PROGRAMMING ASSIGNMENT 3:

Read: Savitch: Chapter 7

Programming: You will have 5 files - all should be located in a dir. named PA3:
- ShapeP3.java
- PointP3.java
- CircleP3.java
- RectangleP3.java
- TriangleP3.java

You are given the main driver/test program: PA3.java

DUE: Friday, May 3, 2002 -- 11:00pm

Turnin: turninPA3

REQUIRED:
You will write 5 classes to implement a small object inheritance heirarchy representing 2-D shapes in a cartesian plane. The object inheritance heirarchy will look like this:

```
ShapeP3
    
PointP3

CircleP3
RectangleP3
TriangleP3
```

ShapeP3, CircleP3, RectangleP3, and TriangleP3 have/use PointP3 objects (Composition - has-a relationship). CircleP3, RectangleP3, and TriangleP3 are specific types of ShapeP3 (Inheritance - is-a relationship). ShapeP3 is an abstract class. CircleP3, RectangleP3, and TriangleP3 are concrete classes.

### ShapeP3

```java
public abstract class ShapeP3 {
    private String name;

    public ShapeP3() { ... }
    public ShapeP3( String name ) { ... }

    public String getName() { ... }
    public abstract PointP3 center();
    public abstract double diameter();
    public abstract double area();
    public abstract double perimeter();
}
```

ShapeP3 is an abstract class. There are some abstract methods declared in the class definitions which will need implementations in any subclass that inherits (extends) from ShapeP3. There is one private member variable name that is a reference to a String object to indicate the name of the shape that is being created.

Two constructors: a one no-arg constructor that should initialize name to the empty string (""), and a constructor that takes a String to initialize name.
PointP3

public class PointP3
    private double x;
    private double y;

    public PointP3() { ... }
    public PointP3( double x, double y ) { ... }
    public PointP3( PointP3 p ) { ... }

    public static double length( PointP3 p1, PointP3 p2 )
    public void translate( double x, double y ) { ... }
    public String toString() { ... }
    public boolean equals( Object o ) { ... }
    public int hashCode() { ... }
    public double getX() { ... }
    public double getY() { ... }

PointP3 is a concrete class used in all the other classes to represent a point (x,y) in the cartesian plane. There are two private instance variables x and y of type double.

Three constructors: no-arg constructor which initializes the x and y values to zero, a constructor that initializes the x and y values to its parameter values, and a copy constructor.

A static (class-wide) method length() calculates the distance between two points. The algorithm to do this is \( \sqrt{(p1_x - p2_x)^2 + (p1_y - p2_y)^2} \)

You can use the Math.sqrt() method of class Math.

The translate() method adds its parameter values to this PointP3 object's x and y, respectively to translate a point to a different location.

See the sample output for an example of the String representation of a PointP3 object (toString() implementation). All numeric values will be output with precision of two decimal places to the right of the decimal point (DecimalFormat).

The equals() method is similar to the Complex equals() – check that o is not null. Check that o is a PointP3 type with getClass(), if so then cast o to PointP3 to access its x and y values to compare with this PointP3 object's x and y values.

hashCode() can be implemented as returning this.toString().hashCode() as a general hash code generator as long as the toString() is implemented representatively as discussed in class.
CircleP3

public class CircleP3 extends ShapeP3
    private PointP3 center;
    private double radius;

    public CircleP3() { ... }
    public CircleP3( PointP3 center, double radius ) { ... }
    public CircleP3( CircleP3 c ) { ... }

    public PointP3 center() { ... }
    public double diameter() { ... }
    public double area() { ... }
    public double perimeter() { ... }
    public double circumference() { ... }
    public String toString() { ... }
    public boolean equals( Object o ) { ... }
    public int hashCode() { ... }
    public PointP3 getCenter() { ... }
    public double getRadius() { ... }

CircleP3 inherits (extends) from ShapeP3 to represent circle objects in the cartesian plane. There are two private instance variables center and radius.

Each of the three constructors will set the name of the object via

    super( "CircleP3" );      or      super( c.getName() );

The no-arg constructor will initialize center to a new point at (0,0) and initialize radius to 0.0. The other two constructors will initialize center with a copy of the center point passed in as the parameter. For example,

    this.center = new PointP3( center );

and initialize radius appropriately.

The area() of a circle is $\pi r^2$ – use Math.PI for the value of $\pi$. The perimeter() and circumference() are the same – $2\pi r$.

The other methods are similar to that already described, except equals() must check that the two points (centers) are equal via this center point's equals() and not by just using the == operator. For example, something like

    this.center.equals( ((CircleP3) o).center )

getceter() returns a new PointP3 object which is a copy of the center point.

Again, all numeric values are output with two digits of precision.
RectangleP3

public class RectangleP3 extends ShapeP3
    private PointP3 upperLeftCorner;
    private double width;
    private double height;

    public RectangleP3() { ... } // other constructors
    public RectangleP3(PointP3 upperLeftCorner, double width, double height) { ... }
    public RectangleP3( RectangleP3 r ) { ... }

    public PointP3 center() { ... }
    public double diameter() { ... }
    public double perimeter() { ... }
    public double area() { ... }
    public String toString() { ... }
    public boolean equals( Object o ) { ... }
    public int hashCode() { ... }
    public PointP3 getUpperLeftCorner() { ... }
    public double getgetWidth() { ... }
    public double getHeight() { ... }

RectangleP3 inherits (extends) from ShapeP3 to represent rectangle objects in the
cartesian plane. There are three private instance variables width, height, and
upperLeftCorner which as the name implies is the point at the upper left corner
of the rectangle.

Each of the three constructors will set the name of the object via

        super( "RectangleP3" );
        or
        super( r.getName() );

The no-arg constructor will initialize upperLeftCorner to a new point at (0,0)
and initialize width and height to 0.0. The other two constructors will
initialize these instance variables appropriately similar to CircleP3
constructors -- create new PointP3 objects which are copies of the upper left
corner point passed in to the constructors to initialize upperLeftCorner.

The center() of a rectangle is calculated by translating (translate()) a copy of
the upperLeftCorner point (using PointP3's copy constructor) by width/2 in the x
plane and -height/2 in the y plane, and then returning this new point.

The diameter() of a rectangle is calculated by determining the length between the
upper left corner (upperLeftCorner) point and the lower right corner point. To
calculate the lower right corner point, translate() a copy of the upperLeftCorner
point by width in the x plane and -height in the y plane similar to how you
calculated the center point in center(). Then return the PointP3.length() between
these two points.

The perimeter() of a rectangle is just adding up all 4 sides (2*width +
2*height).

The area() of a rectangle is (width * height).

The other methods are similar to that already described. (See note about
equals() in CircleP3 section.)

getUpperLeftCorner() returns a new PointP3 object which is a copy of the upper
left corner point.

Again, all numeric values are output with two digits of precision.
TriangleP3

public class TriangleP3 extends ShapeP3
{
    private PointP3 a;
    private PointP3 b;
    private PointP3 c;

    public TriangleP3() { ... }
    public TriangleP3( PointP3 a, PointP3 b, PointP3 c ) { ... }
    public TriangleP3( TriangleP3 tri ) { ... }

    public PointP3 center() { ... }
    public double diameter() { ... }
    public double area() { ... }
    public double perimeter() { ... }
    public String toString() { ... }
    public boolean equals( Object o ) { ... }
    public int hashCode() { ... }

    TriangleP3 inherits (extends) from ShapeP3 to represent triangle objects in the cartesian plane. There are three private instance variables a, b, and c representing the three points of a triangle.

    Each of the three constructors will set the name of the object via

    super( "TriangleP3" );
    or
    super( tri.getName() );

    The no-arg constructor will initialize each point to a new point at (0,0). The other two constructors will initialize these instance variables appropriately -- new PointP3 objects that are copies of the points passed in as parameters.

    The center() of a triangle is calculated by taking the average of the x coordinates ((a_x + b_x + c_x)/3) and the average of the y coordinates ((a_y + b_y + c_y)/3), and then returning this new point.

    The diameter() of a triangle is calculated by determining the longest length between the three defining points. Use PointP3.length() to determine the line lengths between two points. Return the longest of these lengths.

    The perimeter() of a triangle is just adding up all 3 sides. Again, use PointP3.length() to determine the length between two points.

    The area() of a triangle is 0.5*base*height. But if we are given only points, the area can be calculated as follows:

    0.5 * abs( a_x*b_y + b_x*c_y + c_x*a_y - a_y*b_x - b_y*c_x - c_y*a_x )

    Use Math.abs() for the absolute value.

    The other methods are similar to that already described. (See note about equals() in CircleP3 section.) For this assignment, we can check for equality of triangles by just checking if this.a.equals( ((TriangleP3)o).a ) and the same for point b and the same for point c.

    All of these classes will be used/modified in PA4 to draw the shapes graphically using polymorphism and dynamic object creation.
PA3.java

PA3.java is given to you in the Assignments directory. We will use this class as the main test driver for the classes you write.

```java
% java PA3
Usage: java PA3 point x y
    or circle x y radius
    or rectangle x y width height
    or triangle x1 y1 x2 y2 x3 y3

x and y are the coordinates of a point, the center of a circle, upper left corner of a rectangle, or one of the points of a triangle (x2, y2, x3, y3 are the other points of the triangle). radius is the radius of the circle. width and height are the width and height of the rectangle.

SAMPLE OUTPUT:

% java PA3 point 5 -3.3
PointP3: [5.00, -3.30]
    hashCode = -2093596416
    Checking equals() with copy constructor: true

% java PA3 circle 55 44 15
CircleP3: Center = [55.00, 44.00]; Radius = 15.00
    diameter = 30.00
    circumference = 94.25
    area = 706.86
    hashCode = 1347363497
    Checking equals() with copy constructor: true

% java PA3 rectangle -55.5 -99.3 100 55.5
RectangleP3: Upper Left Corner = [-55.50, -99.30]; Width = 100.00; Height = 55.50
    center = [-5.50, -127.05]
    diameter = 114.37
    perimeter = 311.00
    area = 5550.00
    hashCode = 855608753
    Checking equals() with copy constructor: true

% java PA3 triangle 55 55 -100 -100 75.5 –110
TriangleP3: [55.00, 55.00], [-100.00, -100.00], [75.50, -110.00]
    center = [10.17, -51.67]
    diameter = 219.20
    perimeter = 561.26
    area = 14376.25
    hashCode = 494444251
    Checking equals() with copy constructor: true
```

Be sure to run the sample PA3 in the Assignments directory and test various input. Only valid numbers will be entered – no exception handling or other error checking done yet.