CIT 130 Beginning Java Spring 2022

Assignment #10 – Total Points = 100

Assigned Date: 3/28 Due date: 4/5

DO NOT share your answers with anyone. DO NOT collaborate on completing work with anyone. DO NOT use the Internet to search for specific solutions to assignments. DO NOT pay anyone to complete your assignment. **Avoid web sites that offer solutions to assignments. If you copy work from such web sites, keep in mind that other students are also looking at the same information and will therefore submit duplicated work.** Failure to meet these requirements leads to a violation of the academic integrity principles as stated in your syllabus.

Grading Criteria: See the grading criteria information in your course **syllabus addendum (**Click Home and look towards the middle of the page**)**.

Objective: Demonstrate your understanding of Java **Object Oriented Programming (OOP) inheritance concept.** This assignment is based on the material covered in **chapter 11** of your textbook.

Deliverable: Use the file format:

- **Pa10.java** for the source file of your test class.
- **ExamScore.java** for a class that defines an exam.
- **StudentScore.java** for the utility class that inherits from the ExamScore.java class.
- **UML.PDF** for your UML diagram for the **ExamScore and StudentScore** classes
- Compress these files into one ZIP file using the format: **firstNameLastNamecit130_pa10.ZIP**

Documentation: See your course **syllabus addendum** file in Canvas for the required heading documentation and other notes. Make sure you have the header section properly created using tags such as @author, @file, etc. **Methods need to be properly documented**. See the syllabus addendum file, part 2 (e) on proper process to document each method.

Assignment: Write a program to practice with the use of inheritance in Java.

Constraints:

- Do not use break/continue in your loops.
- Do not use the System.exit() method to exit the program.
- All member attributes must be defined as **private.**
- All member methods must be defined as **public**.

Process:

1. Define the utility Class **ExamScore**:

The class should have **2 private** instance fields to store the **name** of an exam and the **score** for an exam. Also provide the following methods:

- A no-arg **constructor** to initialize the instance variables to values of your choice.
- A **Constructor** with 2 parameters to initialize the instance variables to the arguments indicated in the parameter list of the constructor.
- Appropriate methods to **set** the value of each of the private instance fields.
- Appropriate methods to **get** the value of each of the private instance fields.
- A **toString()** method used to display the values of the object using the format: "Exam Name: " followed by the name, followed by " Score: ", followed by the score.

2. Define the utility Class **StudentScore** that inherits from the **ExamScore** and has one **attribute** to store the identification number for the student.

- A no-arg **constructor** to initialize the instance variable to a value of your choice.
- A **Constructor** with 1 parameter to initialize the instance variables to the argument indicated in the parameter list of the constructor.
- Appropriate methods to **set** the value of each of the private instance field.
- Appropriate methods to **get** the value of each of the private instance field.
- A **toString()** method used to display the identification number along with the Exam score information inherited from the **ExamScore** information.

3. Write a main program (**test class**) to test various operations on the objects of the **ExamScore** and **StudentScore** classes. Perform a series of operations to test each of the methods and the constructors. In addition, **show how to define an array of N (ask the user for number of objects to create) StudentScore objects** from the **StudentScore** class and show the functionality of the class with your array. Show how to populate the array with name: FINAL PROJECT, a random value between 1 and 100 to score, and a random number between 100 and 1000 to identification number. Make sure to use loops for processing arrays.

Sample interaction: This is just a simple interaction. Make sure to fully test your code.

Exam Name: Exam 1 Score: 100.0 **<<NOTE: these are my initial values**
Exam Name: Exam 1 Score: 100.0 Identification Number: 1000 **<< NOTE: these are my initial values**
Enter the name of an exam and the corresponding score
midterm exam
88
just updated information for an exam
Exam Name: midterm exam Score: 88.0
Enter the identification number for a student
1234
Enter the name of an exam and the corresponding score for: 1234
practice exam
87
Exam Name: practice exam Score: 87.0 Identification Number: 1234

The array data (of course, our random numbers will be different)
Exam Name: FINAL PROJECT Score: 15.545017796487326 Identification Number: 792
Exam Name: FINAL PROJECT Score: 44.174792489805256 Identification Number: 253
Exam Name: FINAL PROJECT Score: 54.68431229325966 Identification Number: 507
Exam Name: FINAL PROJECT Score: 98.00298560029641 Identification Number: 186
Exam Name: FINAL PROJECT Score: 39.41541087286207 Identification Number: 415