

CSc 110 Assignment 2:

Introduction to Loops, parameters and return values

How to hand it in:

Submit a file named **Loopy.java** through the Assignment 1 link on the CSC110 [conneX](#) site.

Learning Outcomes:

When you have completed this assignment, you should understand:

- How to use a for loop to repeat a statement a specified number of times.
- The effect of escape sequences on printed strings.
- How to write static methods that take parameters.
- How to write static methods that return a value and how to use that value in the calling method.
- The flow of control (i.e. the effects of method calls and assignment statements, parameters passed in and return values).

Part 1 – More Ascii Art

Write a set of methods that will output an ascii image as shown below. Your implementation must include the following static methods:

```
public static void drawShape(int size)
public static void drawImage(int largestShape)
```

DrawShape

Your `drawShape` method will print a set of stars bordered by `/` and `\` characters. The `size` parameter will dictate how many star pairs will be printed.

To receive full marks, you **must** use for loops to print the correct number of `*` `/` and `\` characters. You will need to use a combination of `System.out.print` and `System.out.println` statements – make sure you know that difference.

Examples:

Method Call	Output
<code>drawShape(1)</code>	<code>//\</code> <code>/**/</code> <code>\\//</code>
<code>drawShape(2)</code>	<code>///\</code> <code>/****/</code> <code>\\\\//</code>
<code>drawShape(3)</code>	<code>////\</code> <code>/*****</code> <code>\\\\\\//</code>

DrawImage

Your `drawImage` method will print a sequence of shapes getting increasingly bigger and then decreasingly smaller. The `largestShape` parameter will dictate the size of the largest shape to be printed and subsequently, how many shapes will be printed.

To receive full marks, you **must** use the parameter passed with for-loops to print the correct number and correct size of shapes. You will need to call your `drawShape` method.

Examples:

Method Call	Output
<code>drawImage(1)</code>	<pre>//\\ /**/ \\// //\\ /**/ \\//</pre>
<code>drawImage(2)</code>	<pre>//\\ /**/ \\// ///\\\ /****/ \\\\// ///\\\ /****/ \\\\// //\\ /**/ \\//</pre>
<code>drawImage(3)</code>	<pre>//\\ /**/ \\// ///\\\ /****/ \\\\// ///\\\ /*****/ \\\\// ///\\\ /****/ \\\\// //\\ /**/ \\//</pre>

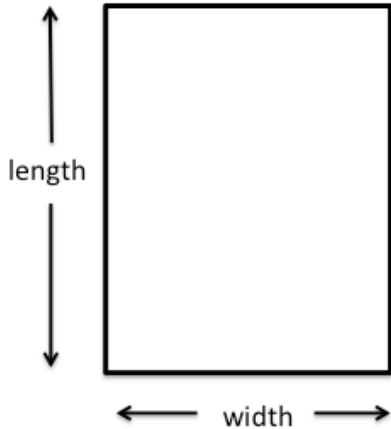
Breaking down the steps

1. Create your Loopy class
 - a. Write the code for your class in your chosen editor with an empty main method.
 - b. Save this file as Loopy.java
 - c. Compile this file in your terminal
 - i. `javac Loopy.java`
2. Write the method `public static void drawShape(int size)`
 - a. Write the `drawShape` method
 - b. Save
 - c. Compile
 - i. `javac Loopy.java`
 - d. Repeat steps a through c until your output matches that described above for the `drawShape` method
3. Write the method `public static void drawImage(int numShapes)`
 - a. Write the `drawImage` method
 - b. Save
 - c. Compile
 - i. `javac Loopy.java`
 - d. Repeat steps a through c until your output matches that described above for the `drawImage` method

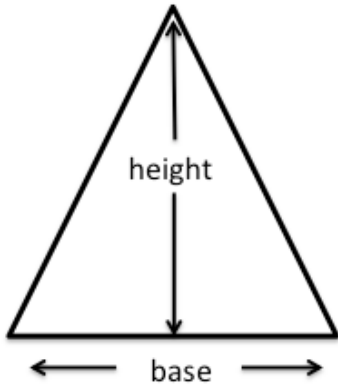
Part II – Math Calculations

Write a set of methods that will, given some parameter input, perform calculations and return a result. Your program must include the following static methods. An image is provided below each method to clarify parameters and calculations to be performed. For your method implementations, wherever possible, you should call the methods you have already written to calculate intermediate results.

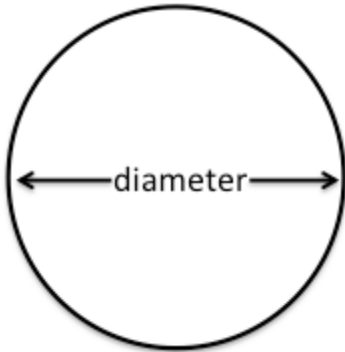
```
public static double rectangleArea(double length, double width)
```



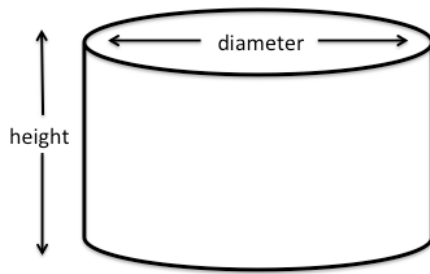
```
public static double triangleArea(double height, double base)
```



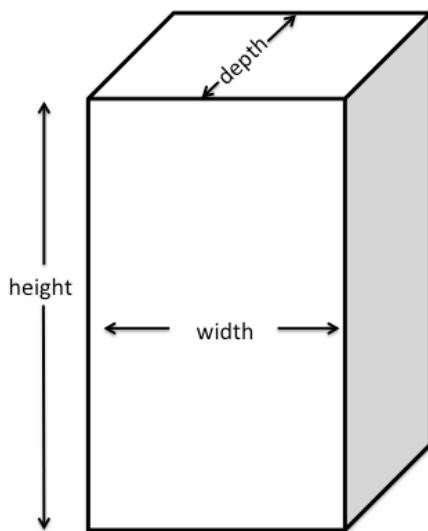
```
public static double circleArea(double diameter)  
public static double circumferenceCircle(double diameter)
```



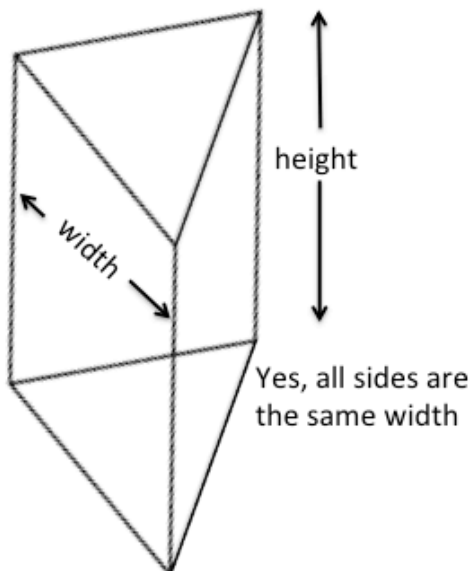
```
public static double cylinderSurfaceArea(double height, double diameter)
```



```
public static double rectPrismSurfaceArea(double height,  
double depth, double width)
```



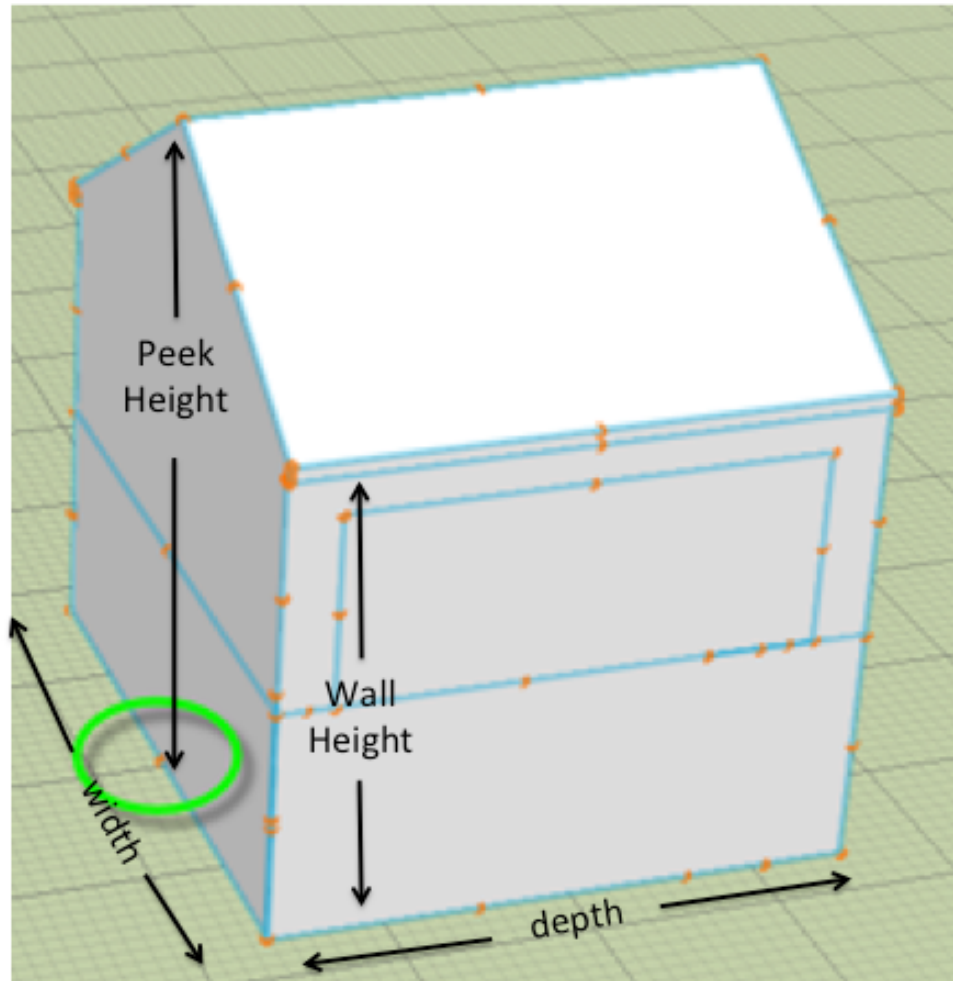
```
public static double triPrismSurfaceArea(double height,  
double width)
```



```
public static double houseSurfaceArea(double wallHeight,  
double peekHeight, double width, double depth)
```

You are a builder and you will be ordering shingles to cover the whole house (roof and siding). You need an approximate surface area. You are ignoring the windows and the roof overhang and assuming the house follows the general dimensions shown below.

For your method implementation, you should use the methods you have already written to calculate the intermediate results and finally return the full surface area of the house.

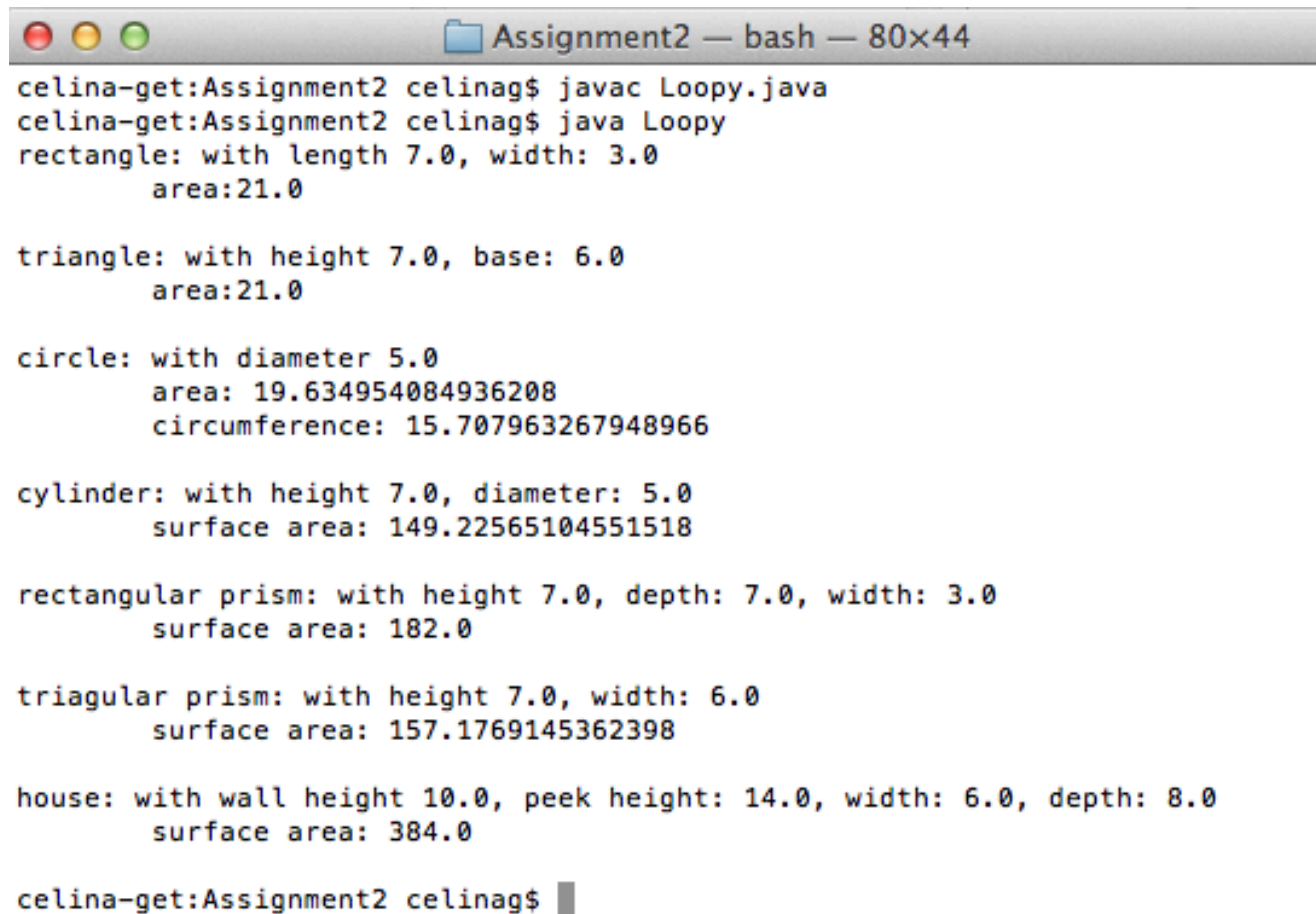


Breaking down the steps

1. Write each method one by one following this approach
 - a. Write the method
 - b. Save
 - c. Compile
 - d. Repeat a-c until it compiles
 - e. Call the method from your main and print out the result to the console
 - f. Repeat a-e until the console output is what you expect.

Sample run of test cases

NOTE: This is only demonstrating one test case for each method. Be sure to test your implementation thoroughly so you can pass all possible test cases.



```
celina-get:Assignment2 celinag$ javac Loopy.java
celina-get:Assignment2 celinag$ java Loopy
rectangle: with length 7.0, width: 3.0
          area:21.0

triangle: with height 7.0, base: 6.0
          area:21.0

circle: with diameter 5.0
        area: 19.634954084936208
        circumference: 15.707963267948966

cylinder: with height 7.0, diameter: 5.0
          surface area: 149.22565104551518

rectangular prism: with height 7.0, depth: 7.0, width: 3.0
                   surface area: 182.0

triangular prism: with height 7.0, width: 6.0
                  surface area: 157.1769145362398

house: with wall height 10.0, peek height: 14.0, width: 6.0, depth: 8.0
       surface area: 384.0

celina-get:Assignment2 celinag$
```

Grading:

Marks will be allocated for the following...

- Your code must compile and run.
- Your code should produce the output exactly as requested.
- Your code must be indented and documented appropriately. Please follow the guidelines in [Style_Guidelines.pdf](#), available in the Resources folder on connex.