```
abstract public class Shape {
    public int x = 0;
    public int y = 0;

    abstract public double getArea();
    abstract public double getPerimeter();

    public void move(int dx, int dy) {
        x += dx;
        y += dy;
    }

    public String toString() {
        return "x = " + x + "; y = " + y;
    }
}
//=======================================================
class Point {
   private int x;
   private int y;

   public Point( int newX, int newY )    {
      x = newX;
      y = newY;
   }

   public int X()  {    return x;   }

   public int Y()  {    return y;   }

   public double DistanceBetween(Point pt)   {
     double distance,xd,yd;
     xd = x - pt.X();
     yd = y - pt.Y();
     distance = Math.sqrt( xd*xd + yd*yd );
     return distance;
   }
}
//=======================================================
public class Rectangle extends Shape {
    public double height;
    public double width;

    public Rectangle(double h, double w) {
        height = h;
        width = w;
    }

    public double getArea() {      return height * width;    }

    public double getPerimeter() {      return 2*height + 2*width;    }

    public String toString() {
        return "height=" + height + "; width=" + width;
    }
}
```

```
public class Square extends Rectangle {

    public Square(double x) {
        super(x, x);
    }

    public double getArea() {      return height ^ 2;    }

    public double getPerimeter() {      return 4 * height;    }

}
//=====================================================
class Triangle extends Shape
{
   protected Point vertexA, vertexB, vertexC;
   protected double sideAB, sideBC, sideAC;

    public Triangle() {  sideAB = sideBC = sideAC = 0; }

    public Triangle (Point newA, Point newB, Point newC)    {
        vertexA = newA;
        vertexB = newB;
        vertexC = newC;

        sideAB = vertexA.DistanceBetween(vertexB);
        sideBC = vertexB.DistanceBetween(vertexC);
        sideAC = vertexA.DistanceBetween(vertexC);
    }

    public double getArea()    {
        double  hp = (sideAB + sideBC + sideAC) / 2:
        return squareRoot(hp * (hp-sideAB) * (hp-sideBC) * (hp-sideAC));
    }

    public double getPerimeter()    {
        return (sideAB + sideBC + sideAC);
    }

    public double LongestSide()    {
        return (sideAB > sideBC ? ( sideAB > sideAC ? sideAB : sideAC)
                                : (sideBC>sideAC ?  sideBC : sideAC) );
    }

    public double ShortestSide()    {
      if ( sideAB < sideBC )
         if( sideAB < sideAC)
            return sideAB;
         else
            return sideAC;
      else
         if( sideBC < sideAC )
            return sideBC;
         else
            return sideAC;
    }
}
```

```
class RightTriangle extends Triangle {

    public RightTriangle(Point newA, Point newB, Point newC)    {
      // assume triangle is  A |\
      //                     | \
      //                   B |__\ C
      // thats is : AB = perpendicular
      //            BC = base
      //            AC = hypotenuse
      super ( newA, newB, newC );
    }

    public double getArea() {      return (sideBC * sideAB)/2;    }

    public double getPerimeter() {      return sideAB + sideBC + sideAC;    }

    public double LongestSide()    {     return sideAC;    }

    public double ShortestSide()    {
      return ( sideAB < sideBC ? sideAB : sideBC );
    }
}
//=====================================================
class EquilateralTriangle extends Triangle {

    public EquilateralTriangle(Point newA, Point newB, Point newC )   {
      vertexA = newA;
      vertexB = newB;
      vertexC = newC;

      sideAB = vertexA.DistanceBetween(vertexB);
      sideBC = sideAB;
      sideAC = sideAB;
    }

    public double getPerimeter()  {      return 3 * sideAB;    }

    public double LongestSide()  {      return sideAB;    }

    public double ShortestSide()   {      return sideAB;    }
}
//=====================================================
public class Circle extends Shape {

    public int radius;

    public Circle(int r) {
        radius = r;
    }

    public double getArea() {
        return (Math.PI * radius * radius);
    }

    public double getPerimeter() {
        return (Math.PI * radius * 2);
    }
}
```