```
------------------------------------------------------------------------
ENCE 688R: Solutions to Homework 1                          March 2017
========================================================================

------------------------------------------------------------------------
```

Problem 7.2: Compute and print sum of numbers below 1000 that are either multiples of 3 or 5.

The source code is as follows:

```
/*
 * ================================================================
 * SimpleSummation.java: Compute and print sum of numbers below
 * 1000 that are multiples of 3 and 5.
 *
 * Written by: Mark Austin                                March 2017
 * ================================================================
 */

public class SimpleSummation {
   public static void main( String[] args ) {
      int sum        = 0;
      int upperLimit = 1000;

        for( int i =0 ; i < upperLimit; i = i + 1 ) {
            if( i % 3 ==0 || i % 5 == 0){
                sum = sum + i;
            }
        }

        System.out.printf("--- Summation = %d ... \n", sum );
    }
}
```

And is the program output:

```
prompt >>
prompt >> java SimpleSummation
--- Summation = 233168 ...
prompt >>

------------------------------------------------------------------------
```

Problem 7.6: This program prompts a user for three lengths of a triangle, checks to see that the lengths define a valid triangle, and then computes and prints the area.

The source code is as follows:

```
/*
 *  ============================================================================
 *  Triangle.java : This Java 2 program prompts a user for three lengths of
 *  a triangle. If the three lengths form a valid triangle, then the area
 *  of the triangle will be computed and printed ...
 *
 *  Written By: Mark Austin                                    November 2005
 *  ============================================================================
 */

import java.lang.Math;
import java.util.*;
import java.io.*;
import java.text.*;

public class Triangle {

    public static void main( String args[] ) {
        float fLength1, fLength2, fLength3;
        float fs, farea, fMax;
        String sLine;

        // Print welcome message.

        System.out.println("Welcome to the Triangle Program");
```

```
    System.out.println("-------------------------------");

    // Prompt user for coefficients in quadratic equation.

    System.out.println("Please Enter lengths of the triangle edges");
    System.out.print("Input Length 1: ");
    sLine    = getTextFromConsole();
    fLength1 = Float.valueOf(sLine).floatValue();

    System.out.print("Input Length 2: ");
    sLine = getTextFromConsole();
    fLength2 = Float.valueOf(sLine).floatValue();

    System.out.print("Input Length 3: ");
    sLine = getTextFromConsole();
    fLength3 = Float.valueOf(sLine).floatValue();

    // Print details of the triangle edges .....

    System.out.println("The edge lengths you have entered are: ");
    System.out.println("Length 1 = " + fLength1 );
    System.out.println("Length 2 = " + fLength2 );
    System.out.println("Length 3 = " + fLength3 );

    // Check that edge lengths are greater than zero ...

    if ( fLength1 < 0.0 || fLength2 < 0.0 || fLength3 < 0.0 ) {
         System.out.println("ERROR: Edge lengths must be positive!");
         return;
    }

    // Check that the lengths form a valid triangle ...

    fMax = (float) Math.max( Math.max(fLength1,fLength2), fLength3);
    if ( 2.0*fMax - fLength1 - fLength2 - fLength3 > 0.0 ) {
         System.out.println("ERROR: Edge lengths do not form a valid triangle!");
         return;
    }

    // Ccmpute and print area of the triangle

    fs    = (float) (fLength1 + fLength2 + fLength3)/2;
    farea = (float) Math.sqrt( fs*(fs-fLength1)*(fs-fLength2)*(fs-fLength3) );

    System.out.printf("Area = %8.3f\n", farea );

}

/*
 *  ============================================================================
 *  Method getTextFromConsole(): Read line of text from console (keyboard input)....
 *
 *  Input  : None.
 *  Output : String sLine -- character string of keyboard input
 *  ============================================================================
 */

public static String getTextFromConsole() {
   String inLine = "";

   // Create buffered reader for keyboard input stream....

   BufferedReader inStream = new BufferedReader (
                             new InputStreamReader(System.in));

   // Try to read input from keyboard ....

   try {
      inLine = inStream.readLine();
   } catch (IOException e) {
      System.out.println("IOException: " + e);
   }
```

```
       return inLine;
    }
}
```

This code is a bit dated. A more modern (2017 vs 2005) implementation would use a Scanner instead of getTextFromConsole().

A simple script of the program I/O is as follows:

```
Script started on Mon Mar 27 10:15:47 2017
prompt >>
prompt >> java Triangle
Welcome to the Triangle Program
-------------------------------
Please Enter lengths of the triangle edges
Input Length 1: 3.0
Input Length 2: 4.0
Input Length 3: 5.0
The edge lengths you have entered are:
Length 1 = 3.0
Length 2 = 4.0
Length 3 = 5.0
Area =    6.000
prompt >>
Script done on Mon Mar 27 10:16:30 2017
```

Problem 7.9: Compute and print primes up to 1,000.

Source code: For completeness, here is the source code to EulerMath.java:

```
import java.lang.Math;

public class EulerMath {
   public static boolean isPrime(long num){
      if (num < 2 || (num % 2 == 0 && num != 2))
          return false;

      for (int i = 3; i <= Math.sqrt(num); i += 2 )
           if (num % i == 0)
               return false;

      return true;
   }
}
```

And here is a program Prime to compute and print the primes:

```
/*
 * ============================================================
 * Prime.java: Compute and print prime numbers ...
 * ============================================================
 */

public class Prime {
   public static void main(String[] args) {
      int counter = 0;

      System.out.println("==========================================");
      System.out.println("Welcome to the prime number " + "generator!");
      System.out.println("The following are the prime numbers " +
                         "between 1 and 1000!");
      System.out.println("==========================================");

      for (int i = 1; i <= 1000 ; i += 1) {
         if (EulerMath.isPrime(i) == true){
             System.out.print(i + "           ");
                counter += 1;
             if (counter % 10 == 0)
                System.out.println("");
         }
      }
      System.out.println("");
```

```
  }
}
```

Output: The abbreviated output is as follows:

```
========================================
Welcome to the prime number generator!
The following are the prime numbers between 1 and 1000!
========================================
2          3          5          7          11         13
17         19         23         29         31         37
41         43         47         53         59         61

... lines of output removed ....

919        929        937        941        947        953
967        971        977        983        991        997
```

Problem 7.10:

Problem 7.11: Computes and prints water speed in three open channel cofigurations.

The source code is as follows:

```
/*
 *  ============================================================
 *  OpenChannel.java: Compute velocity of water in a rectangular
 *  open channel.
 *
 *  Written By: Mark Austin                            April, 2008
 *  ============================================================
 */

public class OpenChannel {
   public static void main ( String args[] ) {
      double n, S, B, H;

      // Define two-dimensional array of channel properties....

      double channel[][] = { { 0.022, 0.0003, 15, 2.5 },
                             { 0.020, 0.0002,  8, 1.0 },
                             { 0.025, 0.0001, 10, 2.0 } };

      // Print channel properties....

      System.out.println("");
      System.out.println("Channel Data");
      System.out.println("========================");

      for (int i = 1; i <= 3; i = i+1 )
         System.out.printf(" %7.3f %7.4f %4.1f %4.1f\n",
              channel[i-1][0], channel[i-1][1], channel[i-1][2],
              channel[i-1][3] );

      // Compute and print water velicities...

      System.out.println("");
      System.out.println("Channel Water Velocities");
      System.out.println("========================");

      for (int i = 1; i <= 3; i = i+1 ) {
         System.out.printf("Channel: %2d\n", i );
         n = channel[i-1][0];
         S = channel[i-1][1];
         B = channel[i-1][2];
         H = channel[i-1][3];

         System.out.printf("Parameter n = %7.3f\n", n );
         System.out.printf("Parameter S = %7.4f\n", S );
         System.out.printf("Parameter B = %7.2f\n", B );
```

```
         System.out.printf("Parameter H = %7.2f\n", H );
         System.out.printf("Water Velocity = %7.2f\n\n",
            Math.sqrt(S)/n*Math.pow((B*H/(B+2*H)), 2.0/3.0) );
      }
   }
}
```

And a script of the program input/output is:

```
Script started on Mon Mar 27 10:31:30 2017
prompt >>
prompt >> java OpenChannel
Channel Data
========================
   0.022  0.0003 15.0  2.5
   0.020  0.0002  8.0  1.0
   0.025  0.0001 10.0  2.0

Channel Water Velocities
========================
Channel:  1
Parameter n =   0.022
Parameter S =  0.0003
Parameter B =   15.00
Parameter H =    2.50
Water Velocity =    1.20

Channel:  2
Parameter n =   0.020
Parameter S =  0.0002
Parameter B =    8.00
Parameter H =    1.00
Water Velocity =    0.61

Channel:  3
Parameter n =   0.025
Parameter S =  0.0001
Parameter B =   10.00
Parameter H =    2.00
Water Velocity =    0.51
prompt >>
Script started on Mon Mar 27 10:31:38 2017
```

Problem 7.17: Systematic treatment of overflows, underflows and divide by zero.

Source code: Here is the source code for an approach that detects the problem before it occurs:

```
/*
 * ========================================================================
 * FunctionEvaluation1.java: This program evaluates and prints values
 * for a function y(x), x ranging from -4 to 10 in increments of 0.2.
 *
 * Three trouble spots need handling:
 *
 * 1. At x = 0, the expression x/sin(x) evaluates to not-a-number.
 * 2. At x = 2, there is a divide by zero.
 * 3. The increment 0.2 cannot be stored exactly inside the computer.
 *    Hence, tests for relational equality should be based on tolerances.
 *
 * Strategy:
 *
 * This program tests for x = 0 and x = 2 before y(x) is actually
 * evaluated, and then a suitable error message is printed.
 *
 * By: Mark Austin                                          November 2007
 * ========================================================================
 */

import java.lang.Math;
import java.util.*;
```

```
import java.io.*;
import java.text.*;

public class FunctionEvaluation1 {
    public static void main( String args[] ) {
            double dY, dX;

            // Print header and setup table...

            System.out.println("");
            System.out.println("     x | Function y(x)");
            System.out.println("----------------------");

            // Loop over range of x values....

        dX = -4.0;
        while ( dX <= 10.0 ) {

           // Detect and handle error conditions apriori.
           // Otherwise evaluate and print x, y(x).

           if ( Math.abs ( dX ) < 0.0005 )
             System.out.println("   0.0    Not-a-Number");
           else if ( Math.abs ( dX - 2.0 ) < 0.0005 )
             System.out.println("   2.0  Divide by zero");
           else {
             dY = (Math.pow (dX,4.0) + (dX/Math.sin(dX)))/(dX-2.0);
             System.out.printf("%6.1f      %10.2f\n", dX, dY );
           }

           // Increment value of dX.

           dX = dX + 0.2;
            }
    }
}
```

And here is the abbreviated code for post detection:

```
/*
 * ====================================================================
 * FunctionEvaluation2.java: This program evaluates and prints values
 * for a function y(x), x ranging from -4 to 10 in increments of 0.2.

 ... comment code removed ...

 * Strategy:
 *
 * This program evaluates y(x), tests for error conditions in the result,
 * and then prints an appropriate message.
 *
 * Note: Program demonstrates use of special error constants, Double.NaN,
 *       Double.NEGATIVE_INFINITY and Double.POSITIVE_INFINITY.
 *
 * By: Mark Austin                                          November 2007
 * ====================================================================
 */

import java.lang.Math;
import java.util.*;
import java.io.*;
import java.text.*;

public class FunctionEvaluation2 {
    public static void main( String args[] ) {
            double dY, dX;

            // Print header and setup table...

            System.out.println("");
            System.out.println("     x | Function y(x)");
            System.out.println("----------------------");
```

```
            // Loop over range of x values....

         dX = -4.0;
         while ( dX <= 10.0 ) {

            // Evaluate y(x) ....

               dY = (Math.pow (dX,4.0) + (dX/Math.sin(dX)))/(dX-2.0);

            // Detect and handle error conditions apriori.
            // Otherwise evaluate and print x, y(x).

            if ( dY == Double.POSITIVE_INFINITY )
              System.out.printf("%6.2f      Positive Infinity\n", dX );
            else if ( dY == Double.NEGATIVE_INFINITY )
              System.out.printf("%6.2f      Negative Infinity\n", dX );
            else if ( dY == Double.NaN )
              System.out.printf("%6.2f      Not a Number\n", dX );
            else
              System.out.printf("%6.2f      %10.2f\n", dX, dY );

            // Increment value of dX.

            dX = dX + 0.25;
             }
    }
}
```

Output: Abbreviated output from the first program is as follows:

```
  ... output removed ...

     x | Function y(x)
-----------------------
  -4.0          -41.79
  -3.8          -33.25

  ... output removed ...

  -0.3           -0.45
   0.0    Not-a-Number
   0.3           -0.58

  ... output removed ...

   1.8          -44.63
   2.0  Divide by zero
   2.3          114.08

  ... output removed ...

  10.0         1247.70
```

And the second program generates the output:

```
     x | Function y(x)
-----------------------
 -4.00          -41.79
 -3.75          -33.25

  ... output removed ...

 -0.25           -0.45
  0.00             NaN
  0.25           -0.58

  ... output removed ...

  1.75          -44.63
  2.00      Positive Infinity
  2.25          114.08

  ... output removed ...
```

```
  9.75         1162.11
 10.00         1247.70
```

Note: In the first version of this question said use a steplength of 0.2.
But then the program doesn't work as expected. Why? The increment 0.2
cannot be stored exactly, hence by the time the loop reaches x = 2,
the error accumulates enough not to trigger Positive Infinity.

Problem 7.24: Compute geometric parameters (area, perimeter) in a
polygon having seven sides.

The solution is comprised of two parts: (1) a text file (polygon.txt)
containing the (x,y) coordinates of the polygon, and (2) the polygon
analysis program.

Here is the polygon.txt file:

```
1.0  1.0
1.0  5.0
6.0  5.0
7.0  3.0
4.0  3.0
3.0  2.0
3.0  1.0
```

And here is the PolygonAnalysis.java source code:

```
/*
 *  ============================================================================
 *  PolygonAnalysis.java : This Java 2 program reads vertex coordinates for a
 *  polygon from a datafile and computes a variety of properties.....
 *
 *  Written By : Mark Austin                                    December, 2002
 *  ============================================================================
 */

import java.lang.Math;
import java.util.*;
import java.io.*;
import java.text.*;

class PolygonAnalysis {

    // Create two-dimensional array to store polygon coordinates.

    static float[][] polygon = new float[7][2];

    public static void main( String args[] ) throws IOException {
        float fXmin, fXmax, fYmin, fYmax;
        float fR, fRmin, fRmax;
        float fPerimeter = 0;
        float fArea      = 0;

        // Read polygon data from "polygon.txt"

        polygonInput();

        // Print details of polygon data ....

        polygonPrint();

        // Compute and print min/max "x" and "y" coordinates

        fXmin = fXmax = polygon[0][0];
        fYmin = fYmax = polygon[0][1];
        for (int i = 1; i <= 6; i = i + 1) {
             if ( polygon[i][0] > fXmax) fXmax = polygon[i][0];
             if ( polygon[i][0] < fXmin) fXmin = polygon[i][0];
             if ( polygon[i][1] > fYmax) fYmax = polygon[i][1];
             if ( polygon[i][1] < fYmin) fYmin = polygon[i][1];
        }
```

```
    System.out.println("Min X coord = " + fXmin );
    System.out.println("Max X coord = " + fXmax );
    System.out.println("Min Y coord = " + fYmin );
    System.out.println("Max Y coord = " + fYmax );

    // Compute and print min/max distance of coordinates from origin

    fRmin = fRmax = (float) Math.sqrt( polygon[0][0]*polygon[0][0] +
                                       polygon[0][1]*polygon[0][1] );
    for (int i = 1; i <= 6; i = i + 1) {
         fR = (float) Math.sqrt( polygon[i][0]*polygon[i][0] +
                                 polygon[i][1]*polygon[i][1] );
         if ( fR > fRmax) fRmax = fR;
         if ( fR < fRmin) fRmin = fR;
    }
    System.out.println("Min nodal distance = " + fRmin );
    System.out.println("Max nodal distance = " + fRmax );

    // Compute and print the polygon perimeter

    for (int i = 1; i <= 6; i = i + 1) {
       fPerimeter = fPerimeter + (float) Math.sqrt(
                    Math.pow( polygon[i][0]-polygon[i-1][0],2.0) +
                    Math.pow( polygon[i][1]-polygon[i-1][1],2.0));
    }
    fPerimeter = fPerimeter + (float) Math.sqrt(
                    Math.pow( polygon[6][0]-polygon[0][0],2.0) +
                    Math.pow( polygon[6][1]-polygon[0][1],2.0));
    System.out.println("Polygon perimeter = " + fPerimeter );

    // Compute and print the polygon area

    for (int i = 1; i <= 6; i = i + 1) {
       fArea = fArea + ( polygon[i][0]-polygon[i-1][0] ) *
                       ( polygon[i][1]+polygon[i-1][1] )/2;
    }
    fArea = fArea + ( polygon[0][0]-polygon[6][0] ) *
                    ( polygon[0][1]+polygon[6][1] )/2;
    System.out.println("Polygon area = " + fArea );
}

/*
 *  ====================================================================
 *  Method polygonInput() : Read data coordinates from "polygon.txt"
 *                          and store in "polygon" array.
 *
 *  Note. This input procedure assumes that the polygon is arranged into
 *  two columns of input, each containing seven items:
 *
 *          1.0 1.0
 *          1.0 5.0
 *          6.0 5.0
 *          7.0 3.0
 *          4.0 3.0
 *          3.0 2.0
 *          3.0 1.0
 *
 *  The input procedure reads a complete line of input and then uses the
 *  StringTokenizer to indentify "tokens" separated by blank spaces.
 *  This strategy avoids the need for a fixed format of data input.
 *
 *  Input  : None.
 *  Output : None.
 *  ====================================================================
 */

static void polygonInput() throws IOException {
float f1, f2;
String sline;
int i = 0;

    FileReader inputFile = new FileReader( "polygon.txt" );
```

```
        BufferedReader    in = new BufferedReader( inputFile );

        try {
           while( i <= 6 ) {

                // Read a new row of polygon coordinates -- this is a string.

                sline = in.readLine();
                StringTokenizer st = new StringTokenizer(sline);

                // Extract the floating-point no's from the string.

                int j = 0;
                while( st.hasMoreTokens() ) {
                   polygon[i][j] = Float.parseFloat( st.nextToken() );
                   j = j + 1;
                }
                i = i + 1;
           }
        }
        catch (FileNotFoundException e){}
        catch (EOFException e){}

        in.close();
    }

    /*
     *  ============================================================
     *  Method polygonPrint() : print details of polygon to screen.
     *
     *  Input  : polygon ... declared at top of the class.
     *  Output : None.
     *  ============================================================
     */

    static void polygonPrint() {
        for (int i = 0; i < polygon.length; i++) {
                System.out.print(polygon[i][0] + " ");
            for (int j = 1; j < polygon[i].length; j++) {
                    System.out.print(polygon[i][j] + " ");
                }
                System.out.println();
        }
    }
}
```

A script of the the program input/output is as follows:

```
Script started on Tue Mar 28 12:25:54 2017
prompt >>
prompt >> java PolygonAnalysis
1.0 1.0
1.0 5.0
6.0 5.0
7.0 3.0
4.0 3.0
3.0 2.0
3.0 1.0
Min X coord = 1.0
Max X coord = 7.0
Min Y coord = 1.0
Max Y coord = 5.0
Min nodal distance = 1.4142135
Max nodal distance = 7.81025
Polygon perimeter = 18.65028
Polygon area = 15.5
prompt >> exit
Script done on Tue Mar 28 12:26:25 2017
```
