CSC 229

Object Oriented Programming

Assignment 4

Inheritance

Due Date: Saturday 04/06 before midnight

Introduction

In this assignment, you will practice:

- Working with derived classes
- Working with previous concepts, such as methods, arrays, loops, and string methods.

Submission

Submit your assignment on GitHub.

Use the following GitHub Classroom assignment link to accept the assignment and create a GitHub repository:

https://classroom.github.com/a/jVR_9UNM

Resources

- Review Section 7.3 about instantiating the class containing the method main( )
- Review Section 7.8 about Java documentation for classes
- Review Section 8.9 about Java documentation for methods
- Review Section 9.1 about derived classes and inheritance.
- Review Section 9.3 about method overriding and accessing of parent methods with the **`super`** keyword.
- Review Section 9.4 about polymorphism and derived/base class conversion
- Review Section 9.5 about creating class diagrams

Provided Code

You will be given two java files in the GitHub repository, **`Course.java`** and **`Program.java`**.

Course class

`Course.java` contains a complete class definition for an SCSU course, called **`Course`**. The Course class contains:

- Four private fields:
  - **`code`**: String field for the course code. The course code is presented in the form of the three-letter subject code and the course number, separated by a hyphen. Ex: “CSC-229”.
  - **`name`**: String field for the course name. Ex: “Object-Oriented Programming”.
  - **`subject`**: String field for the course subject. Ex: “Computer Science”

    - **`name`**: Integer field for the number of credits the course is worth.

- An argument constructor, which takes four arguments for each of the four private fields and initializes each of them, respectively.
- Four accessor(getter) methods for each of the private fields.
- A **`displayResults`** method, which outputs information for the course when called.

Program class

`Program.java` contains the main class for the program, which has:

- A field for an array of Course objects, called **`courses`**.
- An argument constructor, which takes an integer array size and initializes the **`courses`** field to a blank array of Course objects of the size given as an argument to the method.
- An empty method called **`addCourses`**, which takes a 2D array of strings as a parameter.
- The main method. In the main method, you are given code that:
    - Instantiates an object of the Program class
    - Creates a 2D array of strings representing SCSU courses stored in a variable **`courseInfo`**.
    - Calls the **`addCourses`** method, passing the **`courseInfo`** array.

Each sub-array in the **`courseInfo`** array represents an SCSU course, with information about each course given in the following format:

```
{<Subject Code>, <Course Number>, <Name>, <Subject>, <Credit Count>}
```

For example, one of the subarrays looks like:

```
{"CSC", "229", "Object-Oriented Programming", "Computer Science", "3"}
```

Every course subarray in **`courseInfo`** is in this format.

Prompt

Your task is to develop complete developing the given program that creates object for the courses then print the information about each course. To do so, follow the following instructions:

1. Start by cloning the GitHub repository. Notice that the local repository (directory) qill have the two mentioned Java files.
2. Create two new subclasses of the **`Course`** class, called **`LabCourse`** and **`WritingCourse`**:
    - Both subclasses should contain an argument constructor that takes the same arguments as the argument constructor on the **`Course`** class.
    - Within the constructor on each of these subclasses, call the constructor of the **`Course`** base class using the **`super`** keyword.
    - Both subclasses should override the **`displayResults`** method.
    - Within the **`displayResults`** method on each of these subclasses:

    - The same information output by the **`Course`** class should be output.
    - For the **`LabCourse`** class, the **`displayResults`** method should additionally output that the course has a lab component and that additional fees may apply.
    - For the **`WritingCourse`** class, the **`displayResults`** method should additionally output that the course has is a writing course and that it will satisfy W-course requirements.

3. In the **`addCourses`** method of the main class, create an object for each of the courses in the **`courseInfo`** array. For each course:
    - If the course has a course number that ends in "W", then the course is a writing course, so create a **`WritingCourse`** object for the course.
    - If the course is worth 4 credits, then the course has a lab component, so create a **`LabCourse`** object for the course.
    - If neither of the above are true, then create an object for the course using the **`Course`** class.
    - Add all created objects to the **`courses`** array defined on the main class.
4. Create a new method on the main class called **`displayCourses`**.
    - The method should not take any arguments, nor should it return anything.
    - Within this method, display the information for all courses in the **`courses`** field.
    - Call this method at the end of the main method.

    See sample outputs for more information.

5. Document all classes and methods (**the ones you write** and **the ones given to you**) by following the Java documentation guidelines described in sections 7.8 and 8.9.
6. Submit a UML Class Diagram. Include it as a pdf file or an image file in the same repository.

Notes:

- **DO NOT** modify the **`Course.java`** file. Everything outlined in this assignment can and should be accomplished without doing so.
- Notice that the subject code and course number are separate in the **`courseInfo`** array, but the constructor for the **`Course`** class takes only one argument for the course code.

- All the courses given are either writing courses, lab courses, or neither. In other words, There is no course given that is both a Lab and Writing course, so you do not have to account for it.
- There is no extra credit for this assignment.

Sample Output

```
CHE-120: General Chemistry I
Chemistry | 4 Credits
This course has a lab component
Additional laboratory fee may apply

CHE-121: General Chemistry II
Chemistry | 4 Credits
This course has a lab component
Additional laboratory fee may apply

CSC-207: Computer Systems
Computer Science | 4 Credits
This course has a lab component
Additional laboratory fee may apply

CSC-229: Object-Oriented Programming
Computer Science | 3 Credits

CSC-324W: Computer Ethics
Computer Science | 3 Credits
This course is a writing course
This course will satisfy "W-course" requirements

CSC-330: Software Design and Development
Computer Science | 3 Credits

ENG-112: Writing Arguments
English | 3 Credits

ENG-217W: Introduction to Literature
English | 3 Credits
This course is a writing course
This course will satisfy "W-course" requirements
```

```
MAT-125: Applied Business Mathematics
Mathematics | 3 Credits

MAT-150: Calculus I
Mathematics | 4 Credits
This course has a lab component
Additional laboratory fee may apply

MAT-151: Calculus II
Mathematics | 4 Credits
This course has a lab component
Additional laboratory fee may apply

MAT-221: Intermediate Applied Statistics
Mathematics | 4 Credits
This course has a lab component
Additional laboratory fee may apply

PHY-230: Physics for Scientists and Engineers I
Physics | 4 Credits
This course has a lab component
Additional laboratory fee may apply

PHY-231: Physics for Scientists and Engineers II
Physics | 4 Credits
This course has a lab component
Additional laboratory fee may apply
```

Grading and Submission

UML class diagram	10
Creating WritingCourse and LabCourse subclasses	5
Creating subclass Constructors	5
Calling Course class constructor within subclass constructors	5
Overriding the displayResults method	5
Correct functionality of displayResults methods	15
Creation of course objects with proper classes	10
Adding course objects to the Object array	10
Defining displayCourses method	5
Displaying results	10
Class and method documentation	15
References and collaborations File	5
Total	100