ASSIGNMENT 1

COMP-202, Winter 2017, All Sections

Due: January 25^{th}, 2017 (23:59)

Please read the entire PDF before starting. You must do this assignment individually.

Question 1:	50 points
Question 2:	50 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

To get full marks, you must:

- Follow all directions below
- Make sure that your code compiles
  - Non-compiling code will receive a very low mark
- Write your name and student name is written as a comment in all .java files you hand in
- Indent your code properly
- Name your variables appropriately
  - The purpose of each variable should be obvious from the name
- Comment your work
  - A comment every line is not needed, but there should be enough comments to fully understand your program

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Create a file called `HelloWorld.java`, and in this file, declare a class called `HelloWorld`. This class should define only one method called `main()`. In the body of this method, use `System.out.println()` to display "`Hello world!`". You can find such a class in the lecture slides; make sure you can compile and run it properly.

Warm-up Question 2 (0 points)

Create a file called `Diagram.java`, and in this file, declare a class called `Diagram`. This class should define only one method called `main()`. In the body of this method, use five statements of `System.out.println()` to display the following pattern:

```
  22
2   2
   2
 2
22222
```

Use Strings composed out of the space character and the character '2'. For extra practice, try to draw the pattern '202'.

Warm-up Question 3 (0 points)

Practice with Binary:

We usually use base 10 in our daily lives, because we have ten fingers. When operating in base 10, numbers have a `one`s column, a `ten`s column, a `100`s column, etc. These are all the powers of 10.

There is nothing special about 10 though. This can in fact be done with any number. In base 2, we have each column representing (from right to left) 1,2,4,8,16,etc. In base 3, it would be 1,3,9,27, etc.

Answer the following short questions about number representation and counting.

1. In base 10, what is the largest digit that you can put in each column? What about base 2? Base 3? Base n?
2. Represent the number thirteen in base 5.
3. Represent the number thirteen in base 2.
4. What binary number is equal to the sum of these two binary numbers? 10101011 + 10010001
5. What is the number from the previous part in base 10?
6. What is the binary number for 11010010 + 11000101?
7. And what is the number from the previous part in base 10?

Warm-up Question 4 (0 points)

Logic

1. What does the following logical expression evaluate to?

   ```
   (false or false) and (true and (not false))
   ```

2. Let a and b be boolean variables. Is it possible to set values for a and b to have the following expression evaluate as *false*?

   ```
   b or (((not a) or (not a)) or (a or (not b)))
   ```

Part 2

The questions in this part of the assignment will be graded.

Question 1: Calculator Program (50 points)

Attached to this assignment is a file called `Calculator.java`. Note that there is a marked section where your code must go. The code outside of this area must not be modified.

Write Java code in the marked area to print the following calculations:

1. The sum of the a and b variables
2. The product of the a and b variables
3. The result of dividing a by b
4. The result of dividing a by c
5. A statement saying whether a is larger than b
6. A statement saying whether a an odd number. Use the mod operator $\%$.

For example, if the numbers entered are *5*, *5*, and *1*, the output should be:

```
Sum of a and b:  10
Product of a and b:  25
Dividing a by b:  1
Dividing a by c:  5.0
Is a larger than b:  false
Is a odd:  true
```

Be sure to include the specified text on each output line. That is, concatenate a String literal with the value of a variable.

Note that this program is run by providing *input arguments*. For example, once this program is compiled, it can be run by typing the text `run Calculator 5 5 1` in the Dr. Java *Interactions Pane* and pressing Enter.

Question 2: Creating a Grading Program (50 points)

The goal of this question is to write several *methods* to create a program for outputting student grades. All the code for this question must be placed in a file named `GradingProgram.java`. Note that this means the class must also be named `GradingProgram`.

2a)Void Method for Confirming Entry

Write a method `printInput` that takes as input three `double` arguments and **prints** these numbers. You must include all three numbers as part of a message, separated by commas. For example, your message could be "You entered 34.0, -12.2, and 4.0".

Note that for full marks, this message **must** be written on one line. Research the `+` operator for Strings, or the `System.out.print()` and `System.out.println()` statements.

Hint: To test your method, create a main method. The main method will not be graded in GradingProgram.java, but without it, you won't know whether or not your method works! Your main method should call this `printInput` method and verify the results. You should think of other cases to test!

2b)Methods for Calculations

To make the GradingProgram more interesting, we will write a division method and a maximum method.

Write a method `divide` inside of `GradingProgram.java` that takes as input two `double` values. This method should return the result of dividing the first method parameter by the second. The return value must be a `double` value.

Note that division doesn't work if the second parameter is zero. Therefore, if the second parameter is zero, the method should print an appropriate error message, and then return zero. Your method must **not** print anything if the second parameter is not zero.

To test your `divide` method, you will need to call it from your main method. Think about how you can call the method and then display the answer.

As well, write another method `getMax` that takes two `doubles` values as input. `getMax` must returns the larger of the two input values, and the return type must be double. In the case of a tie, return either value. Don't use the built-in *max* method.

2c)Method Calling

Now we will use the `divide` and `getMax` methods to calculate student grades.

Write a method `finalGrade` that takes as input three `double` values and returns the final percentage out of 100 for a COMP 202 student. The return value must also be a `double`.

- The first value corresponds to the total assignment grade out of 35.
- The second is the midterm mark out of 20.
- And the third is the final exam mark out of 45.

Recall that if the student does better on the final than on the midterm, the mark for the final replaces the mark for the midterm. Therefore, you will have to calculate which is greater:

1. The assignment grade plus the midterm grade plus the final grade divided by 100 OR
2. The assignment grade plus the final grade divided by 80

Note that you must use the `divide` method that you created earlier to divide the sums. Then you must use your `getMax` method to determine which percentage would be higher. Finally, the percentage will then be less than 1, so multiply the answer by 100 before returning it.

For example, a student might have 28 out of 35 for assignments, 18 on 20 for the midterm, and 30 on 45 for the final. The `finalGrade` method takes these numbers `28.0, 18.0, 30.0` as input, in order, and would output a final grade of 76 (to represent 76%).

A student who instead had marks of `28.0, 16.0, 38.0` would have a final grade of 82.5 (to represent 82.5%). In this case, the midterm grade is dropped, because the student's performance in the course is higher using the alternate grading scheme.

What To Submit

You have to submit one zip file with all your files in it to myCourses under Assignment 1. If you do not know how to zip files, please ask any search engine or friends. Google will be your best friend with this, and a lot of different little problems as well.

These files should all be inside your zip. Do not submit any other files, especially .class files.

`Calculator.java`
`GradingProgram.java`
`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise they would not.