Assignment 1

Object and Instance

Before attempting this assignment, be sure you have completed all the reading assignments, non-graded exercises, discussions, and assignments to date.

Overview

This assignment consists of two classes that will be implemented. A **Weight.java** class that has three private variables, two private methods and four public methods and a **Project.java** class that has three private methods. The goal of this assignment is to perform calculations using **Weight** objects (instances of **Weight** class).

Weight Class

Weight class should have three private variables, two private methods and four public methods.

Variables

Hint: How should/can you access these private variables?

1. A private constant variable called **OUNCES_IN_A_POUND** that defines the number of ounces in a pound (**16**).
2. A private variable called **pounds** with a data type of **integer**.
3. A private variable called **ounces** with a data type of **double**.

Methods

1. A public parameterized constructor, which initializes the private variables.

```
public Weight (int pounds, double ounces){
        // implementation
}
```

2. A private instance method called **toOunces** with a return type of **double**. This method has no parameters and should return the total number of ounces. For full credit, reuse this method across other methods when possible.

3. A private instance method called **normalize** with a return type of **void**. This method has no parameters and should ensure that the number of ounces is less than the number of ounces in a pound. For full credit, reuse this method across other methods when possible.

4. A public instance method called **lessThan** with a return type of **boolean**. This method should accept a **Weight** object as a parameter and determine if the object is greater or less than the initialized values.

```
public boolean lessThan (Weight weight){
      // implementation
}
```

5. A public instance method called **addTo** with a return type of **void**. This method should accept a **Weight** object as a parameter and add the object's weight values to the initialized values.

```
public void addTo (Weight weight){
      // implementation
}
```

The design flexibility is given to the student, and if the student would like to also create another method such as divideBy method, then the student may do so.

6. A public instance method called **toString** with a return type of **String**. This method has no parameters and should have the following format:

```
x pounds and y ounces
```

where *x* is the number of pounds and *y* the number of ounces. Ounces should be displayed with two decimal places.

Project1 Class

Project1 class should have three private methods and one public method.

Methods

1. A private class method named **findMinimum** with a return type of **Weight**. This method should accept three **Weight** objects as parameters and compare each **Weight** object's weight values to find the minimum. The Weight object with the minimum weight value should be returned and then printed using **toString** in the following format:

```
The minimum weight is x pounds and y ounces
```

where *x* is the number of pounds and *y* the number of ounces. Ounces should be displayed with two decimal places.

2. A private class method named **findMaximum** with a return type of **Weight**. This method should accept three **Weight** objects as parameters and compare each **Weight** object's weight

values to find the maximum. The Weight object with the maximum weight value should be returned and then printed using **toString** in the following format:

```
The maximum weight is x pounds and y ounces
```

where *x* is the number of pounds and *y* the number of ounces. Ounces should be displayed with two decimal places.

3. A private class method named **findAverage** with a return type of **Weight**. This method should accept three **Weight** objects as parameters and calculate the average weight value. A new Weight object with the average weight values should be returned and then printed using **toString** in the following format:

```
The average weight is x pounds and y ounces
```

where *x* is the number of pounds and *y* the number of ounces. Ounces should be displayed with two decimal places.

4. A public method named **main** with a return type of **void**. This method should exercise the correct functionality of **findMinimum**, **findMaximum** and **findAverage** by creating three Weight objects using the hardcoded values below:

```
Weight weight1 = new Weight(11, 3);
Weight weight2 = new Weight(7, 20); // Hint: normalize method should be
used to translate into 8 pounds and 4 ounces
Weight weight3 = new Weight(14, 6);
```

Additional Note:
It is indeed true that the that this coding can be done without ever using the pound attribute since the pound/ounces are only making sense when displaying the result; one can just use the ounces attribute to hold the total weight. However, both pound and ounce are kept in the constructor as teachable lessons.

Submission Requirements

Style and Documentation

Make sure your Java program is using the recommended style such as:

- Javadoc comment with your name as author, date, and brief purpose of the program
- Comments for variables and blocks of code to describe major functionality
- Meaningful variable names and prompts
- Class names are written in upper CamelCase
- Constants are written in All Capitals
- Use proper spacing and empty lines to make code human readable

Deliverables

The submission requires uploading a single zip file, which will contain:

1. The source code of Weight.java (do not include the class file)
2. The source code of Project1.java (do not include the class file)
3. A single document (DOC/DOCX or PDF), which includes the test case(s) in a tabular format **and** screenshot(s) of the test runs for the test case(s). Do not submit the screenshot(s) separately as they should be included in this document. Below is an example of a test case table to follow:

Test #	Input Values (Values as Input)	Expected Output (What is Expected as Result)	Actual Output (Output obtained by code execution)	Pass or Fail (If test passed or failed)
Test 1	Weight weight1 = new Weight(11, 3) Weight weight2 = new Weight(7, 20) Weight weight3 = new Weight(14, 6)	Created weight1 with 11 pounds and 3 ounces Created weight2 with 8 pounds and 4 ounces Created weight3 with 14 pounds and 6 ounces		
Screenshot(s) of the Output (Result obtained by executing the code)	Include the screenshot(s) here			

Grading Rubric

Criteria	Level 3	Level 2	Level 1
Weight class	(31-40 points) Implemented the specifications and logic components	(21-30 points) Not implemented some of the logic and specification components	(0-20 points) Not implemented most of the logic and specification components
Project1 class	(31-40 points) Implemented the specifications and logic components	(21-30 points) Not implemented some of the logic and specification components	(0-20 points) Not implemented most of the logic and specification components
Test cases	(11-20 points)	(6 - 10 points)	(0 – 5 points)

	Included the test cases with the screen shot(s) using the tabular format	Not included some of the test case using the screen shot(s) using the tabular format	Not included most of the test cases with the screen shot(s)