100/100

Very nice!

I had to rename your discussion log. Please follow instructions.

```
import java.awt.Point;

/**
 * Write a description of class Triangle here.
 *
 * @author Phil Fevry
 * @version 1.0
 */
public class Triangle
{
    ~~// instance variables~~
    private double angle1, angle2, angle3;
    private Point p1, p2, p3;

    /**
     * Constructor for objects of class Triangle
     */
    public Triangle(Point p1, Point p2, Point p3)
    {
        ~~// initialise instance variables~~
        this.p1 = p1;
        this.p2 = p2;
        this.p3 = p3;

        computeAngles();
    }

    /**
     * Computes and returns the distance between points p1 and p2
     *
     * @return     the distance between points p1 and p2
     */
    public double getSide1_Length()
    {
        double d;
        double x1, x2, y1, y2;

        x1 = p1.getX();
        y1 = p1.getY();

        x2 = p2.getX();
        y2 = p2.getY();

        ~~d =~~ Math.sqrt(Math.pow((x2 - x1),2) + Math.pow((y2 - y1),2));

        return ~~d;~~
    }

    /**
     * Computes and returns the distance between points p2 and p3
     *
     * @return     the distance between points p2 and p3
     */
    public double getSide2_Length()
    {
        double d;
        double x1, x2, y1, y2;

        x1 = p2.getX();
        y1 = p2.getY();

        x2 = p3.getX();
        y2 = p3.getY();
    d = Math.sqrt(Math.pow(x2 - x1,2) + Math.pow(y2 - y1,2));

    return d;
}

/**
 * Computes and returns the distance between points p1 and p3
 *
 * @return     the distance between points p1 and p3
 */
public double getSide3_Length()
{
    double d;
    double x1, x2, y1, y2;

    x1 = p1.getX();
    y1 = p1.getY();

    x2 = p3.getX();
    y2 = p3.getY();

    d = Math.sqrt(Math.pow(x2 - x1,2) + Math.pow(y2 - y1,2));

    return d;
}

/**
* Takes a parameter of type Point and sets the appropriate corner point.
*
* @param    point   sets corner point for Point 1
*/
public void setPoint1(Point p)
{
    p1.setLocation(p);
    computeAngles();
}

/**
* Takes a parameter of type Point and sets the appropriate corner point.
*
* @param    point   sets corner point for Point 2
*/
public void setPoint2(Point p)
{
    p2.setLocation(p);
    computeAngles();
}

/**
* Takes a parameter of type Point and sets the appropriate corner point.
*
* @param    point   sets corner point for Point 3
*/
public void setPoint3(Point p)
{
    p3.setLocation(p);
    computeAngles();
}

/**
 * Returns coordinates of Point 1
 *
```

```
     * @return     point 1 location
     */
    public Point getPoint1()
    {
        return p1.getLocation();
    }

    /**
     * Returns coordinates of Point 2
     *
     * @return     point 2 location
     */
    public Point getPoint2()
    {
        return p2.getLocation();
    }

     /**
     * Returns coordinates of Point 3
     *
     * @return     point 3 location
     */
    public Point getPoint3()
    {
        return p3.getLocation();
    }

    /**
     * Computes anles
     */
    private void computeAngles()
    {
        double a, b, c;
        double smallest, middle, largest;
        double cosB, sinC, sinC_2;

        // Find Largest Side
        a = getSide1_Length();
        b = getSide2_Length();
        c = getSide3_Length();

        if (Math.max(a,b) > c)
        {
            smallest = c;
            middle = Math.min(a,b);
            largest = Math.max(a,b);
        } else {
            smallest = Math.min(a,b);
            middle = Math.max(a,b);
            largest = c;
        }


        // Largest angle 'B' (Angle 1)

       cosB = (Math.pow(smallest,2) + Math.pow(middle,2) - Math.pow(largest,2))/
(2*smallest*middle);
       angle1 = Math.acos(cosB);

        // A remaining angle (Angle 2)
       sinC = ((middle * Math.sin(Math.toRadians(getAngle1())))/largest);
       sinC_2 = ((smallest * Math.sin(Math.toRadians(getAngle1())))/largest);
```

```
        // Find final angle depending on what Angle 1 and 2 are (Angle 3)
        if (sinC > sinC_2)
        {
            angle2 = Math.asin(sinC);
            angle3 = 180-(getAngle1()+getAngle2());
         } else{
            angle2 = Math.asin(sinC_2);
            angle3 = 180-(getAngle1()+getAngle2());
        }
    }

     /**
     * Returns the angle of Point 1
     *
     * @return     Point 1 angle (in degrees)
     */
    public double getAngle1()
    {
        return Math.toDegrees(angle1);
    }

     /**
     * Returns the angle of Point 2
     *
     * @return     Point 2 angle (in degrees)
     */
    public double getAngle2()
    {
         return Math.toDegrees(angle2);
    }

     /**
     * Returns the angle of Point 3
     *
     * @return     Point 3 angle (in degrees)
     */
    public double getAngle3()
    {
         return angle3;
    }

     /**
     * Returns point locations, lengths of sides, and angles of a triangle.
     *
     * @return     formatted string of triangle object
     */
    public String toString()
    {

         final int POINT1 = 0, LENGTH1 = 0, ANGLE1 = 0;
         final int POINT2 = 1, LENGTH2 = 1, ANGLE2 = 1;
         final int POINT3 = 2, LENGTH3 = 2, ANGLE3 = 2;

         final int X = 0;
         final int Y = 1;

         double [][] coordinates;
         double [] lengths, angles;

         coordinates = new double [3][2];
         coordinates[POINT1][X] = p1.getX();
         coordinates[POINT1][Y] = p1.getY();
         coordinates[POINT2][X] = p2.getX();
```

Not a good idea to store angle1 and angle2 in radians, and angle3 in degrees...

```
        coordinates[POINT2][Y] = p2.getY();
        coordinates[POINT3][X] = p3.getX();
        coordinates[POINT3][Y] = p3.getY();

        lengths = new double [3];
        lengths[LENGTH1] = getSide1_Length();
        lengths[LENGTH2] = getSide2_Length();
        lengths[LENGTH3] = getSide3_Length();

        angles = new double [3];
        angles[ANGLE1] = getAngle1();
        angles[ANGLE2] = getAngle2();
        angles[ANGLE3] = getAngle3();

        String text = String.format(
        "First Corner Point: [%1$.1f, %2$.1f]\n" +
        "Second Corner Point: [%3$.1f, %4$.1f]\n" +
        "Third Corner Point: [%5$.1f, %6$.1f]\n" +
        "Side 1 length: %7$.2f\n" +
        "Side 2 length: %8$.2f\n" +
        "Side 3 length: %9$.2f\n" +
        "Angle1: %10$.2f\n" +
        "Angle2: %11$.2f\n" +
        "Angle3: %12$.2f",
        coordinates[POINT1][X], coordinates[POINT1][Y],
        coordinates[POINT2][X], coordinates[POINT2][Y],
        coordinates[POINT3][X], coordinates[POINT3][Y],
        lengths[LENGTH1], lengths[LENGTH2], lengths[LENGTH3],
        angles[ANGLE1], angles[ANGLE2], angles[ANGLE3]);

        return text;
    }
}
```

```
import java.awt.Point;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

/**
 * The test class TriangleTest.
 *
 * @author  CS 140 Instructors
 * @version 2/16/2017
 */
public class TriangleTest
{
    private Triangle t1, t2, t3;
    private final double EPSILON = 0.0001;
    /**
     * Sets up the test fixture.
     *
     * Called before every test case method.
     */
    @Before
    public void setUp()
    {
        t1 = new Triangle(new Point(1, 3),
                          new Point(-2, -2),
                          new Point(3, -1) );

        t2 = new Triangle(new Point(1, 5),
                          new Point(-3, 2),
                          new Point(1, -2) );

        t3 = new Triangle(new Point(-3, 2),
                          new Point(5, 2),
                          new Point(0, -3) );
    }

    // testing the distance between p1 and p2
    @Test
    public void test_getSide1_Length()  {
        // rounding side1_Length to two decimal places first
        double roundOff = (double) Math.round(t1.getSide1_Length() * 100) / 100;

        assertEquals(5.83, roundOff, EPSILON);
    }


    // testing the distance between p2 and p3
    @Test
    public void test_getSide2_Length() {
        // rounding side2_Length to two decimal places first
        double roundOff = (double) Math.round(t1.getSide2_Length() * 100) / 100;

        assertEquals(5.10, roundOff, EPSILON);
    }

    // testing the distance between p3 and p1
    @Test
    public void test_getSide3_Length() {
        // rounding side3_Length to two decimal places first
        double roundOff = (double) Math.round(t1.getSide3_Length() * 100) / 100;

        assertEquals(4.47, roundOff, EPSILON);
    }

    // testing the largest angle
    @Test
    public void test_Angle1()
    {
        // rounding largest angle to two decimal places first
        double roundOff = (double) Math.round(t1.getAngle1() * 100) / 100;

        assertEquals(74.74, roundOff, EPSILON);
    }

    @Test
    public void test1_toString()
    {
        String s =
        "First Corner Point: [1.0, 3.0]\n" +
        "Second Corner Point: [-2.0, -2.0]\n" +
        "Third Corner Point: [3.0, -1.0]\n" +
        "Side 1 length: 5.83\n" +
        "Side 2 length: 5.10\n" +
        "Side 3 length: 4.47\n" +
        "Angle1: 74.74\n" +
        "Angle2: 57.53\n" +
        "Angle3: 47.73";

        assertEquals(s, t1.toString());
    }

    // After changing the first Corner point of t1 to (1,8)
    @Test
    public void test_setPoint1()
    {
        t1.setPoint1(new Point(1, 8));
        String s =
        "First Corner Point: [1.0, 8.0]\n" +
        "Second Corner Point: [-2.0, -2.0]\n" +
        "Third Corner Point: [3.0, -1.0]\n" +
        "Side 1 length: 10.44\n" +
        "Side 2 length: 5.10\n" +
        "Side 3 length: 9.22\n" +
        "Angle1: 88.78\n" +
        "Angle2: 61.99\n" +
        "Angle3: 29.23";

        assertEquals(s, t1.toString());
    }

    @Test
    public void test2_toString()
    {
        String s =
        "First Corner Point: [1.0, 5.0]\n" +
        "Second Corner Point: [-3.0, 2.0]\n" +
        "Third Corner Point: [1.0, -2.0]\n" +
        "Side 1 length: 5.00\n" +
        "Side 2 length: 5.66\n" +
        "Side 3 length: 7.00\n" +
        "Angle1: 81.87\n" +
        "Angle2: 53.13\n" +
        "Angle3: 45.00";

        assertEquals(s, t2.toString());
```

```
}

// After changing the second corner point in t2, t2 has to be:
@Test
public void test_setPoint2()
{
    t2.setPoint2(new Point(-3, 0));
    String s =
    "First Corner Point: [1.0, 5.0]\n" +
    "Second Corner Point: [-3.0, 0.0]\n" +
    "Third Corner Point: [1.0, -2.0]\n" +
    "Side 1 length: 6.40\n" +
    "Side 2 length: 4.47\n" +
    "Side 3 length: 7.00\n" +
    "Angle1: 77.91\n"+
    "Angle2: 63.43\n"+
    "Angle3: 38.66";

    assertEquals(s, t2.toString() );
}

@Test
public void test3_toString()
{
    String s =
    "First Corner Point: [-3.0, 2.0]\n" +
    "Second Corner Point: [5.0, 2.0]\n" +
    "Third Corner Point: [0.0, -3.0]\n" +
    "Side 1 length: 8.00\n" +
    "Side 2 length: 7.07\n" +
    "Side 3 length: 5.83\n" +
    "Angle1: 75.96\n" +
    "Angle2: 59.04\n" +
    "Angle3: 45.00";

    assertEquals(s, t3.toString());
}


// After changing the third Corner Point, t3 has to be
@Test
public void test_setPoint3()
{
    t3.setPoint3(new Point(-2, -3));
    String s =
    "First Corner Point: [-3.0, 2.0]\n" +
    "Second Corner Point: [5.0, 2.0]\n" +
    "Third Corner Point: [-2.0, -3.0]\n" +
    "Side 1 length: 8.00\n" +
    "Side 2 length: 8.60\n" +
    "Side 3 length: 5.10\n" +
    "Angle1: 78.69\n" +
    "Angle2: 65.77\n" +
    "Angle3: 35.54";

    assertEquals(s, t3.toString());
}

@Test
// Testing getPoint1 method.  Since getPoint1 returns
// a copy of the point1, the Triangle object should
// not change even if the copy changes
public void test_getPoint1()
```

```
    {
        Point pt1 = t1.getPoint1();
        pt1.setLocation(2, 4);  // t1’s pt1 should not change

        String s =
        "First Corner Point: [1.0, 3.0]\n" +
        "Second Corner Point: [−2.0, −2.0]\n" +
        "Third Corner Point: [3.0, −1.0]\n" +
        "Side 1 length: 5.83\n" +
        "Side 2 length: 5.10\n" +
        "Side 3 length: 4.47\n" +
        "Angle1: 74.74\n" +
        "Angle2: 57.53\n" +
        "Angle3: 47.73";

        assertEquals(s, t1.toString());
    }
}
```

```
Discussion Log
Assignment: Project 3
Name: Phil Fevry
Date: 3/6/17

Things I Learned:
- Solidified my understanding of Arrays. (1)
- AWT means abstract window toolkit (2)
- How to format Strings (3)
- JUnit’s expected beings exactly where the mismatch occurs as opposed to showin
g the entire value of both expected and calculated values.
- How more efficient it is to use JUnit.


Difficulties Faced:
- had trouble getting the math to work

- was confused why ’^2’ didn’t work for square rooting but then I realized Java
has a Math method for it

- values weren’t updating after calling setPoint# method. Realized I had to call
 computeAngles() method to recalculate the values.

- formatting text was confusing at first but I understand it now.

- was confused about why I had to use an if-else check to have two different sin
C calculations to find the second angle. through trial-and-error I found a solut
ion by swapping out the ’middle’ and ’smallest’ variables between the two sinC c
alculations but I still don’t understand why I had to do that.

Time taken to complete project:
- Around 3.5 hours (most of the time spent was trying to figure out why the math
 calculations in the computeAngles() method werenâM-^@M-^Yt working.


Resources Used:
(1) Java API
(2) Google
(3) https://www.dotnetperls.com/format-java
```

```
commit 6debe2de085df2956f60384d0bb401f4da996f70
Author: Phil Fevry <pfevry@worcester.edu>
Date:   Tue Mar 7 01:09:46 2017 -0500

    Project 3 Final Version

commit cde3ca8e36eac646c6a6c0b1cedb7aa6f34b3a77
Author: Aparna Mahadev <amahadev@worcester.edu>
Date:   Tue Feb 21 10:04:20 2017 -0500

    Project3 with gitignore added

commit 73ef8835979050af19ce48d34fed9c3f72f6bf58
Author: Aparna Mahadev <amahadev@worcester.edu>
Date:   Mon Feb 20 15:47:18 2017 -0500

    Project 3
```