Tutorial 2

COMP 1006/1406 Fall 2017

Objectives

Writing a basic Java program, using methods from the *String* and primitive wrapper classes, using objects.

A) If you did not complete part (d) of Tutorial 1, complete this now. If you did complete this, move on to part (B) of this tutorial.

Write a Java program that queries the user for a number, call is N, and then draws a an NxN grid of Qs. For example, if the user specifies 5 then the output would be

```
QQQQQ
QQQQQ
QQQQQ
QQQQQ
QQQQQ
```

The logic that generates this should have nested loops (i.e., a for loop inside another for loop).

Next, modify your code so that your program displays 4 triangles based on the input N. For example, if the number was 4, then the output should be

```
Q
QQ
QQQ
QQQQ

QQQQ
QQQ
QQ
Q

QQQQ
 QQQ
  QQ
   Q

   Q
  QQ
 QQQ
QQQQ
```

B) Write a program with a method that computes square roots using the *Babylonian* method.

https://en.wikipedia.org/wiki/Methods_of_computing_square_roots#Babylonian_method

It is an iterative method that keeps making better and better approximations to the square root of a number. The

algorithm is terminated when the improvement in successive approximations becomes small.

Our version of the Babylonian method to find the square root of a number *N* is as follows:

1. Let M1 := N/2 be our first guess
2. Let M2 := average of M1 and N/M1
3. If |M1-M2| < epsilon then stop and output M2
4. Otherwise, Let M1 := M2 and go back to step 2

Here, |x| is the absolute value of x and epsilon is a small positive number (like 0.0001). We use := to denote assignment above.

Your class should have a function

```
public static double sqrtBabylonian(double N, double epsilon)
```

Your *main* method should test your method by computing the square root of several numbers (using a small epsilon) and comparing the answer to what Math.sqrt gives.

C) Write a Java program that queries the user for an integer and a decimal number. Use a *Scanner* object for user input using only the **next()** method (which always returns a *String* object).

Convert the strings to an *int* and a *double* using the conversion functions in the associated primitive wrapper classes. Next, add these numbers and then convert the sum to a string using a conversion method in the *String* class.

For help on the conversion methods, see the API for each of the classes.

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html
https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html
https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html

D) Write a program that generates a two-dimensional array of random positive integers (all greater than zero). Once the array is created your program should

1. Print the array nicely (each row should be on its own line).
2. Compute a *greedy* sum in the array as follows: imagine the array as a grid that you can walk through (moving up, down, left or right). Starting at row and column zero, move one step (horizontally or vertically, but not diagonally) to the position with the largest number in it (of all valid possible steps). Repeat this until you reach either the bottom or right side of the grid. As you walk through the grid, add up all elements of the positions you visit. When you visit a position, change its value to zero.
3. When the greedy sum (walk) is completed, display the array again. The zeroes will illustrate the path taken.