Lab1 - Filling the Fountain

Reminder: You must first have your completed lab checked off by your TA (either in-person or by e-mail). Afterwards, you must submit your lab via Brightspace. Failure to do either will result in a grade of 0.

Goals

In this Lab you will:

1. Explore and write a basic Java program that, ultimately, uses a few variables to do some math.
2. As part of that program, request input from a user, and deal with invalid input.
3. Test that your program works correctly.

Resources

You will want to refer to each and all of the following to complete this Lab:

1. **Lab Overview** (this document).
2. **FountainLab.java**, the Starter Code for this Lab (attached).
3. **Sample Exchange**, showing a sample input and output exchange between the program and a user running it on the command line, once the program is correctly written (attached).

Background

In a fictional setup, Felix's Fountains (LLC) needs your help writing a simple command line program their designers can use to see how much water will be needed to fill a given fountain design. Felix's Fountains are known for their elegant creations: a simple circular basin, with a statue at the middle, always resting on a rectangular base.

Their fountains look similar to the following:

Image courtesy of Rusty Clark, `https://www.flickr.com/photos/rusty_clark/15697500752`

Felix's Fountains has hundreds of different possible designs, each using slightly different shapes for both the statue and the pool - we can't just solve the problem for a particular fountain. For the purposes of this Lab, you can ignore water that might be in the statue (for instance, the bowl of water, above), or water that might be shooting into the air or falling down, etc. We are only concerned about the basic pool the statue rests in, and how much water it will need. It's not quite as simple as a regular circular pool - we have to account for the space displaced by that rectangular base under the statue.

Specifically, you can imagine the pool as a cylinder of water (green below), with a rectangular prism of water (red below) removed from its center and replaced with stone instead. We're interested in calculating the total volume of water in the pool, given four variables:

1. **`radius`**, the (inner) radius of the pool, in feet.
2. **`width1`**, the x-dimension width of the statue's rectangular base, in feet.
3. **`width2`**, the y-dimension width of the statue's rectangular base, in feet.
4. **`depth`**, the depth of the water when the pool is filled to capacity, in feet.

What To Do, And How

You will write a single file Java program in a file called **`FountainLab.java`** (feel free to begin with the attached Starter Code later in this handout), that asks the user to input each of the four variables **`radius`**, **`width1`**, **`width2`**, and **`depth`**, discussed in the **Background** section, and then reports the following statistics:

1. The total volume of water needed, in cubic feet.
2. The total volume of water needed, in gallons.
3. The amount of money, in USD, that water will cost to procure.

Calculating Output Statistics: Math Formulae, Conversions, And Tips

The volume of a cylinder with radius r and height h is:

$$Volume = r^2 * \pi * h$$

Note that the variable `Math.PI` can be used in a Java program for pi above.

The volume of a rectangular prism with x-axis width w, y-axis width h, and depth d, is:

$$Volume = w * h * d$$

1 cubic foot of water is 7.481 gallons of water.

1 gallon of water will cost Felix's Fountains 10 cents to provide. Gallons of water are sold in whole units, so for example if you needed just a smidge more than 10 gallons, it will still cost the full $1.10. You will want to make use of `Math.ceil(...)` to round a double value up to the next largest integer.

You will want to use `NumberFormat.getCurrencyInstance().format(...)` to print the cost, in dollars. You can see an example of doing this in the Starter Code.

Asking For, and Getting Input

For each of the four variables, you must prompt the user to input a value from the command line. However, do not simply trust the user to give valid input; you must handle the user trying to type

invalid numbers into the program. The Starter Code gives an example of how you could verify and keep asking the user for input; you will need to modify this to also tell the user there was an error if they input something incorrect.

Each variable should be a number with a decimal point somewhere in the following ranges:

1. **`radius`** should be between 10.0 and 15.0, inclusive.
2. **`width1`** should be between 2.0 and 8.0, inclusive.
3. **`width2`** should be between 2.0 and 8.0, inclusive.
4. **`depth`** should be between 1.0 and 3.0, inclusive.

Additionally, an **Example Exchange** with the user is a Resource for this Lab, and is attached. Model your prompts and warnings off of the text in this exchange.

Testing your Program

It is insufficient to simply write your code; you also need to confirm your program works by running it, and continuing to edit it until it performs correctly. Specifically, you should be able to replay the inputs shown in the **Example Exchange** and see your program run exactly the same way.

```
import java.util.*;
import java.text.NumberFormat;

// A sample input/output exchange for a correctly completed version of
// this program is listed as a separate attachment.
public class FountainLab
{
    public static void main(String args[])
    {
        // Create a Scanner to read from user input
        Scanner in = new Scanner(System.in);

        // TODO - delete this section - it simply shows you how you might
        // keep asking the user for an input.
        double ignoredVariable = 0.0;
        while(ignoredVariable < 3.0 || ignoredVariable > 7.0)
        {
            System.out.print("Enter a number between 3.0 and 7.0: ");
            ignoredVariable = in.nextDouble();
        }

        // TODO - ask for each of `radius`, `width1`, `width2`, `depth`.
        // Remember to keep asking if the value is out of range, and to
        // tell the user that they've entered an invalid value.

        // TODO - print the calculated statistics specified in
        // "What To Do, And How"

        // You can print dollar amounts with the following code.  Note that
        // if you just print a double, it may not print two decimal places
        // worth of pennies  (e.g. "$8.80" would appear as "$8.8").
        double cost = 0.0;
        System.out.println("That amount of water will cost: " +
                           NumberFormat.getCurrencyInstance().format(cost));

    }
}
```

This is an example of running the program, including the text that prints out when using `System.out.print` and `System.out.println` and the actual text the user typed into the program (formatted **`like this`** below):

```
Please enter the radius of the fountain (10.0 to 15.0 feet):  1.0
(INCORRECT VALUE DETECTED. Please enter a value between 10.0 and 15.0)
Please enter the radius of the fountain (10.0 to 15.0 feet):  10.0
Please enter the rectangular pedestal's x-axis width (2.0 to 8.0 feet):  10.0
(INCORRECT VALUE DETECTED. Please enter a value between 2.0 and 8.0)
Please enter the rectangular pedestal's x-axis width (2.0 to 8.0 feet):  2.0
Please enter the rectangular pedestal's y-axis width (2.0 to 8.0 feet):  0
(INCORRECT VALUE DETECTED. Please enter a value between 2.0 and 8.0)
Please enter the rectangular pedestal's y-axis width (2.0 to 8.0 feet):  2.0
Please enter the water's depth when full (1.0 to 3.0 feet):  -1
(INCORRECT VALUE DETECTED. Please enter a value between 1.0 and 3.0)
Please enter the water's depth when full (1.0 to 3.0 feet):  1.0
Water Volume in cubic feet:  310.1592653589793
Water Volume in gallons:  2320.301464150524
That amount of water will cost:  $232.10
```