

# 1 Before Getting Started on the Exercises

The exercises in this lab are similar to the examples we have been looking at in class. They are of the form

Input  $\Rightarrow$  Do Calculations  $\Rightarrow$  Output

like the Sphere Properties example we discussed in class, and is below. Make sure you understand what each line is doing. Also, review the data types we discussed in class, what they hold, and when you should use each type.

```
1 import java.util.Scanner;
2
3 public class SphereProperties {
4
5     /*-
6      * This will find the surface area and volume of a sphere.
7      * Don Spickler
8      * 9/1/2016
9      */
10
11     public static void main(String[] args) {
12         Scanner keyboard = new Scanner(System.in);
13         System.out.print("Input the radius of the sphere: ");
14         double radius = keyboard.nextDouble();
15         double volume = 4.0 / 3.0 * Math.PI * Math.pow(radius, 3);
16         double surfacearea = 4 * Math.PI * Math.pow(radius, 2);
17
18         System.out.print("The Surface Area is: ");
19         System.out.println(surfacearea);
20         System.out.print("The Volume is: ");
21         System.out.println(volume);
22     }
23 }
```

## 2 The printf Statement

From time to time you will want to have a little more control over the output of a program than the simple print and println commands will give. For example, say you are writing a program that displays amounts of money. You would not want the output to have 10 decimal places since you only need two. In this case, having an output of \$123.4567890 would be silly. You would also not want an output of \$123.4, having \$123.40 would look much better.

Java has a nice function for formatted output, the printf function. The syntax for the printf function is a string with “inserts” followed by values that will be inserted. An insert is a special character sequence that starts with % and then a type and format designation.

For doubles (and floats) the inserts have the form %*#*.*#*f where the first number is the total width given for printing the value (including the decimal point) and the second number is the number of places to the right of the decimal point that are used. If the first number is left blank then the value is printed left justified to the previous prints. So the command,

```
System.out.printf("Pi = %10.2f \n", Math.PI)
```

will print out `Pi =` followed by the value of  $\pi$  using 10 spaces and two decimal places. That is,

```
Pi =          3.14
```

Displaying the spaces from the above output, note that there is one space before and after the `=` in the string before the insert. So there are 6 leading spaces along with the 4 places for 3.14. So  $\pi$  was written with 10 total places and two decimal places.

```
Pi_=_3.14
```

and

```
System.out.printf("Pi = %.2f \n", Math.PI);
```

will print out `Pi =` followed by the value of  $\pi$  using two decimal places. That is,

```
Pi = 3.14
```

You can also put multiple inserts into one `printf` statement, you just add more values after the string, as you can see in the last line of the following example. Note that the `\n` at the end is a new line character so it simply includes a return to the next line.

```
1 public class PrintfExample {
2
3     public static void main(String[] args) {
4         System.out.printf("Pi = %10.2f \n", Math.PI);
5         System.out.printf("Pi = %10.4f \n", Math.PI);
6         System.out.printf("Pi = %.2f \n", Math.PI);
7         System.out.printf("Pi = %.7f \n", Math.PI);
8         System.out.printf("Pi = %.20f \n", Math.PI);
9         System.out.printf("Pi = %20f \n", Math.PI);
10        System.out.println();
11        double var1 = 25.6892348;
12        double var2 = 2.718281828459045;
13        System.out.printf("The value of var1 is %.5f and The value of var2 is %15.10f \n", var1,
14                           var2);
15    }
```

## Program Output

```
Pi =          3.14
Pi =          3.1416
Pi = 3.14
Pi = 3.1415927
Pi = 3.14159265358979300000
Pi =          3.141593
```

```
The value of var1 is 25.68923 and The value of var2 is      2.7182818285
```

### 3 Exercises

Paired Programming is a common method used in software engineering, both in education and in industry, where two programmers will pair up and code a program or sections of a more complex program. For the following exercises, pair up with someone else in the class and create a project for each exercise. In your comment section at the top of each of the programs make sure that both of your names are listed in the Author line.

In paired programming both programmers work on one computer together. It does not mean, and defeats the purpose, if you simply divide up the exercises and do them alone. Working on your own can be done with the homework exercises. You will get a lot out of working together on these. After the lab is finished make sure that each group member has the code files you worked on.

For each of the following create a new project with an appropriate name and then write a program that solves the given problem.

Remember:

- Shift+Ctrl+F to format the program, or Shift+Command+F on the Mac.
- The standard comments of at the top,
  - Names of the two authors of the program.
  - Date
  - Description

For each program, you will be submitting the java code file through MyClasses, as you did before. I also want either a Microsoft Word docx file, LibreOffice Writer odt, or a text file (which you can create with NotePad++) which contains the following.

- The answers to the three main questions:
  - WANT:** What do I want as the final result of the program?
  - HOW:** What is the calculation I will need to do?
  - NEED:** What do I need to complete this calculation?
- The answers above reformatted into an algorithm.
- Output of at least three runs of each program on different data inputs.

This docx, odt, or text file is to be uploaded to MyClasses as well. You can copy and paste output from the Eclipse console area to the word or text program. Only one of you will need to submit the files.

1. As you know from elementary geometry you can calculate the area of a triangle by  $\frac{1}{2}bh$  where  $b$  is the length of the base and  $h$  is the length of the height of the triangle. If you do not know the height of the triangle and just know the lengths of the three sides you can use Heron's formula. Heron's formula states that given the lengths of the sides of the triangle ( $a$ ,  $b$ , and  $c$ ) we can calculate its area by

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

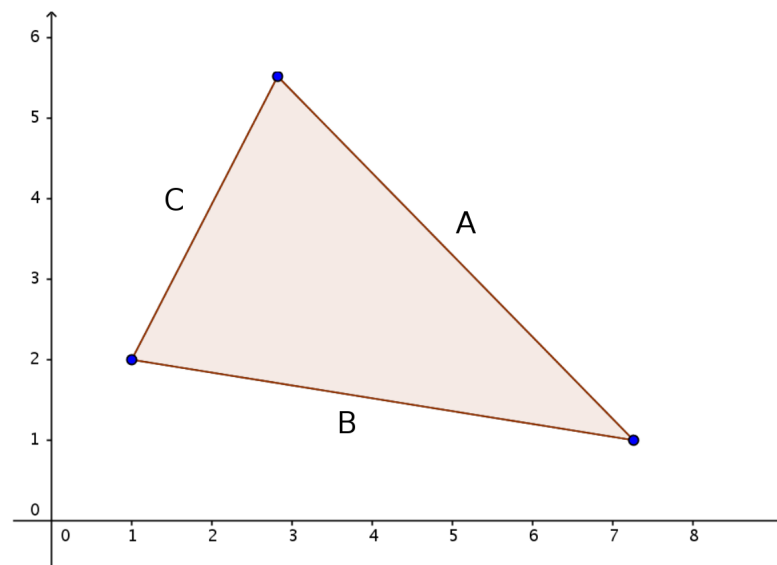
where  $p$  is the semi-perimeter of the triangle  $p = \frac{1}{2}(a + b + c)$ . Note that to take a square root in Java use the command `Math.sqrt(x)`. So the following command will store the square root of the value of the variable  $t$  into the variable  $root$ .

```
double root = Math.sqrt(t);
```

You can put any valid expression inside the square root as well, so for example

```
root = Math.sqrt(15.8 * t + 27.9/y - 4 * z);
```

is a valid assignment statement.



Sample run,

```
Input the length of side A: 4
Input the length of side B: 5
Input the length of side C: 7
The area is = 9.797958971132712
```

2. Write a program that will take as input the total on a bill at a restaurant (before tax) and outputs the bill amount, the tax (assume 6%), the tip which is 15% of the pre-tax

bill, and the total amount the dinner is going to cost you (that is, the bill amount with both tax and tip included).

For example, say that your food costs \$35, then this is your pre-tax bill. The tax will be  $35 \cdot 0.06 = 2.10$ . The tip is off of the pre-tax bill (the \$35 not \$37.10), so it would be  $35 \cdot 0.15 = 5.25$ . So the total amount would be  $35 + 2.10 + 5.25 = 42.35$ . Two sample runs of the program are below. Note that the decimal points line up.

```
What is the bill amount? 35
```

```
Bill      35.00
```

```
Tax       2.10
```

```
Tip       5.25
```

```
Total    42.35
```

```
What is the bill amount? 89.23
```

```
Bill      89.23
```

```
Tax       5.35
```

```
Tip      13.38
```

```
Total   107.97
```