Homework 4

Due 9am, May 8. No extensions!

What to hand in. For each problem, hand in a copy of your java source code and a script file showing input/output from typical program runs.

Problem 1 (Intermediate): Solve problem 40 in the green class reader.

Problem 2 (Intermediate): This problem will give you practice at using the DataArray class to read, compute, and print the statistics of data collected from a structural engineering experiment in which strain measurements are recorded over an extended period of time.

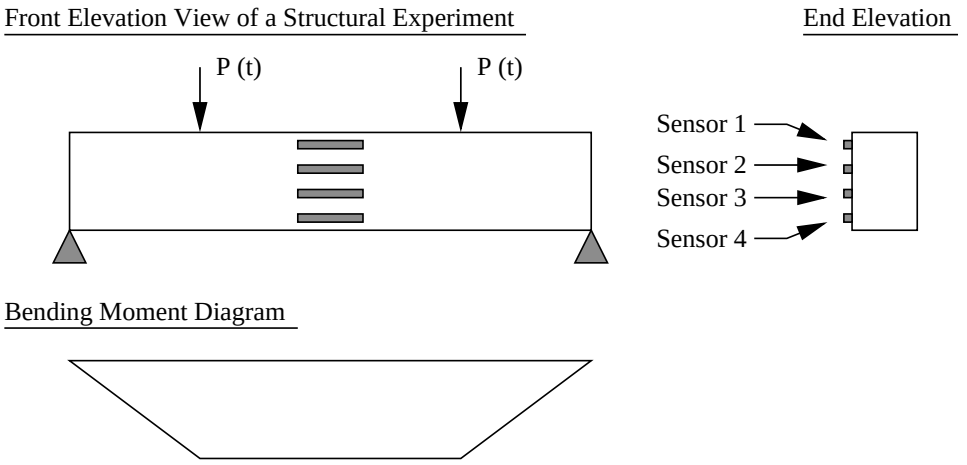

Figure 1: Structural experiment.

Figure 1 shows front and end elevation views of the experimental setup, and the bending moment diagram that will result from the symmetrically applied loading. Four sensors are attached to the center of the beam (where the bending moment will be constant. Since shear forces will be zero, any cracks that will develop will be vertical).

Now suppose that four files (i.e., `sensor1.txt`, `sensor2.txt`, `sensor3.txt` and `sensor4.txt`) contain the sensor strain measurements:

```
Sensor 1        Sensor 2         Sensor 3        Sensor 4
-------------------------------------------------------------
  -0.010          -0.0025           0.0025           0.010
  -0.011          -0.0030           0.0026           0.011
  -0.010          -0.0032           0.0027           0.010
  -0.011          -0.0034           0.0030           0.015
  -0.012          -0.0036           0.0100           0.120
  -0.015          -0.0040           0.0150           0.250
  -0.017          -0.0042           0.0250           ***** <- failed!
-------------------------------------------------------------
```

Things to do:

1. Download, compile, and run the DataArray.java program from the java examples web page.
2. Create four data files for the experimental data. I suggest that you call them `sensor1.txt` .. etc. Notice that sensor 4 fails, and hence contains an incomplete set of measurements.
3. Write a program (e.g., ExperimentalAnalysis.java) that will read and store each set of experimental measurements in a DataArray object. For each set of data, compute and print the maximum and minimum values, the range, average and standard deviation.

Note. Your solution should have six files: ExperimentalAnalysis.java (which you will write), DataArray.java (given), and the data files `sensor1.txt` through `sensor4.txt`.

Problem 3 (Intermediate): This problem will give you practice at using abstract classes to simplify the implementation of engineering property (e.g., position of the centroid, moments of inertia, orientation of the principal axes) computations for element cross sections. The computation of these properties can be complicated by irregular section shapes and/or cross section shapes that change as a function of loading (e.g., crack patterns in a concrete beam).

Figure 2 shows, for example, a spatial layout of rectangle, circle and triangle shapes. The small and large rectangles have sidelength 0.25 and 0.5 respectively. The small and large circles have radius 0.5 and 1.0 respectively. Circle and rectangle shapes are positioned at their center points. Triangles are defined by the position of the three nodal/corner points.

Engineering Property Formulae. If the total number of shapes is denoted by N, then the total area of the grid, A, is given by

$$A = \sum_{i=1}^{N} A_i \tag{1}$$

The (x,y) coordinates of the grid centroid are defined by:

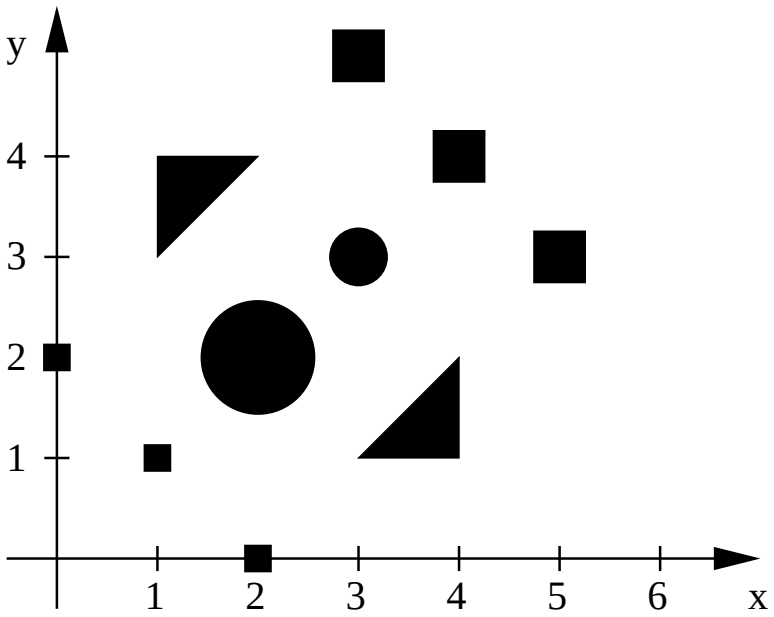

Figure 2: Spatial layout of circle, rectangle and triangle shapes.

$$A\bar{x} = \sum_{i=1}^{N} x_i \cdot A_i \quad \text{and} \quad A\bar{y} = \sum_{i=1}^{N} y_i \cdot A_i \tag{2}$$

The area moments of inertia about the x- and y-axes are given by:

$$I_{xx} = \sum_{i=1}^{N} y_i^2 \cdot A_i \quad \text{and} \quad I_{yy} = \sum_{i=1}^{N} x_i^2 \cdot A_i \tag{3}$$

respectively. Similarly the cross moment of inertia is given by

$$I_{xy} = \sum_{i=1}^{N} x_i \cdot y_i \cdot A_i \tag{4}$$

The corresponding moments of inertia about the centroid are given by the parallel axes theorem. Finally, the orientation of the principle axes are given by

$$\tan(2\theta) = \left[\frac{2I_{xy}}{I_{xx} - I_{yy}}\right] \tag{5}$$

Things to do.

The computation of engineering properties for this spatial layout can be simplified if the basic algorithms for area, centroid, and inertia calculations are specified in terms of shapes. Java will take care of the details of calling the appropriate methods within each specific shape object.

1. Download, compile and run the abstract shape example (e.g., Shape.java, Location.java, TestShape.java) from the java examples page. Then download, compile and run the Triangle code from the java examples web page.

2. The computation of engineering properties depends on quantities such as the cross section area and centroid (i.e., (x,y) location). An algorithm will need to retrieve this information from each of the object types.

   Extend the abstract shape class so that methods for retrieving the x and y coordinates are included. I suggest that you simply add the method declarations:

   ```
   public abstract double getX();
   public abstract double getY();
   ```

   to Shape.java and then add concrete implementations of of the methods `getX()` and `getY()` to Circle.java, Rectangle.java and Triangle.java.

3. Modify the Triangle code so that it extends Shape, i.e.,

   ```
   public class Triangle extends Shape { ....
   ```

   Triangles are defined by the (x,y) coordinates of the three corner points. The center point of a triangle should be defined as the average of the three respective coordinate values, i.e.,

   $$c.x = \left[\frac{x_1 + x_2 + x_3}{3}\right] \quad \text{and} \quad c.y = \left[\frac{y_1 + y_2 + y_3}{3}\right]. \tag{6}$$

4. Write a Java program that will initialize and position circle, rectangle and triangle shapes as shown on Figure 2, and then compute and print the grid area, x and y coordinates of the grid centroid, moments of inertia I_{xx}, I_{yy}, and I_{xy} computed about the axes/origin and, finally, the grid centroid. For details, see equations 1 through 4.

Hint. Notice that the layout of shapes in Figure 2 is symmetric about the line x = y. Hence, you should expect that: (1) the centroid will lie along this line, and (2) the principal axes will be oriented along this line.

Hint. To help you get started, here is the basic structure of my implementation for the engineering properties computation:

```
/*
 *  ================================================================
 *  InertiaComputation.java: Compute moments of inertia for a grid
 *                           of rectangular shapes.
 *
 *  Written By: Mark Austin                                April, 2009
 *  ================================================================
 */

public class InertiaComputation {
   public static void main ( String args[] ) {

      // Create array of shapes

      Shape s[] = new Shape [10] ;

      s[0] = new Circle   ( 0.0, 2.0, 1.0 );

      .... details of code removed ....

      s[7] = new Rectangle( 2.5, 2.5, 5.0, 3.0 );

      // Compute and print total shape area .....

      .... details of code removed ....

      // Compute and print Ixx computed about the x-axis ....

      .... details of code removed ....

      // Compute and print Iyy computed about the y-axis ....

      .... details of code removed ....

      // Compute and print Ixy computed about the origin ....

      .... details of code removed ....

      // Compute and print the centroid coordinates origin ....

      .... details of code removed ....

      // Compute and print moments of inertia about the centroid....

      .... details of code removed ....

      // Compute and print angle of principle axes....

      .... details of code removed ....
   }
}
```

Problem 4 (Intermediate): Repeat problem 3, but this time use an array list to store the shapes.