CMPSC 111
Introduction to Computer Science I
Spring 2017

Lab 7
Assigned: March 2, 2017
Due: March 9, 2017 by 2:30pm

Objectives

In this laboratory assignment, you will learn more about using the `java.lang.Math` class to perform numerical calculations, further explore the creation of formatted output, learn how to use enumerated types, and practice calling methods in another Java class. Additionally, since real-world software developers often have to debug source code created by other developers and add features to existing code, you will participate in a "bug hunt" and add new source code to an existing system. Ultimately, you will create a working program comprised of two Java classes.

General Guidelines for Labs

This is another team-based assignment. As in the past assignments, you must work in a team of two during the laboratory session and throughout the coming week. You may select your own partner, as long as you are not working with the same person from a previous assignment. As this is the another challenging team-based assignment, you and your partner should plan your time this week accordingly and work on it incrementally. Please make sure that you use Slack and your team's Git repository to collaborate effectively. Also, you and your partner should remember to work on the Alden Hall computers and to update your Git repository on a regular basis. Before starting this assignment you and your partner should carefully review the Honor Code policy.

Reading Assignment

After reviewing all of the assignment sheets for the past laboratory and practical sessions and the course slides and notes, you should review Sections 3.5 through 3.8 of your textbook. To enhance your understanding of some points in this lab you should additionally examine Figures 4.7 and 4.8

Participating in a "Bug Hunt"

After changing into the `cs111S2017-share/` directory, which contains our course's version control repository, you should type the command "`git pull`" to download the source code for this assignment. Now, move the code into the `labs/lab7/` directory in your team's repository and use `gvim` to study the source code of the `CommandLineGeometer.java` and `GeometricCalculator.java` files. What methods do these classes provide? How do they work? Does any of this code look incorrect?

After carefully reviewing the source code and PP 3.6, 3.7, and 3.9 on page 158 of your textbook and then compiling and running the `CommandLineGeometer` class, you should notice that there are several defects in this program. As such, you will need to take part in a "bug hunt" to find and fix all of the problems! First, you should find the method responsible for calculating the volume of a sphere. Using the equation in PP 3.6 as a reference point, what is the defect in this method?

The `GeometricCalculator` also provides a method to calculate a triangle's area. Once again, there is a mistake in this method. Can you find and fix it? How did you know that this was the bug? If you investigate the source code of the method for calculating the volume of a cylinder, you will notice that there is another defect lurking in the source code. Wait! If you carefully study the way in which the `CommandLineGeometer` calls the method provided by the `GeometricCalculator` and then displays the resulting output, you will realize that there is another bug in this program. Make sure you have found all of the problems before continuing with this laboratory assignment.

Extending the Geometry Calculator

After reviewing the aforementioned programming projects on page 158 of your textbook, you will also notice that the `GeometricCalculator.java` does not contain methods for calculating the surface area of a sphere or a cylinder. While avoiding the types of mistakes that you corrected in the previous phase of this assignment, please add in new methods to perform these calculations. In addition, you will need to add appropriate input and output statements and method calls to the `CommandLineGeometer` to ensure that the entire program works correctly. For instance, you will need to implement a new method called `calculateSphereSurfaceArea` to the `GeometricCalculator.java` file and then add input and output code to the `CommandLineGeometer.java` file. Whenever possible, try to follow the correct pattern established in the given source code. Please see the course instructor if you have questions!

As you continue to critically review the source code of the `CommandLineGeometer`, you will notice that it does not always consistently produce output for the user. For example, even though it displays the user-input radius before calling `calculateSphereVolume`, it does not appropriately display the sides of the triangle for the `computeTriangleArea` method—can you please add in this feature? Moreover, none of the output of the `double` variables in the `CommandLineGeometer` is formatted in a consistent fashion. To solve this problem, you should use one of the techniques described in Section 3.6 of your textbook to format the output of all decimals to contain four decimal points. For instance, you may consider creating an instance of the `DecimalFormat` class.

Finally, please notice that none of the provided code is commented. As part of this assignment, you should add detailed comments to your code, following the textbook's standard. Please see the instructor if you are not sure how or where to add comments to your program. Don't forget that you and your partner should evenly divide up the work needed to complete this assignment.

Exploring Features of Java

The `CommandLineGeometer` program uses an enumerated type, as described in Section 3.7, to store specific values in a variable. Intuitively, an enumerated type allows a variable to take on one of a pre-specified set of values or levels. In this case, the `GeometricShape` enumerated type can take on three possible values. What are they? Why is it useful to declare and use this type of variable?

You will notice that this laboratory assignment organizes the methods into two separate classes, as you have seen in past assignments and in-class exercises. In particular, the `CommandLineGeometer` provides the user interface for our program and the `GeometricCalculator` furnishes the methods that perform the required computations. If you want to make changes to the way in which the program accepts input or produces output, then you will need to modify the `CommandLineGeometer`. Otherwise, if you want to modify the way in which the program performs a computation, or add a

new computation, then you must make changes to the `GeometricCalculator`. Overall, these two Java classes complete their work by following a pattern similar to that which is outlined in Figures 4.7 and 4.8 of your textbook. Please see the instructor if you have questions about this approach.

Additionally, it is important to note that this assignment asks you to add new methods to the `GeometricCalculator.java` file. To complete this task, you should directly copy the pattern that you see in the provided methods, only making changes to implement the new functionality. Also, these methods accept parameters, of type `double`, that are passed from the `main` method in the `CommandLineGeometer` to one of the "calculate" methods in the `GeometricCalculator`. You should also notice that all of the methods return a `double` variable to the method that calls it. Intuitively, the parameters are the "input" to a method and the return values are the "output" of the method. When you create the required new methods, you should follow the pattern of the previously implemented method, ensuring that you have the same types of input and output.

Required Deliverables

This assignment invites you to submit electronic versions of the following deliverables through the Bitbucket repository that your team created. As you complete this step, you should make sure that you created a `lab7/` directory within your team's repository. Then, you can save all of the required deliverables in the `lab7/` directory—please see the course instructor or a teaching assistant if you are not able to create your repository properly, following the steps from previous assignments. Students also should submit signed and printed versions of all the required deliverables.

1. A completed, properly commented and formatted `CommandLineGeometer` class.
2. A completed, properly commented and formatted `GeometricCalculator` class.
3. The output from running `CommandLineGeometer` in the terminal. You may use `gvim` to save your output as follows: using the mouse, select everything from the "`java CommandLineGeometer`" command to the end of your output. Right-click on the selected text and copy it. Type "`gvim output`"—note that this *not* a Java program!—and use the "Edit/Paste" menu item to paste the output into the file. Now, use "`:w`" or the "File/Save" menu item to save this file.
4. A written reflection on the challenges that you faced during the completion of this laboratory assignment. Again using the `gvim` text editor, you can input your reflections into a file called "`reflection`" and then save this file in your Bitbucket team's repository.
5. A detailed listing of the tasks that each team member completed for this assignment.

Share your program and the output file with me through your team's Git repository by correctly using the "`git add`", "`git commit`", and "`git push`" commands. When you are done, please ensure that the Bitbucket web site has a `lab7/` directory in your repository with the five files called `CommandLineGeometer.java`, `GeometricCalculator.java`, `output`, `reflection`, and `tasks`.

In adherence to the Honor Code, students should complete this assignment on an per-team basis. While it is appropriate for students in this class to have high-level conversations about the assignment, it is necessary to distinguish carefully between the team that discusses the principles underlying a problem with others and the team that produces assignments that are identical to, or merely variations on, someone else's work. With the exception of the source code that was provided by the course instructor through the Git repository, deliverables that are nearly identical to the work of others will be taken as evidence of violating Allegheny College's Honor Code.