1.124J Foundations of Software Engineering

Problem Set 5

Due Date: Tuesday 10/24/00

Reference Readings: *From Java Tutorial*

- *Getting Started: Lessons 1-3*
- *Learning the Java Language: Lessons 4-7*
  - *4. Object-Oriented Programming Concepts*
  - *5. The Nuts and Bolts of the Java Language*
  - *6. Objects and Classes in Java*
  - *7. More Features of the Java Language*

Problem 1: [30%]

This problem consists of 3 small parts. In the first part, you need to provide a java application, then in the second part you must write a java applet, and, finally, at the third part you need to provide, in one file, a class which can be executed both as an applet and as an application.

For all 3 parts you need to provide a class for Point objects named ***MyPoint***. The data members of the class should be the following:

- two doubles, named ***x*** and ***y***, to store the x and y coordinates of the point.
- one static int, named ***numberMyPoints*** to keep truck of the number of points that have been created

The class should be called *MyPoint* and should be provided in a file ***MyPoint.java***. It should have two constructors:

- a non-argument constructor which should install zero values to both *x* and *y*
- a constructor which has two double as arguments by the names *x* and *y*.

The constructors should take care of increasing accordingly the number of *MyPoint* objects that are created using the static member data *numberMyPoints*.

The class should also provide a ***toString*** method which should return a string consisting of the string "(x,y) =" and the values of x and y enclosed in parentheses and separated with a comma e.g.: *(x,y) = (-4.6 , 9.5)*

Finally, the class *MyPoint* should have a member function named ***move*** which have two parameters named *dx* and *dy* of type *double*. The function *move()* should add the dx and dy to the values of the member data x and y, respectively.

Part A:

In this part you need to provide an application, i.e. which can be executed from a console (e.g. an xterm), in a file named ***ps5_1a.java***. The main function should print out the number of the *MyPoint* objects (which initially should be set to zero, since no objects would have been created).

Then, it should create (using *new*) an object named ***p1*** using the non-argument constructor (which gives zero values). Then, the number of *MyPoint* objects should be printed again out, using the corresponding static variable. The *p1* object should also be printed out making use of the *toString()* function.

Then, *MyPoint* object, named ***p2***, should be created and given the values -4.6 and 9.5, for *x* and the *y*, respectively, using the other constructor. Then, the number of *MyPoint* objects should be printed, again, out followed by the printing of the object *p2* making use of the *toString()* function.

Finally, move the point ***p1*** by *dx=4.5* and *dy=0.7*, using the member function ***move()*** of the class *MyPoint*.

The execution of the program should have an output similar to the following:

> java ps5_1a

Number of MyPoint objects = 0

Number of MyPoint objects = 1

p1: (x,y) = (0.0 , 0.0)

Number of MyPoint objects = 2

p2 = (x,y) = (-4.6 , 9.5)

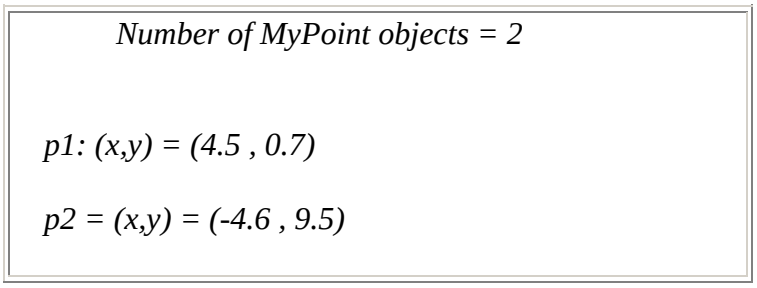
p1: (x,y) = (4.5 , 0.7)

Part B:

In this part you need to provide an applet, i.e. which can be executed from a browser or using appletviewer, in a file named ***ps5_1b.java***.

The applet should use the class *MyPoint* to define two objects, as above, ***p1*** and ***p2*** with the same values as above and using the same constructors. Create a point *p1* using the non-argument constructor and then move it as above by 4.5 and 0.7. Instead of printing to a console (xterm) it should paint to the browser, or to the appletviewer, the number of the *MyPoint* objects that have been created and the two objects, using the *toString()* member function.

The resulting image of the browser or appletviewer loading the applet *ps5_1b* should look like below:



To be able to load the applet you need to write an html file, named ***ps5_1b.html,*** which should also be submitted for grading.

Part C:

In this part you need to provide in a file named ***ps5_1c.java*** a class that can be called from the console (i.e. run as an application), or from a browser (i.e. run as an applet). If it is executed as an application should give an output similar to the one provided to part A. Otherwise, if it is loaded as an applet it should result to an image similar to the one given in part B.

Note:

Overall, for all 3 parts, you need to submit the files ***ps5_1a.java, ps5_1b.java***, and ***ps5_1c.java*** for parts A, B and C, respectively. You also need to submit the file ***MyPoint.java*** in which the class *MyPoint* should be defined. Finally, you need to submit the html files ***ps5_1b.html*** and ***ps5_1c.html*** for parts B and C.

Please, also, submit screendumps of the execution of each part, the output of parts A and C and screendump of the appletviewer (or browser) for parts B and C.

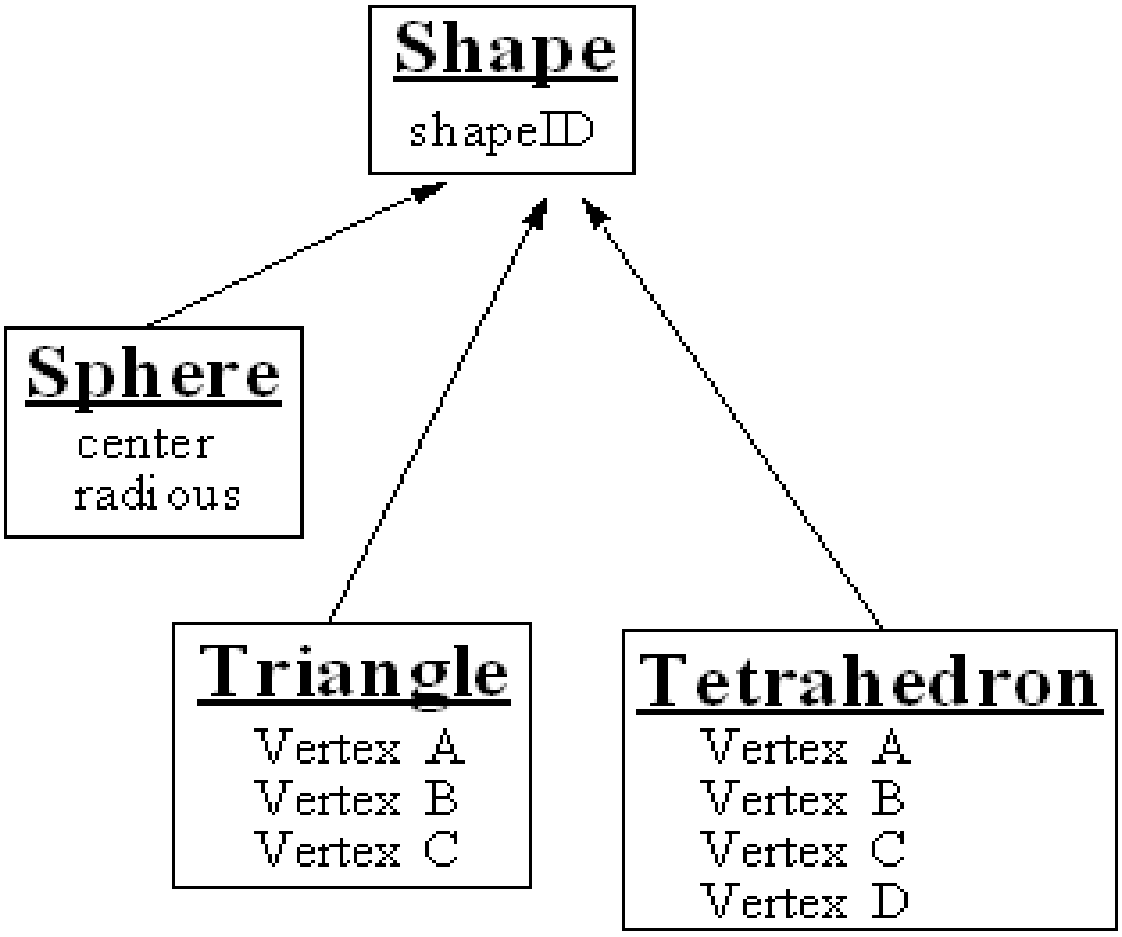
Problem 2: [35%]

For this problem you need to provide five classes, *Shape*, *Sphere*, *Triangle*, *Tetrahedron*, and *Point*, in the files ***Shape.java***, ***Sphere.java***, ***Triangle.java***, ***Tetrahedron.java,*** and ***Point.java***, respectively.

The provided classes should be able to support the provided in the file *ps5_2.java* application. This application should be able to read a number of shapes and use an array of references to *Shape* objects. The shapes can be spheres, triangles, tetrahedrons and their characteristics should be stored in the corresponding (*Sphere, Triangle, and Tetrahedron*, respectively) classes which are subclasses of the *Shape* class. You should not make **any** changes to the provided *ps5_2.java* file, which has the *main()* function and the other provided functions.

The shapes are read and then stored in an array of references to *Shape* objects. Then, all shapes are printed out using polymorphism. Finally, the references to shape objects are set to *null* and the *System.runFinalization()* and the *System.gc()* are called to force the finalizing of the objects and the garbage collector to run, respectively.

The hierarchy to be used is shown on the following figure:



You need to provide each of the above classes, namely *Shape, Sphere, Triangle, and Tetrahedron,* in the files ***Shape.java***, ***Sphere.java***, ***Triangle.java***, and ***Tetrahedron.java,*** respectively. You also need to provide the class Point, which is used by the above classes, in the file ***Point.java.*** You should figure out from the file ***ps5_2.java*** what member functions are required in order to be able to compile and run the completed program.

Several constructors, static and non-static member functions, and finalizers are required. All member data that are not explicitly specified, e.g. when a non-argument constructor is called, should be set to zero. The constructors should increase the static variable which counts the number of instances of that class. The finalize methods should decrease those numbers whenever an object is finalized. Finally, all classes should have a *toString()* method, that can be used to print the object using the *print()* method of the *System* class.

In particular you need to provide:

1. The **abstract** class ***Shape*** in the file ***Shape.java***: This class should have the following private members:

- int *shapeId*: to store the ID of the shape
- int *numberShapes:* which should be static and initialized to 0, to store the number of *Shape* objects. The constructor(s) should increase the *numberShapes* accordingly.

2. The class ***Sphere*** in the file ***Sphere.java***: This class should have the following private members:

- double *radius*: to store the radius of the sphere
- Point *center*: to store the center of the sphere
- int *numberSpheres*, : which should be static and initialized to 0, to store the number of *Sphere* objects

3. The class ***Triangle*** in the file ***Triangle.java***: This class should have the following private members:

- Point *a*: to store vertex a
- Point *b*: to store vertex b
- Point *c*: to store vertex c
- int *numberTriangles*, : which should be static and initialized to 0, to store the number of *Triangle* objects

4. The class ***Tetrahedron*** in the file ***Tetrahedron.java***: This class should have the following private members:

- Point *a*: to store vertex a
- Point *b*: to store vertex b
- Point *c*: to store vertex c
- Point *d*: to store vertex d
- int *numberTetrahedrons*, : which should be static and initialized to 0, to store the number of *Tetrahedron* objects

4. The class ***Point*** in the file ***Point.java***: This class should have the following private members:

- double *x*: to store the x coordinate of the point
- double *y*: to store the y coordinate of the point

The ***main()*** and some other functions are provided in the file ***ps5_2.java***:

ps5_2.java

```
class ps5_2
{
  static final int SIZE = 100;
  static Shape shapes[];


  public static void main(String args[])
     {
       System.out.print("\n Reading the shapes...");
       readShapes();

       System.out.print("\n Printing the shapes...");
       printShapes();

       System.out.print("\n Cleaning-up the shapes...");
       cleanUpShapes();
     }


  static void readShapes()
     {
       shapes = new Shape[SIZE];
       Point p1,p2,p3,p4,p5,p6,p7, p8;

            p1 = new Point(4.1,5.7);
            p2 = new Point(-3.6,-1.2);
            p3 = new Point(2.3,-8.2);
            p4 = new Point(-9.5,3.1);
            p5 = new Point(-5.2,4.2);
            p6 = new Point(-6.2,9.5);
            p7 = new Point(-11.6,8.6);
            p8 = new Point(-9.6, -13.6);


       shapes[Shape.getNumberShapes()] = new Sphere(11);

       ((Sphere)shapes[0]).setRadius(0.25);
       ((Sphere)shapes[0]).setCenter(-6.8,5.3);

       shapes[Shape.getNumberShapes()] = new Triangle(33,p1,p2,p3);
```

```
        shapes[Shape.getNumberShapes()] = new Sphere(101);
        shapes[Shape.getNumberShapes()] = new Triangle();
        shapes[Shape.getNumberShapes()] = new Tetrahedron(44,p4,p5,p6,p7);
        shapes[Shape.getNumberShapes()] = new Sphere();
        shapes[Shape.getNumberShapes()-1].setID(147);
        shapes[Shape.getNumberShapes()] = new Tetrahedron();
        shapes[Shape.getNumberShapes()-1].setID(67);
        ((Tetrahedron)shapes[Shape.getNumberShapes()-1]).setVertices(p1,p3,p8,p6);
        shapes[Shape.getNumberShapes()] = new Sphere();
    }


static void printShapes()
    {
        System.out.println("\n Number of shapes: " +
                                Shape.getNumberShapes());

        System.out.print("\n   Number of Spheres: " +
                               Sphere.getNumberSpheres());

        System.out.print("\n   Number of Triangles: " +
                             Triangle.getNumberTriangles());
        System.out.println("\n   Number of Tetrahedrons: " +
                    Tetrahedron.getNumberTetrahedrons());

        for(int i=0;i<Shape.getNumberShapes();i++)
          {
            System.out.print("\nShapes [ " + (i+1) + " ]: "  );

            if(shapes[i] instanceof Sphere)
                System.out.println("  Sphere " );
            else if(shapes[i] instanceof Triangle)
                System.out.println("  Triangle " );
            else if(shapes[i] instanceof Tetrahedron)
                System.out.println("  Tetrahedron " );

            System.out.println(shapes[i]);
          }
    }

static void cleanUpShapes()
    {
      System.out.println("\n\n References to shape objects are set to null");
      int n = Shape.getNumberShapes();
```

```
        for(int i=0;i<n;i++)
          {
            System.out.print("\nSetting shape [ " + (i+1) + " ]: to null"  );
            shapes[i] = null;
          }

        System.out.println("\n\n Finalizing objects");
        System.runFinalization();

        System.out.println("\n Running the Garbage Collector\n");
        System.gc();
      }
}
```

The **output** from your program should look like the following:

```
 Reading the shapes...
Printing the shapes...
Number of shapes: 8

  Number of Spheres: 4
  Number of Triangles: 2
  Number of Tetrahedrons: 2

Shapes [ 1 ]:   Sphere
 Shape:   ID = 11
       Radius = 0.25
       Center: (x,y) = (-6.8 , 5.3)

Shapes [ 2 ]:   Triangle
 Shape:   ID = 33
       Vertex a: (x,y) = (4.1 , 5.7)
       Vertex b: (x,y) = (-3.6 , -1.2)
       Vertex c: (x,y) = (2.3 , -8.2)

Shapes [ 3 ]:   Sphere
 Shape:   ID = 101
       Radius = 0.0
       Center: (x,y) = (0 , 0)

Shapes [ 4 ]:   Triangle
 Shape:   ID = 0
       Vertex a: (x,y) = (0 , 0)
       Vertex b: (x,y) = (0 , 0)
       Vertex c: (x,y) = (0 , 0)
```

Shapes [5]: Tetrahedron
Shape: ID = 44
Vertex a: (x,y) = (-9.5 , 3.1)
Vertex b: (x,y) = (-5.2 , 4.2)
Vertex c: (x,y) = (-6.2 , 9.5)
Vertex d: (x,y) = (-6.2 , 9.5)

Shapes [6]: Sphere
Shape: ID = 147
Radius = 0.0
Center: (x,y) = (0 , 0)

Shapes [7]: Tetrahedron
Shape: ID = 67
Vertex a: (x,y) = (4.1 , 5.7)
Vertex b: (x,y) = (2.3 , -8.2)
Vertex c: (x,y) = (-9.6 , -13.6)
Vertex d: (x,y) = (-6.2 , 9.5)

Shapes [8]: Sphere
Shape: ID = 0
Radius = 0.0
Center: (x,y) = (0 , 0)

Cleaning-up the shapes...

References to shape objects are set to null

Setting shape [1]: to null
Setting shape [2]: to null
Setting shape [3]: to null
Setting shape [4]: to null
Setting shape [5]: to null
Setting shape [6]: to null
Setting shape [7]: to null
Setting shape [8]: to null

Finalizing objects

Running the Garbage Collector

In Sphere finalize
In Shape finalize

In Triangle finalize
In Shape finalize

In Sphere finalize
In Shape finalize

In Triangle finalize
In Shape finalize

In Tetrahedron finalize
In Shape finalize

In Sphere finalize
In Shape finalize

In Tetrahedron finalize
In Shape finalize

In Sphere finalize
In Shape finalize

Note:

You must submit, both electronically and in a hardcopy form, the files files ***Shape.java***, ***Sphere.java***, ***Triangle.java***, ***Tetrahedron.java,*** and ***Point.java***, in which you should provide the corresponding classes as explained above. You also need to submit the provided file***ps5_2.java*** to facilitate the compilation and running of your program during grading. Please, submit a printout of the output from the completed program.

Problem 3: [35%]

In this problem you must write a Java applet that reads a number of rectangles and determines their total area and their centroid. The source code for the applet should be provided in the file ***ps5_3.java*** as a class ***ps5_3***.
The source code for the class Rectangle should be provided in the file Rectangle.java.

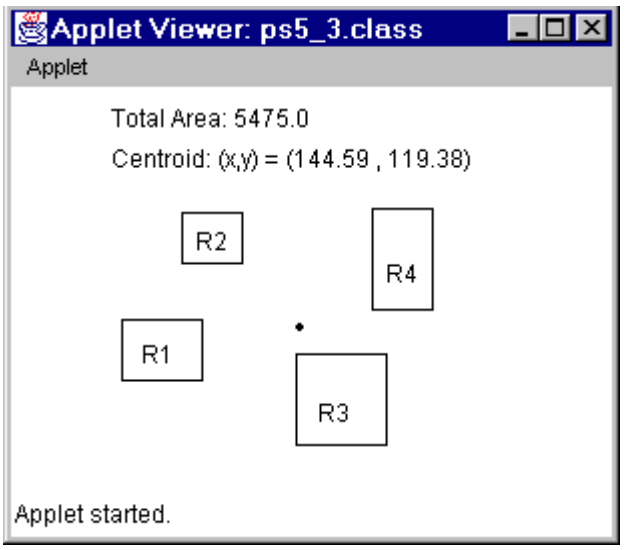
Class ***Rectangle*** should have 3 private members:

- the *center* of the rectangle, which should be a point. You may use the same class that you have already developed for the class Point for the previous problems.
- the *width* and the *height* of the rectangle, which should be doubles

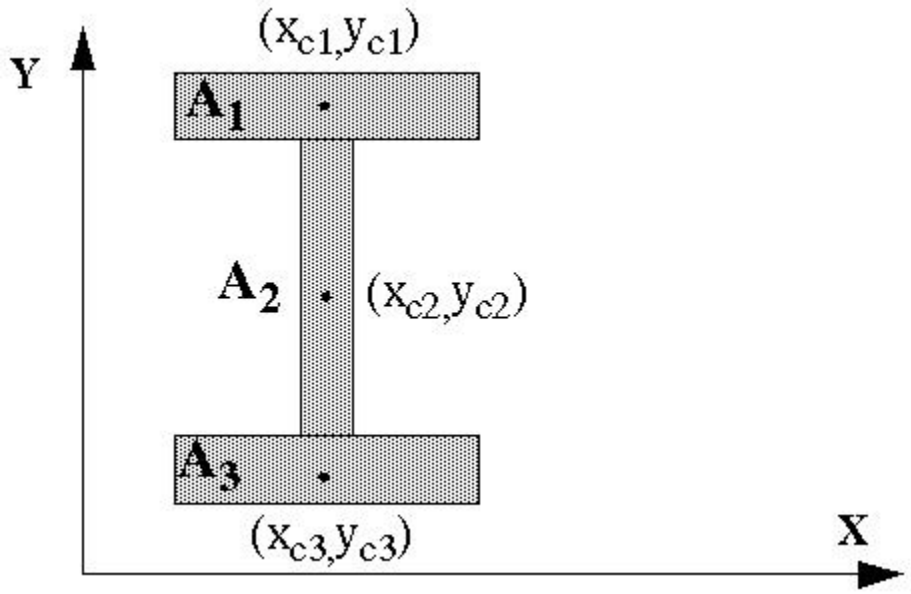
The applet ps5_3 should have an array of Rectangles of size 10. The following 4 rectangles should be created and referred from the array of rectangles. For simplicity assume that all rectangles are parallel to the x and y axes.

Rectangle	Xc	Yc	Width	Height
1	75	130	40	30
2	100	75	30	25
3	165	155	45	45
4	195	85	30	50

Then, the applet should draw each rectangle. The total area of rectangles and their centroid should be computed and printed out using two decimal points only. You also need to write the number of the rectangle, R1 for rectangle, R2 for rectangle 2, and so on. Also a circle of diameter 4 should be drawn at the centroid. The following figure shows how your applet should appear:



The centroid of a set of shapes is computed as the sum of the products of the area of each shape times its centroid coordinate (X_{ci} and Y_{ci}) divided by the total area of the cross-section. For example, the centroid (X_c,Y_c) of the following section is defined by the following formulas:



$$x_c = \frac{\sum_{\text{sections}} A_i \cdot x_{ci}}{\sum_{\text{sections}} A_i} \qquad y_c = \frac{\sum_{\text{sections}} A_i \cdot y_{ci}}{\sum_{\text{sections}} A_i}$$

You need to submit all files necessary to compile and run your applet, i.e. *ps5_3.java, Rectangle.java*, and *ps5_3.html*. You also need to submit a screendump of your applet.

Note:

Please submit **both** printouts of the source code you have written (preferably using *% enscript -2Gr -Pprinter filename*) and (or screen dumps of) the execution output (using *%xdpr -Pprinter*), with your name and username clearly written on the first page of the **stapled** submitted problem set. The submitted code must be identical to that electronically turned in (as described above).
