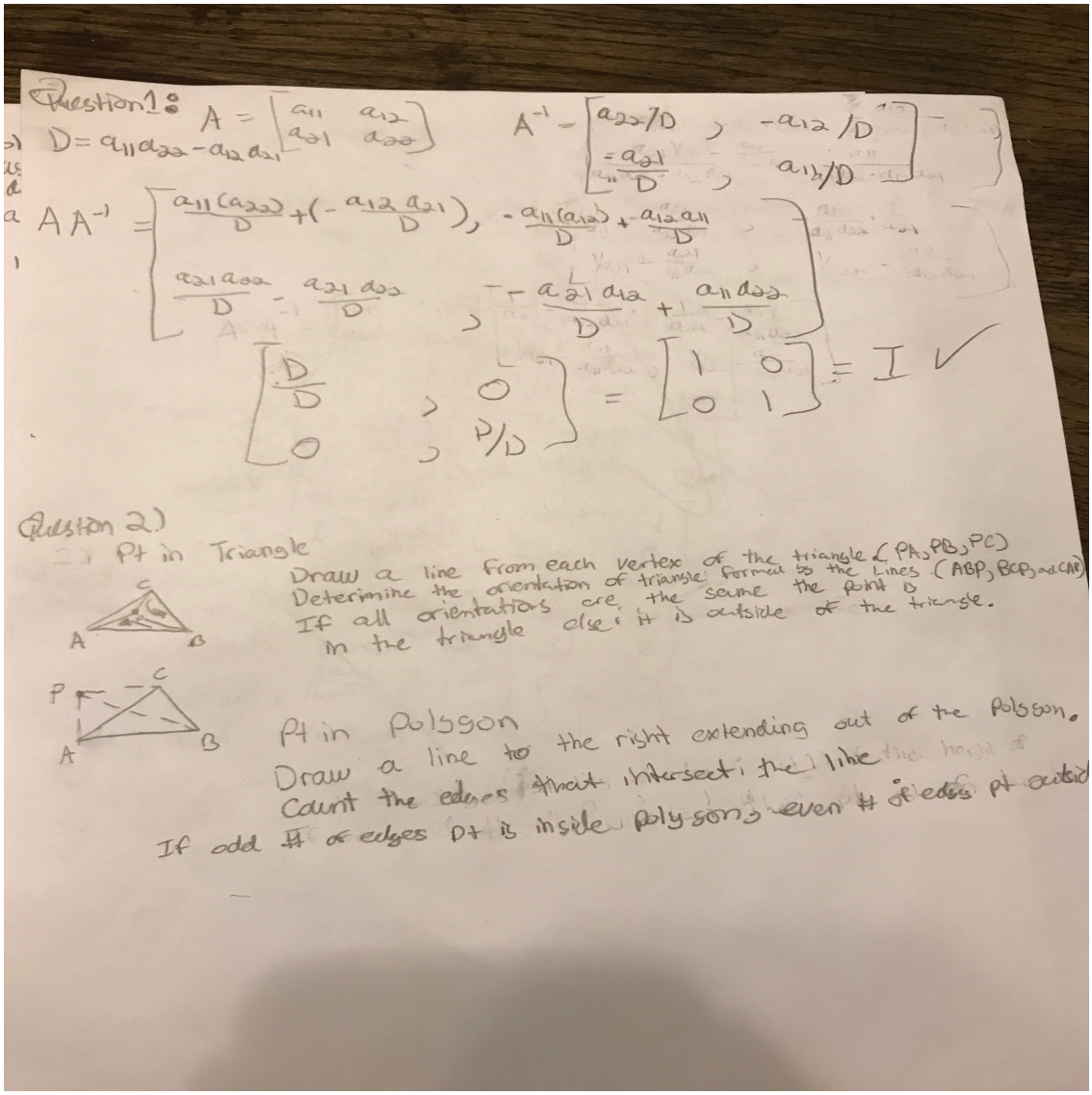
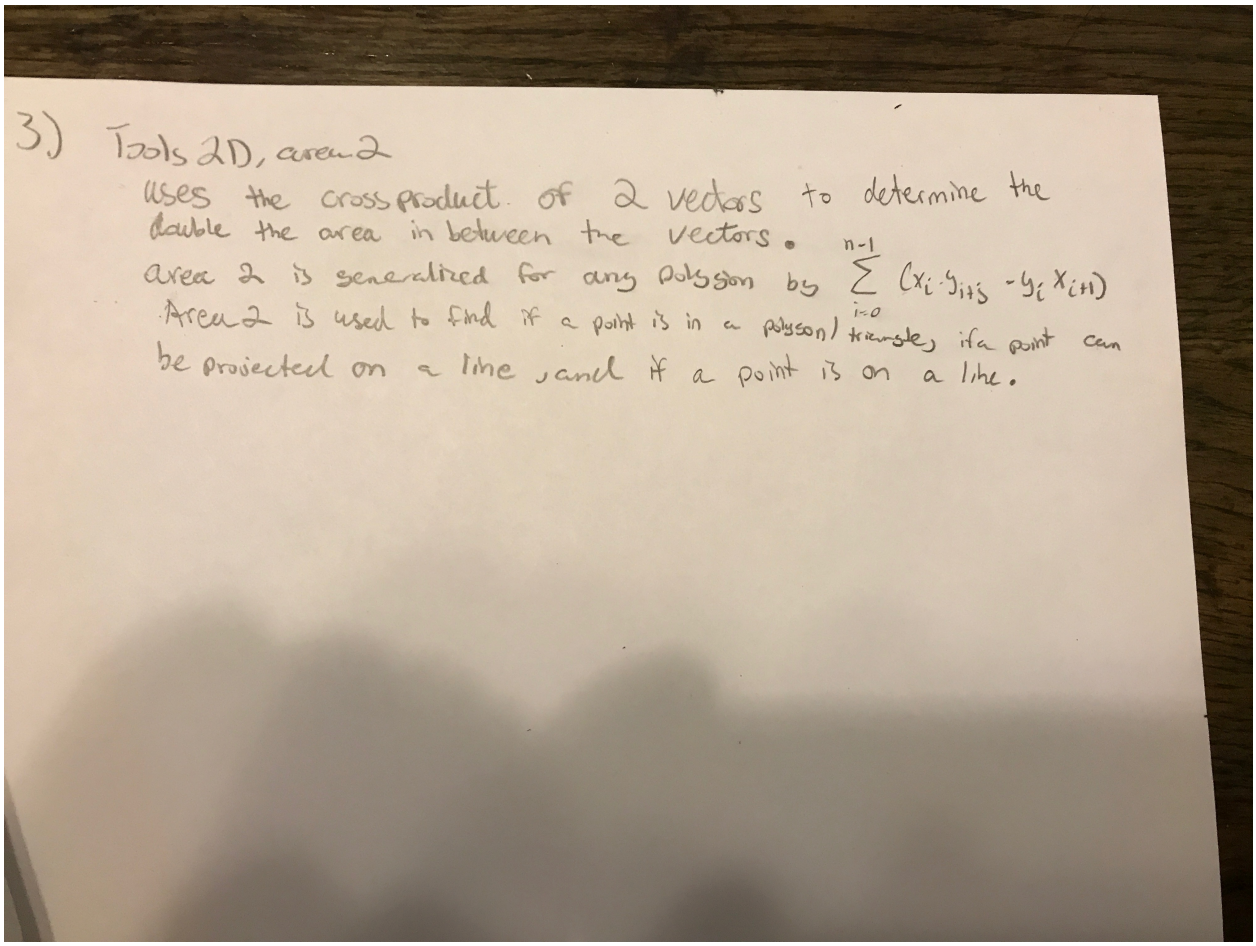
Ryan Reynolds
2693018
CIS 457

Homework 2

Questions 1 and 2:



Question 3:

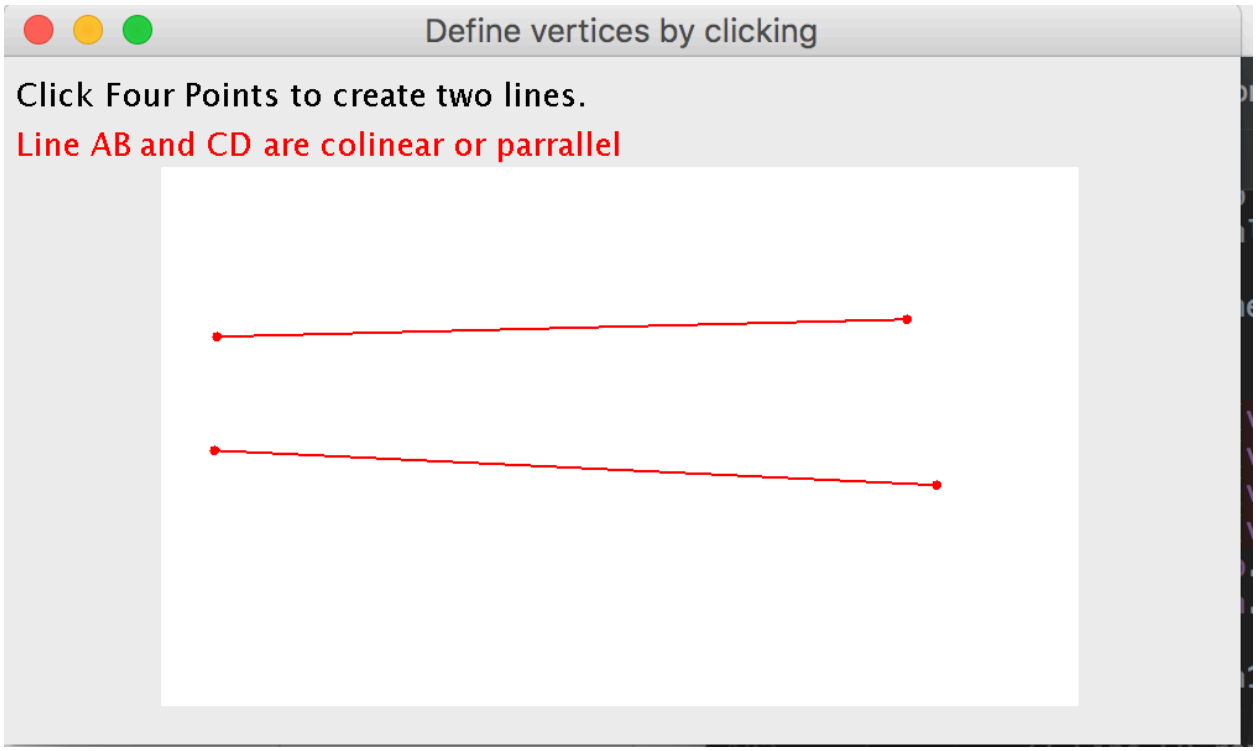


Quesiton 4:

```
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.Vector;


public class Question4 extends Frame {

    public static void main(String[] args)


    {new Question4();}
    Question4()
    {

        super("Define vertices by clicking");
```

```
        addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent e)
        { System.exit(0);}
        });


        setSize (500, 300);
        add("Center", new Lines());
        setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
        show();
    }
}



class Lines extends Canvas {


    Vector v = new Vector();

    Point2D a, b, c, d;
    float x0, y0, rWidth = 10.0F, rHeight = 7.5F, pixelSize, xI,yI;
    boolean ready = true;
    int centerX, centerY;
    int clickCounter = 0;

    Lines() {
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent evt) {
                float xA = fx(evt.getX()), yA = fy(evt.getY());
                float dx = xA - x0, dy = yA - y0;
                if (clickCounter < 4) {
                    v.addElement(new Point2D(xA, yA));
                    clickCounter = clickCounter + 1;
                    repaint();
                    ready = true;
                } else if (clickCounter == 4) {
                    ready = false;
                    v.addElement(new Point2D(xA, yA));
                }
            }
        });
    }

    public void paint(Graphics g) {

        g.drawString("Click Four Points to create two lines.", 5, 20);

        g.setColor(Color.white);

        selection();
        int left = iX(-rWidth / 2), right = iX(rWidth / 2),
                bottom = iY(-rHeight / 2), top = iY(rHeight / 2) + 60;
        g.drawRect(left, top - 15, right - left, bottom - top);
        g.fillRect(left, top - 15, right - left, bottom - top);
        g.setColor(Color.red);

        int n = 4;
        if (n == 0) return;
        a = (Point2D) (v.elementAt(0));
        g.fillOval(iX(a.x) - 2, iY(a.y) - 2, 4, 4);
```

```
        for (int i = 1; i<n; i++) {
            if (i == (4) && !ready) {
                break;
            }

            Point2D b = (Point2D) (v.elementAt(i % n));
            g.fillOval(iX(b.x) - 2, iY(b.y) - 2, 4, 4);
            if(i!=2)
            g.drawLine(iX(a.x), iY(a.y), iX(b.x), iY(b.y));
            a = b;
        }
        a=(Point2D) (v.elementAt(0));
        b=(Point2D) (v.elementAt(1));
        c=(Point2D) (v.elementAt(2));
        d=(Point2D) (v.elementAt(3));
        double a1 = b.y - a.y;
        double b1 = a.x - b.x;

        double c1 = a1*(a.x) + b1*(a.y);

        // Line CD represented as a2x + b2y = c2
        double a2 = d.y - c.y;
        double b2 = d.x - c.x;
        double c2 = a2*(c.x)+ b2*(c.y);

        double determinant = a1*b2 - a2*b1;
        double epsilon=(Math.pow(10.0,-
3.0)*(Math.pow(b1,2)+Math.pow(a1,2)+Math.pow(d.x-c.x,2)+Math.pow(a2,2)));
        if (determinant<=epsilon)
        {
            g.drawString("Line AB and CD are colinear or parrallel", 5, 40);
        }
        else
        {
            double x = (b2*c1 - b1*c2)/determinant;
            double y = (a1*c2 - a2*c1)/determinant;
            g.drawOval(iX((float)x)-5, iY((float)y)-5, 10, 10);
        }
```



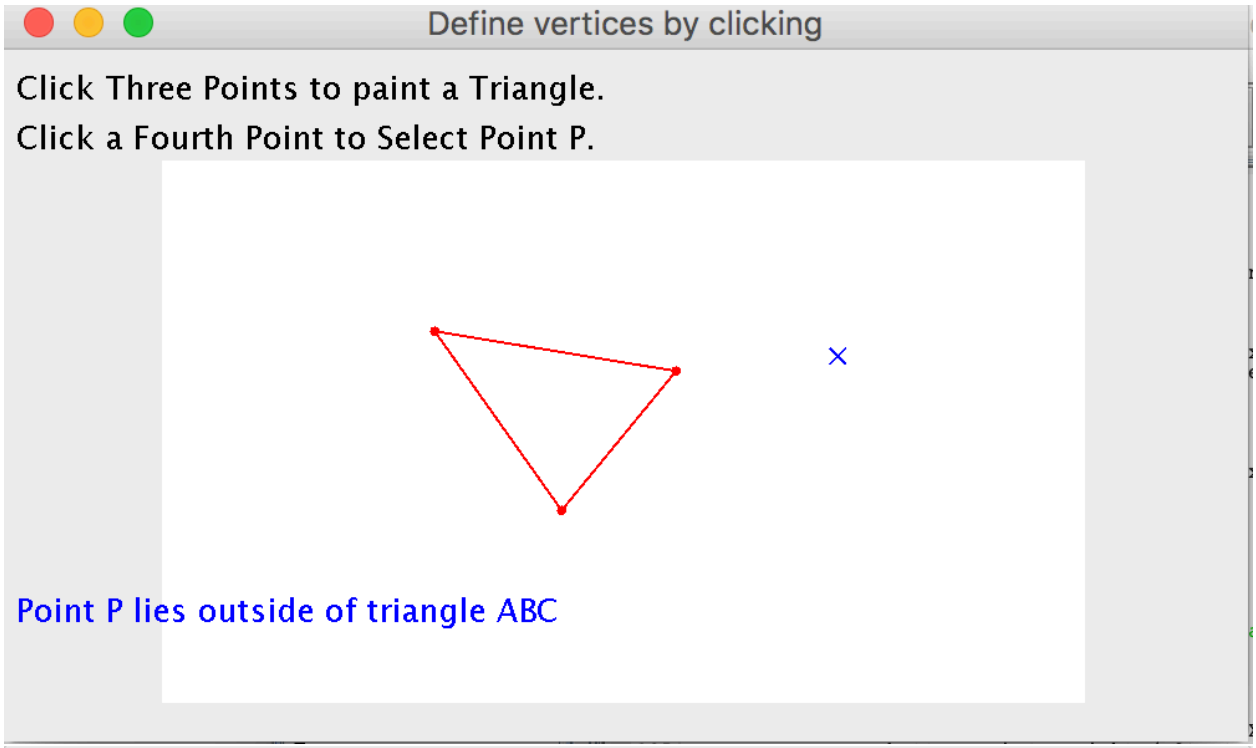
Question 5:

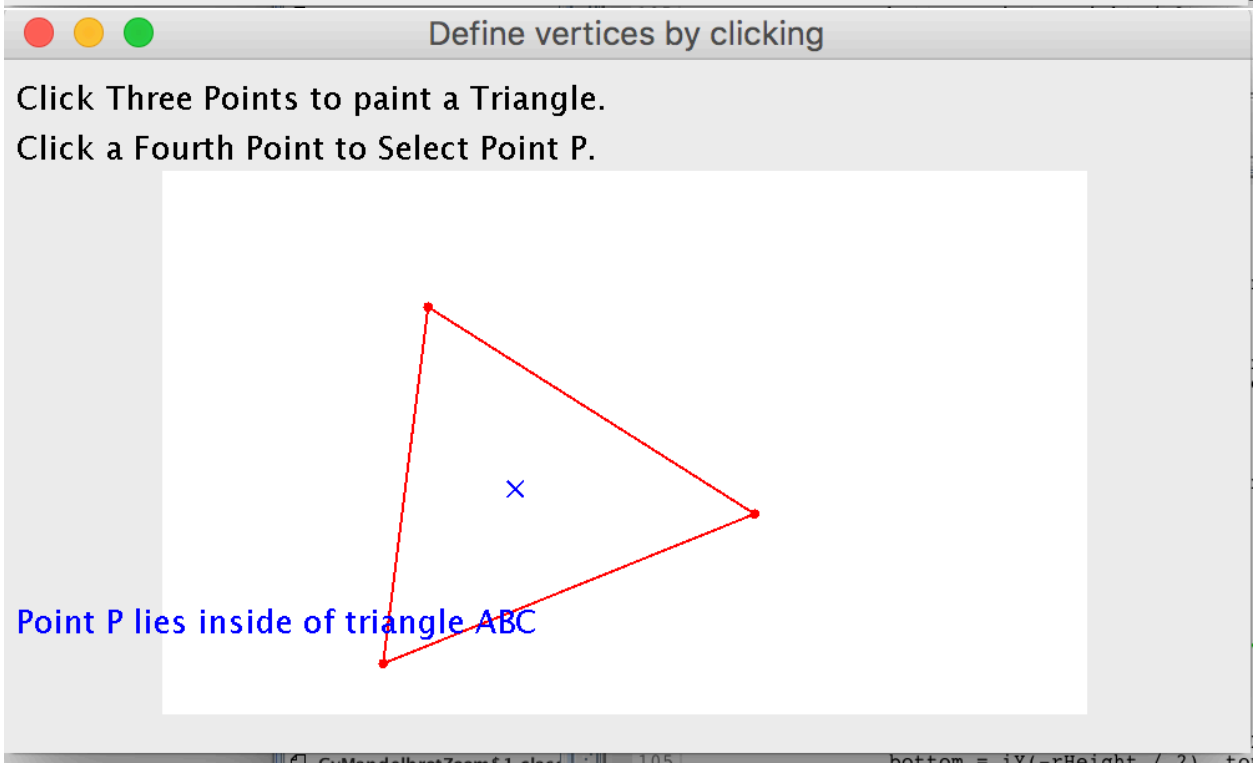
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.awt.Graphics;

public class Question5 extends Frame {

    public static void main(String[] args)


    {new Question5();}
    Question5()
    {

        super("Define vertices by clicking");
        addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent e)
        { System.exit(0);}
        });


        setSize (500, 300);
        add("Center", new Poly());
        setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
        show();
    }
}
class Tools2D
{
```

```
    static float area2(Point2D a, Point2D b, Point2D c)
    {  return (a.x - c.x) * (b.y - c.y) - (a.y - c.y) * (b.x - c.x);}


    static boolean onSegment(Point2D a, Point2D b, Point2D p)
    {  double dx = b.x - a.x, dy = b.y - a.y,
            eps = 0.001 * (dx * dx + dy * dy);
        return
                (a.x != b.x &&
                        (a.x <= p.x && p.x <= b.x || b.x <= p.x && p.x <= a.x)
                        || a.x == b.x &&
                        (a.y <= p.y && p.y <= b.y || b.y <= p.y && p.y <= a.y))
                        && Math.abs(Tools2D.area2(a, b, p)) < eps; }

    static boolean ccw(Point2D[] p)
    { int n = p.length,k=0;
        for (int i=1; i<n; i++)
            if (p[i].x <= p[k].x && (p[i].x < p[k].x || p[i].y <
                    p[k].y))
                k=i;
        int prev = k - 1, next = k + 1;
        if (prev == -1) prev = n - 1;
        if (next == n) next = 0;
        return Tools2D.area2(p[prev], p[k], p[next]) > 0; }

    static boolean insideTriangle(Point2D a, Point2D b, Point2D c, Point2D p)
    {   return
            Tools2D.area2(a, b, p) >= 0 &&
                    Tools2D.area2(b, c, p) >= 0 &&
                    Tools2D.area2(c, a, p) >= 0; }
}


class Poly extends Canvas {


    Vector v = new Vector();
    Vector p = new Vector();
    Point2D aa, bb, cc, pp;
    float x0, y0, rWidth = 10.0F, rHeight = 7.5F, pixelSize;
    boolean ready = true;
    int centerX, centerY;
    int clickCounter = 0;

    Poly() {
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent evt) {
                float xA = fx(evt.getX()), yA = fy(evt.getY());
                float dx = xA - x0, dy = yA - y0;
                if (clickCounter < 4) {
                    v.addElement(new Point2D(xA, yA));
                    clickCounter = clickCounter + 1;
                    repaint();
                    ready = true;
                } else if (clickCounter == 4) {
                    ready = false;
                    v.addElement(new Point2D(xA, yA));
                }
            }
```

```
        });
    }

    public void paint(Graphics g) {
        Tools2D tools2D = new Tools2D();
        g.drawString("Click Three Points to paint a Triangle.", 5, 20);
        g.drawString("Click a Fourth Point to Select Point P.", 5, 40);
        g.setColor(Color.white);

        selection();
        int left = iX(-rWidth / 2), right = iX(rWidth / 2),
                bottom = iY(-rHeight / 2), top = iY(rHeight / 2) + 60;
        g.drawRect(left, top - 15, right - left, bottom - top);
        g.fillRect(left, top - 15, right - left, bottom - top);
        g.setColor(Color.red);

        int n = 3;
        if (n == 0) return;
        Point2D a = (Point2D) (v.elementAt(0));
        g.fillOval(iX(a.x) - 2, iY(a.y) - 2, 4, 4);


        for (int i = 1; i <= n; i++) {
            if (i == (n) && !ready) {
                break;
            }
            Point2D b = (Point2D) (v.elementAt(i % n));
            g.fillOval(iX(b.x) - 2, iY(b.y) - 2, 4, 4);
            g.drawLine(iX(a.x), iY(a.y), iX(b.x), iY(b.y));
            a = b;
        }

        //Draws x on fourth point p as an X:
        g.setColor(Color.blue);
        Point2D p = (Point2D) (v.elementAt(3));
        float pX = p.x, pY = p.y;
        g.drawLine(iX(pX) - 3, iY(pY) + 3, iX(pX) + 3, iY(pY) - 3);
        g.drawLine(iX(pX) - 3, iY(pY) - 3, iX(pX) + 3, iY(pY) + 3);
        Point2D[] pointsArray = new Point2D[4];
        int k = 0;
        for (k = 0; k < pointsArray.length; k++) {
            pointsArray[k] = (Point2D) (v.elementAt(k));
        }


        if (Tools2D.onSegment(pointsArray[2], pointsArray[0], pointsArray[3]) == true
                || Tools2D.onSegment(pointsArray[0], pointsArray[1], pointsArray[3])
== true
                || Tools2D.onSegment(pointsArray[1], pointsArray[2], pointsArray[3])
== true) {
            g.drawString("Point P lies on an edge of triangle ABC.", 5, 230);
        } else {
            //Test if Triangle orientation is ccw
            if (Tools2D.ccw(pointsArray)) {
                if (Tools2D.insideTriangle(pointsArray[0], pointsArray[1],
pointsArray[2], pointsArray[3])) {
                    g.drawString("Point P lies inside of triangle ABC", 5, 230);
                } else {
                    g.drawString("Point P lies outside of triangle ABC", 5, 230);
                }
            } else
```

```
            {
                if (tools2D.insideTriangle(pointsArray[2], pointsArray[1],
pointsArray[0], pointsArray[3])) {
                    g.drawString("Point P lies inside of triangle ABC", 5, 230);
                } else {
                    g.drawString("Point P lies outside of triangle ABC", 5, 230);
                }
            }
        }
    }


    void selection() {
        Dimension d = getSize();
        int maxX = d.width - 1, maxY = d.height - 1;
        pixelSize = Math.max(rWidth / maxX, rHeight / maxY);
        centerX = maxX / 2;
        centerY = maxY / 2;
    }

    int iX(float x) {
        return Math.round(centerX + x / pixelSize);
    }

    int iY(float y) {
        return Math.round(centerY - y / pixelSize);
    }

    float fx(int x) {
        return (x - centerX) * pixelSize;
    }

    float fy(int y) {
        return (centerY - y) * pixelSize;
    }
}
class Point2D {
    float x, y;
    Point2D(float x, float y){this.x = x; this.y = y;}}
```


Define vertices by clicking
Click Three Points to paint a Triangle.
Click a Fourth Point to Select Point P.
Point P lies outside of triangle ABC


Define vertices by clicking
Click Three Points to paint a Triangle.
Click a Fourth Point to Select Point P.
Point P lies inside of triangle ABC