Problem 8.1: Model rectangles with Vertices ....

Source code: Here is a very short implementation of the vertex code:

```
/**
  *  ========================================================
  *  Vertex.java: Source code for two-dimensional vertex..
  *
  *  Written by: Mark Austin                       April, 2006
  *  ========================================================
  */

public class Vertex {
   protected double dX;
   protected double dY;

   // Constructor methods ....

   public Vertex() {
      dX = dY = 0.0;
   }

   public Vertex( double dX, double dY ) {
      this.dX = dX;
      this.dY = dY;
   }

   // Convert vector to a string ...

   public String toString() {
      return "Vertex(" + dX + ", " + dY + ")";
   }
}
```

Now let's move on to the rectangle code:

```
/*
 *  ============================================================
 *  Rectangle2.java : A library of methods for creating and
 *                    managing rectangles
 *
 *  Definition of Rectangles
 *  ------------------------
 *  Rectangles are defined by the (x,y) coordinates of corner
 *  points that are diagonally opposite.
 *
 *  Methods
 *  -------------------------
 *
 *  double      area() -- returns the area of a rectangle
 *  double perimeter() -- returns the perimeter of a rectangle
 *
 *  Written By: Mark Austin                              April 2006
 *  ============================================================
 */
```

```
import java.lang.Math;

public class Rectangle2 {
   protected Vertex vertex1;  // First corner point....
   protected Vertex vertex2;  // Second corner point....

   // Constructor methods ....

   public Rectangle2() {
      vertex1 = new Vertex();
      vertex2 = new Vertex();
   }

   public Rectangle2( double dX1, double dY1, double dX2, double dY2 ) {

      vertex1 = new Vertex();
      vertex1.dX = dX1;
      vertex1.dY = dY1;

      vertex2 = new Vertex();
      vertex2.dX = dX2;
      vertex2.dY = dY2;
   }

   // Convert rectangle details to a string ...

   public String toString() {
          return "Rectangle: " + vertex1.toString() + "\n" +
                 "           " + vertex2.toString() + "\n";
   }

   // ===================================
   // Compute rectangle area and perimeter
   // ===================================

   public double area() {
       return Math.abs( (vertex2.dX - vertex1.dX) *
                        (vertex2.dY - vertex1.dY) );
   }

   public double perimeter() {
       return 2.0*Math.abs( vertex2.dX - vertex1.dX ) +
              2.0*Math.abs( vertex2.dY - vertex1.dY );
   }

   // Exercise methods in the Rectangle class ....

   public static void main ( String args[] ) {

      System.out.println("Rectangle test program     ");
      System.out.println("==========================");

      // Setup and print details of a small rectangle....

      Rectangle2 rA = new Rectangle2( 1.0, 1.0, 3.0, 4.0 );
      System.out.println( rA.toString() );

      // Print perimeter and area of the small rectangle....
```

```
      System.out.println( "Perimeter = " + rA.perimeter() );
      System.out.println( "Area      = " + rA.area() );
   }
}
```

Output: The program output is as follows:

```
prompt >> java Rectangle2
Rectangle test program
==========================
Rectangle: Vertex(1.0, 1.0)
           Vertex(3.0, 4.0)

Perimeter = 10.0
Area      = 6.0
prompt >>
```

Problem 8.2:

Source code: Here is a partial solution to the set operation methods:

```
/*
 *  ========================================================================
 *  Rectangle3.java : A library of methods for creating and
 *                    managing rectangles
 *
 *  Definition of Rectangles
 *  ------------------------
 *  Rectangles are defined by the (x,y) coordinates of corner points that
 *  are diagonally opposite.
 *
 *  Methods
 *  -----------------------
 *
 *  double      area() -- returns the area of a rectangle
 *  double perimeter() -- returns the perimeter of a rectangle
 *
 *  Written By: Mark Austin                                    April 2006
 *  ========================================================================
 */

import java.lang.Math;

public class Rectangle3 {
   protected Vertex vertex1;  // First corner point....
   protected Vertex vertex2;  // Second corner point....

   // Constructor methods ....

   public Rectangle3() {
      vertex1 = new Vertex();
      vertex2 = new Vertex();
   }

   public Rectangle3( double dX1, double dY1, double dX2, double dY2 ) {

      vertex1 = new Vertex();
      vertex1.dX = dX1;
      vertex1.dY = dY1;
```

```
   vertex2 = new Vertex();
   vertex2.dX = dX2;
   vertex2.dY = dY2;
}

// Convert rectangle details to a string ...

public String toString() {
       return "Rectangle: " + vertex1.toString() + "\n" +
              "           " + vertex2.toString() + "\n";
}

// ===================================
// Compute rectangle area and perimeter
// ===================================

public double area() {
    return Math.abs( (vertex2.dX - vertex1.dX) *
                     (vertex2.dY - vertex1.dY) );
}

public double perimeter() {
    return 2.0*Math.abs( vertex2.dX - vertex1.dX ) +
           2.0*Math.abs( vertex2.dY - vertex1.dY );
}


// ======================================================
// Evaluate (x,y) coordinate relative to a rectangle.
// ======================================================

public boolean isInside ( Vertex v ) {
   if ( Math.min ( vertex1.dX, vertex2.dX ) < v.dX &&
        v.dX < Math.max ( vertex1.dX, vertex2.dX ) &&
        Math.min ( vertex1.dY, vertex2.dY ) < v.dY &&
        v.dY < Math.max ( vertex1.dY, vertex2.dY ) )
      return true;
   else
      return false;
}

// ... details of these methods not implemented yet ....

public boolean isOutside ( Vertex v ) {
   return false;
}

public boolean isOnPerimeter ( Vertex v ) {
   return false;
}

public boolean isAbove ( Vertex v ) {
   return false;
}

public boolean isBelow  ( Vertex v ) {
   return false;
}
```

```
   public boolean isLeft ( Vertex v ) {
      return false;
   }

   public boolean isRight ( Vertex v ) {
      return false;
   }

   // ============================================
   // Exercise methods in the Rectangle class ....
   // ============================================

   public static void main ( String args[] ) {

      System.out.println("Rectangle test program     ");
      System.out.println("==========================");

      // Setup and print details of a small rectangle....

      Rectangle3 rA = new Rectangle3( 1.0, 1.0, 3.0, 4.0 );
      System.out.println( rA.toString() );

      // Create and print test points ....

      Vertex v1 = new Vertex( 0.0, 0.0 );
      System.out.println ( v1.toString() );
      Vertex v2 = new Vertex( 2.0, 2.0 );
      System.out.println ( v2.toString() );

      if ( rA.isInside( v1 ) == true )
           System.out.println("Vertex v1 is \"inside\" Rectangle rA");
      else
           System.out.println("Vertex v1 is not \"inside\" Rectangle rA");

      if ( rA.isInside( v2 ) == true )
           System.out.println("Vertex v2 is \"inside\" Rectangle rA");
      else
           System.out.println("Vertex v2 is not \"inside\" Rectangle rA");

   }
}

Output: The program output is as follows:

prompt >> java Rectangle3
Rectangle test program
==========================
Rectangle: Vertex(1.0, 1.0)
           Vertex(3.0, 4.0)

Vertex(0.0, 0.0)
Vertex(2.0, 2.0)
Vertex v1 is not "inside" Rectangle rA
Vertex v2 is "inside" Rectangle rA
prompt >>


------------------------------------------------------------------------
Problem 8.4: Working with the dataarray class and rainfall measurements.
```

Source code: For completeness, here is DataArray.java:

```
/*
 *  ====================================================================
 *  DataArray.java: Compute simple operations on a one-dimensional array
 *                  of double precision floating point numbers.
 *
 *  Written by: Mark Austin                                 November 2006
 *  ====================================================================
 */

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

public class DataArray {
    private String        sName;
    private    int  iNoElements;
    private double  data[];

    // Define NoColums to be a "constant" .... (see use below)

    public final static int NoColumns = 5;

    // Array constructors ....

    DataArray( int iNoElements )  {
        this.iNoElements = iNoElements;
        this.data = new double [ iNoElements ];

        // Initialize array elements (strictly speaking, don't need to do
this)...

        for(int i = 0; i < iNoElements; i++)
            setElement( i, (double) 0.0 );
    }

    DataArray( String sName, int iNoElements )  {
        this.sName = sName;
        this.iNoElements = iNoElements;
        this.data = new double [ iNoElements ];

        // Initialize array elements

        for(int i = 0; i < iNoElements; i++)
            setElement( i, (double) 0.0 );
    }

    // ====================================
    // Set/get name for data array
    // ====================================

    public void setName ( String sName ) {
       this.sName = sName;
    }

    public String getName() { return this.sName; }

    // ====================================
```

```
    // Retrieve and set matrix element values
    // ======================================

    public double getElement( int i ) {
        double returnValue;

        if( i < 0 || i >= iNoElements ) throw new
            RuntimeException("*** In getElement(): Array element index out of
range");

        returnValue = data[i];
        return returnValue;
    }

    public void setElement( int i, double value )  {

        if( i < 0 || i >= iNoElements ) throw new
            RuntimeException("*** In setElement(): Array element index out of
range");

        this.data[i] = value;
    }

    // ============================================
    // Convert array to string format ....
    // ============================================

    public String toString() {
         String s = "Array: " + this.sName + "\n";

        for (int i = 1; i <= iNoElements; i = i + 1) {
             s += " " + data[i-1];
             if (i % NoColumns == 0 || i == iNoElements )
                 s += "\n";
        }

         return s;
    }

    // ======================================================
    // Compute sum and difference of two data arrays...
    // ======================================================

    public DataArray Add( DataArray dA ) {

       // Check compatibility of array lengths

       if( this.iNoElements != dA.iNoElements ) throw new
           RuntimeException("*** In Add(): Incompatible array lengths");

       // Compute sum of data arrays ....

       DataArray daSum = new DataArray( this.iNoElements );
       for(int i = 0; i < iNoElements; i++)
           daSum.data[i] = data[i] + dA.data[i];

       return (daSum);
    }

    public DataArray Sub( DataArray dA ) {
```

```
   // Check compatibility of array lengths

   if( this.iNoElements != dA.iNoElements ) throw new
       RuntimeException("*** In Sub(): Incompatible array lengths");

   // Compute difference of data array values ....

   DataArray daDiff = new DataArray( this.iNoElements );
   for(int i = 0; i < iNoElements; i++)
       daDiff.data[i] = data[i] - dA.data[i];

   return (daDiff);
}

// =======================================================
// Compute basic statistics on array values ....
// =======================================================

public double max() {
   double dMaxValue = data[0];

   for(int i = 1; i < iNoElements; i++)
       dMaxValue = Math.max( data[i], dMaxValue );

   return (dMaxValue);
}

public double min() {
   double dMinValue = data[0];

   for(int i = 1; i < iNoElements; i++)
       dMinValue = Math.min( data[i], dMinValue );

   return (dMinValue);
}

public double average () {
   double dSum = 0.0;

   for(int i = 0; i < iNoElements; i++)
       dSum = dSum + data[i];

   return (dSum/iNoElements);
}

public double range () {
   return (max() - min());
}

public double std() {
  double dMean = average();

  double dSum = 0.0;
  for (int i = 0; i < iNoElements; i = i + 1)
     dSum = dSum + data[i]*data[i];

  return Math.sqrt(dSum/iNoElements - dMean*dMean);
}
```

```
    // ========================================
    // Exercise methods in array class ....
    // ========================================

    public static void main(String[] args) {

        // Create, initialize and print a small array .....

        DataArray A = new DataArray("First", 4);
        A.setElement( 0, 1.0);
        A.setElement( 1, 2.0);
        A.setElement( 2, 3.0);
        A.setElement( 3, 4.0);

        System.out.println( A.toString() );
        System.out.println( A.getName() + ".(Max element value) = " +
A.max() );
        System.out.println( A.getName() + ".(Min element value) = " +
A.min() );
        System.out.println( A.getName() + ".(Range) = " + A.range() );
        System.out.println( A.getName() + ".(Average value)      = " +
A.average() );
        System.out.println( A.getName() + ".(Standard Deviation) = " +
A.std() );
        System.out.println( "" );

        // Initialize and print a second data array ...

        DataArray B = new DataArray("Second", 4);
        B.setElement( 0, 5.0);
        B.setElement( 1, 7.0);
        B.setElement( 2, 6.0);
        B.setElement( 3, 8.0);
        System.out.println( B );

        // Compute/print A + B ...

        DataArray daSum  = A.Add(B);
        daSum.setName("A+B");
        System.out.println( daSum );

        // Compute/print A - B ...

        DataArray daDiff = A.Sub(B);
        daDiff.setName("A-B");
        System.out.println( daDiff );
    }
}
```

And here is the RainfallAnalysis.java file:

```
/*
 *  ============================================================
 *  RainfallAnalysis.java: Use data array objects to store and
 *  compute statistical analysis on monthly rainfall measurements.
 *
 *  Written By: Mark Austin                          November 2006
 *  ============================================================
 */
```

```
import java.lang.Math;

public class RainfallAnalysis {

   public static void main ( String [] args ) {

       // [a] Create data array (measured over 30 days)

       double [] fRainfall = { 1.1, 0.8, 1.1, 1.2, 0.5, 0.2,
                               0.05, 0.0, 0.0, 0.0, 0.4, 0.5,
                                0.6, 0.8, 1.0, 1.2, 3.0, 2.4,
                                1.5, 1.0, 1.0, 0.0, 0.0, 0.0,
                                0.0, 0.0, 0.0, 2.0, 2.0, 2.4 };

       // [b] Create and populate rainfall data array

       DataArray rainfall = new DataArray("March Rainfall Measurements",
30 );
       for (int i = 0; i < fRainfall.length; i = i + 1)
            rainfall.setElement ( i, fRainfall [i] );

       // [c] Print rainfall data array

       System.out.println ( rainfall.toString() );

       // [d] Compute and print rainfall measurement statistics..

       System.out.println("");
       System.out.println("Statistics of Daily Rainfall");
       System.out.println("============================");
       System.out.printf("Max value = %5.2f\n", rainfall.max() );
       System.out.printf("Min value = %5.2f\n", rainfall.min() );
       System.out.printf("Average value      = %5.2f\n",
rainfall.average() );
       System.out.printf("Standard deviation = %5.2f\n", rainfall.std() );
       System.out.println("============================");

   }
}
```

Output: The program output is as follows:

```
prompt >> java RainfallAnalysis
Array: March Rainfall Measurements
 1.1 0.8 1.1 1.2 0.5
 0.2 0.05 0.0 0.0 0.0
 0.4 0.5 0.6 0.8 1.0
 1.2 3.0 2.4 1.5 1.0
 1.0 0.0 0.0 0.0 0.0
 0.0 0.0 2.0 2.0 2.4


Statistics of Daily Rainfall
============================
Max value =  3.00
Min value =  0.00
Average value      =  0.83
Standard deviation =  0.84
============================
prompt >>
```

Problem 8.6: Comparing the sum of geometric series composed of complex numbers.

Source code: Here we assume that the files:

    Complex.java
    GeometricSeries.java

are in the same folder. The implementation uses getTextFromConsole() to read lines of input provided at the keyboard. You could also use the class Scanner.

```
/*
 *  ====================================================================
 *  GeometricSeries.java : Compute sum of a geometric series of complex
 *                         objects.
 *
 *  Written By : Mark Austin                                 March 2016
 *  ====================================================================
 */

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

public class GeometricSeries {

   public static void main( String args[] ) {
   Complex cA = new Complex();
   int    NoTermsInSeries;
   String sLine;

      // Print Welcome Message

      System.out.println("Welcome to the Geometriic Series Computer");
      System.out.println("-----------------------------------------");

      // Prompt user for coefficients in geometric series...

      System.out.println("Please enter complex number \"a\"");

      // Coefficient cA ....

      System.out.print("Coefficent a : Real : ");
      sLine    = getTextFromConsole();
      cA.dReal = Double.valueOf(sLine).doubleValue();

      System.out.print("Coefficent a : Imaginary : ");
      sLine         = getTextFromConsole();
      cA.dImaginary = Double.valueOf(sLine).doubleValue();

      // Number of terms in the series .....

      System.out.println("Please enter number of terms in series \"a\"");

      System.out.print("No of terms in series : ");
      sLine = getTextFromConsole();
```

```
      NoTermsInSeries = Integer.valueOf(sLine).intValue();

      // Print details of input to screen ...

      System.out.print("The complex no you have entered is :");
      System.out.println("s = " + cA.toString() );
      System.out.println("No of terms in series is = " + NoTermsInSeries );

      // Check that NoTermsInSeries is greater than or equal to one..

      if ( NoTermsInSeries < 1 ) {
         System.out.println("ERROS: No of terms in series must be at least
one:");
         return;
      }

      // Use basic for loop to compute sumation of geometric series .....

      System.out.println( "Summation with basic for-loop        " );
      System.out.println( "==================================== " );

      Complex cSum = new Complex();
      for (int ii = 1; ii <= NoTermsInSeries; ii = ii + 1) {

          // Compute s^(ii) ......

          Complex cB   = new Complex();
          for (int ij = 1; ij <= ii; ij = ij + 1) {
               if (ij == 1)
                   cB = cB.Add( cA );
               else
                   cB = cB.Mult( cA );
          }

          // Add s^(ii) to series sum ......

          cSum = cSum.Add( cB );
      }
      System.out.println( "Summation 1 = " + cSum.toString() );

      // Compute sum of geometric series using Horner's rule ....

      System.out.println( "" );
      System.out.println( "Summation using Horner's formula" );
      System.out.println( "==================================== " );

      Complex cI = new Complex();
      cI.dReal = 1.0; cI.dImaginary = 0.0;

      Complex cSum2 = new Complex();
      cSum2.dReal      = cA.dReal;
      cSum2.dImaginary = cA.dImaginary;

      for (int ij = 1; ij <= NoTermsInSeries - 1; ij = ij + 1) {
           cSum2 = cA.Mult( cSum2.Add(cI) );
      }

      System.out.println( "Summation 2 = " + cSum2.toString() );

      // Use geometric formula to series sum....
```

```
      System.out.println( "" );
      System.out.println( "Summation with geometric-series formula " );
      System.out.println( "====================================== " );

      Complex cB = new Complex();
      for (int ij = 1; ij <= NoTermsInSeries; ij = ij + 1) {
               if (ij == 1)
                   cB = cB.Add( cA );
               else
                   cB = cB.Mult( cA );
      }

      Complex cNumerator   = cA.Mult(cI.Sub(cB));
      Complex cDenominator = cI.Sub(cA);
      Complex cSum3 = cNumerator.Div( cDenominator );
      System.out.println( "Summation 3 = " + cSum3.toString() );

   }

   /*
    *  ============================================================
    *  Method getTextFromConsole(): Read line of text from console
    *                               (keyboard input)....
    *
    *  Input  : None.
    *  Output : String inLine -- 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;
    }
}
```

Output:

```
Script started on Fri Mar 18 09:38:05 2016
prompt >> java GeometricSeries
Welcome to the Geometriic Series Computer
-----------------------------------------
Please enter complex number "a"
Coefficent a : Real : 1
Coefficent a : Imaginary : 2
Please enter number of terms in series "a"
```

```
No of terms in series : 3
The complex no you have entered is :s = 1.0+2.0i
No of terms in series is = 3
Summation with basic for-loop
=====================================
Summation 1 = -13.0+4.0i

Summation using Horner's formula
=====================================
Summation 2 = -13.0+4.0i

Summation with geometric-series formula
=====================================
Summation 3 = -13.0+4.0i
prompt >> exit
Script done on Fri Mar 18 09:38:26 2016
```
