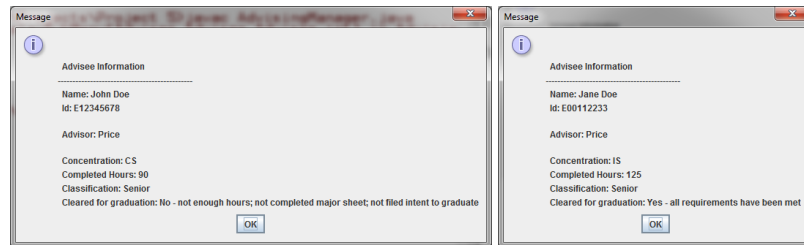
(See the example screenshots below for examples of the output)

 - Note: this method does NOT display the statement, it simply creates a string that holds the information in it. This string will be returned to the calling method and the calling method may then display the string.

CSCI1250 INTRODUCTION TO COMPUTER SCIENCE 1

PROJECT 4 – SPRING 2018

- Create an **equals** method to determine if two advisees are the same
- give a student advisee report including all information formatted as shown in the example
 - should be called **toString**
 - returns a string containing the student advisee's information
 - the string should include all of the information as shown in the example and be formatted as in the example.
 - Note: this method does NOT display the student advisee's record, it simply creates a string that holds all of the student advisee's information. This string will be returned to the calling method and the calling method may then display the string.



PROJECT TO DO LIST:

- ☐ Develop the UML diagram for the project (you must use Astah* to create the UML diagram)
- ☐ Write the algorithm or flowchart for each class method.
- ☐ Implement the Advisee class...hint, do this a little at a time. Do not try to write it all at once and then test it at the end!
- ☐ Test your Advisee class as you create it!

STOP! You will get the information about the driver requirements soon...! ☺

- ☐ Add your driver program, Project4, to the UML diagram for the project
- ☐ Write the algorithm or flowchart for the main method (and any other method you plan to use in the driver program)
- ☐ Implement the driver program
- ☐ Test the driver program

Notes:

- ☐ You do NOT need to print screenshots
- ☐ Print out the source code using Notepad++
 - *Do not print your source code out in regular Notepad*
 - *Be sure to avoid WORD WRAPPING. Print Landscape if you need to.*
- ☐ All output should be neat, readable, well-aligned and have correct spelling/grammar
- ☐ All output will be done using the JOptionPane message boxes
- ☐ Prompts should leave the cursor on the same line when accepting input

CSCI1250 INTRODUCTION TO COMPUTER SCIENCE 1

PROJECT 4 – SPRING 2018

GRADE SHEET

Type your name here

	Grading %	Comments
Is it Correct?		
Code is executable		
Produces correct results		
Produced using minimalistic code		
Results are not contrived		
Is it Maintainable? (Did you follow programming standards?)		
Coded indented correctly		
Comments for each module		
Comments to enhance readability		
Meaningful identifiers used		
Whitespacing used appropriately		
Constants used where appropriate		
Does it have an Appropriate Interface?		
Results are understandable		
Appropriate prompts for requested information		
Appropriate feedback to the user		
Validates user input		
Acceptable grammar used		
Was it submitted properly?		
Zippped folder to D2L containing: (with appropriate file/folder names)		
UML Class diagram		Print and submit electronically; must be completed using Astah*
Algorithm		Print and submit electronically
Source Code		Print and submit electronically (all .java files)
Screen shots	N/A	
Javadocs		Do not print; include electronically only
File folder containing hard copies of above items (submitted in class – must be in order shown above with grade sheet on top)		
Additional Notes / Comments		
Total Grade		