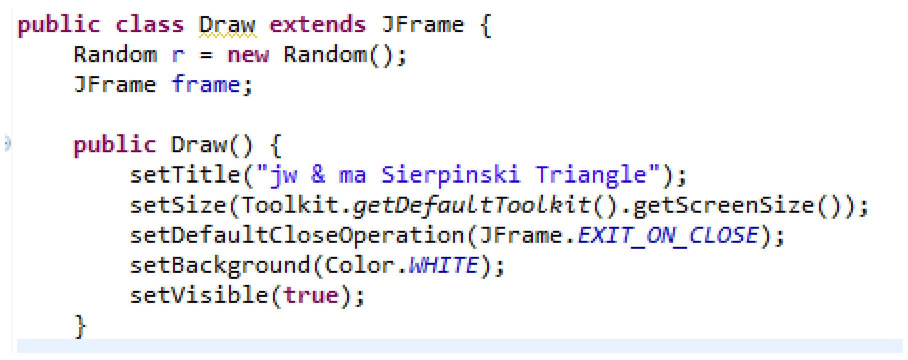
# Laboratory 9: Recursive Triangles

(http://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer9.shtml)

## 1. First set up a Window that can handle drawing. Can you get the Window to draw an equilateral triangle? What is the largest one you can get on the screen?

First of all we create a new class Draw which extends JFrame. We import java.awt.* and javax.swing.* for drawing and java.util.Random for random coloring (gets used later). Now the class is ready for displaying graphics.

The constructor sets the attributes for the JFrame we want to use: frame title, frame size (automatically uses the user's screen resolution), default close operation, background color and visibility.



```
public class Draw extends JFrame {
    Random r = new Random();
    JFrame frame;

    public Draw() {
        setTitle("jw & ma Sierpinski Triangle");
        setSize(Toolkit.getDefaultToolkit().getScreenSize());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBackground(Color.WHITE);
        setVisible(true);
    }
```

Now to the first method: We create a new method paint(Graphics g). Through this the method gets the Graphics object g from JFrame.

In this method we use Dimension size = this.getSize(); to get the resolution of the screen into a variable called size. Then we use int top, bottom, left and right to set some boundaries so that later the triangle can be fitted into the screen easily.

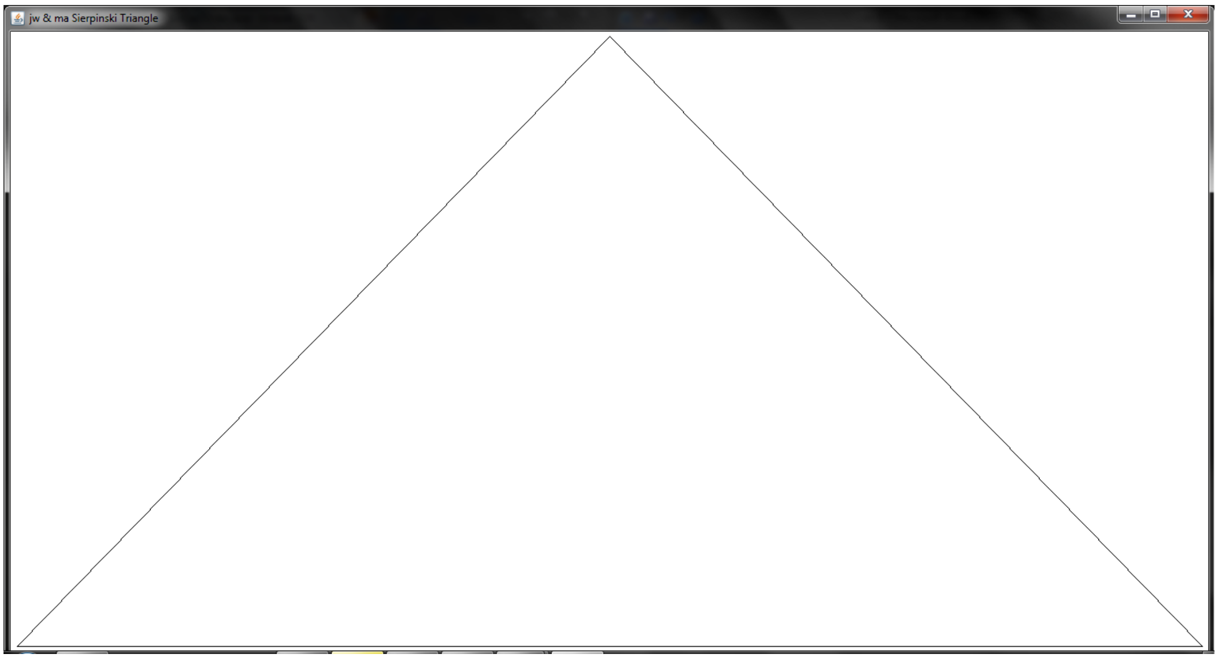
Now we set three different points with the screen resolutions stored in the variable size, 0 and the Integers for the fitting. These are the edges for a triangle which is in the middle of the screen and as big as possible. The points get connected in another method drawTriangle via the Graphics method drawPolygon.

```
public void paint(Graphics g) {
    Dimension size = this.getSize();
    int top = 35;
    int bottom = -45;
    int left = 15;
    int right= -15;

    Point p1 = new Point (0 + left, size.height + bottom);
    Point p2 = new Point(size.width / 2, 0 + top);
    Point p3 = new Point(size.width + right, size.height + bottom);

    drawTriangle(p1, p2, p3, g);
}

public void drawTriangle(Point p1, Point p2, Point p3, Graphics g) {
    int[] x1 = { p1.x, p2.x, p3.x };
    int[] y1 = { p1.y, p2.y, p3.y };

    g.drawPolygon(x1, y1, x1.length);
}
```

Now to start up the program we need a main method: we create a new class Main with a simple main method that calls a new object from class Draw:

```
public class Main {

    public static void main(String[] args) {
        new Draw();
    }
}
```

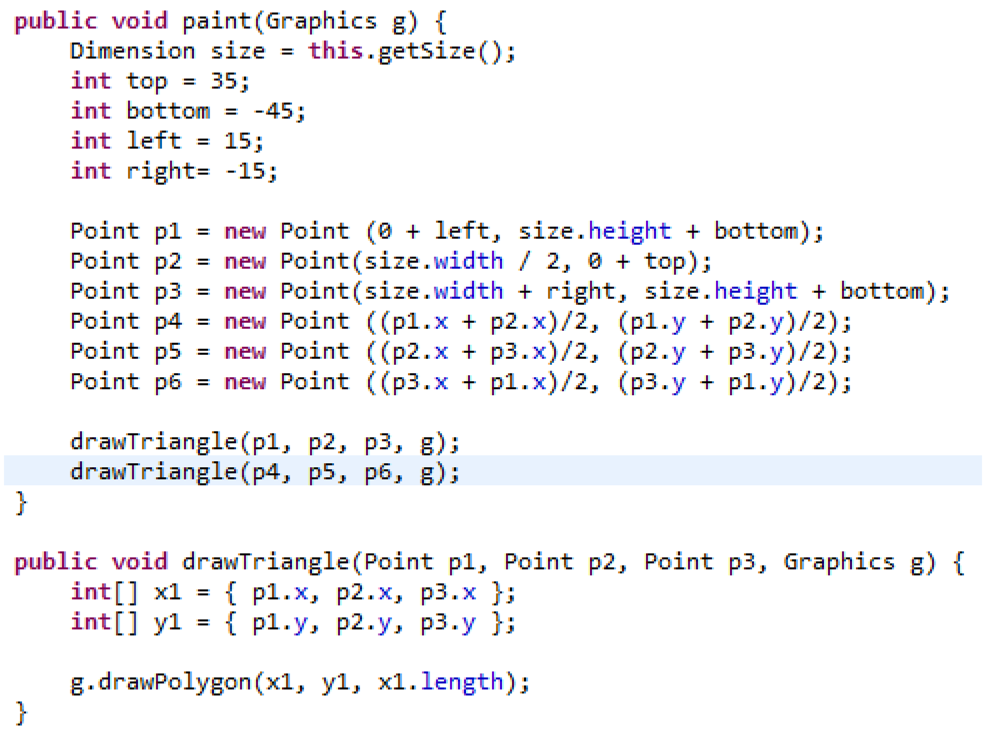
We test the program and…



…it works perfectly! You also can resize the window!

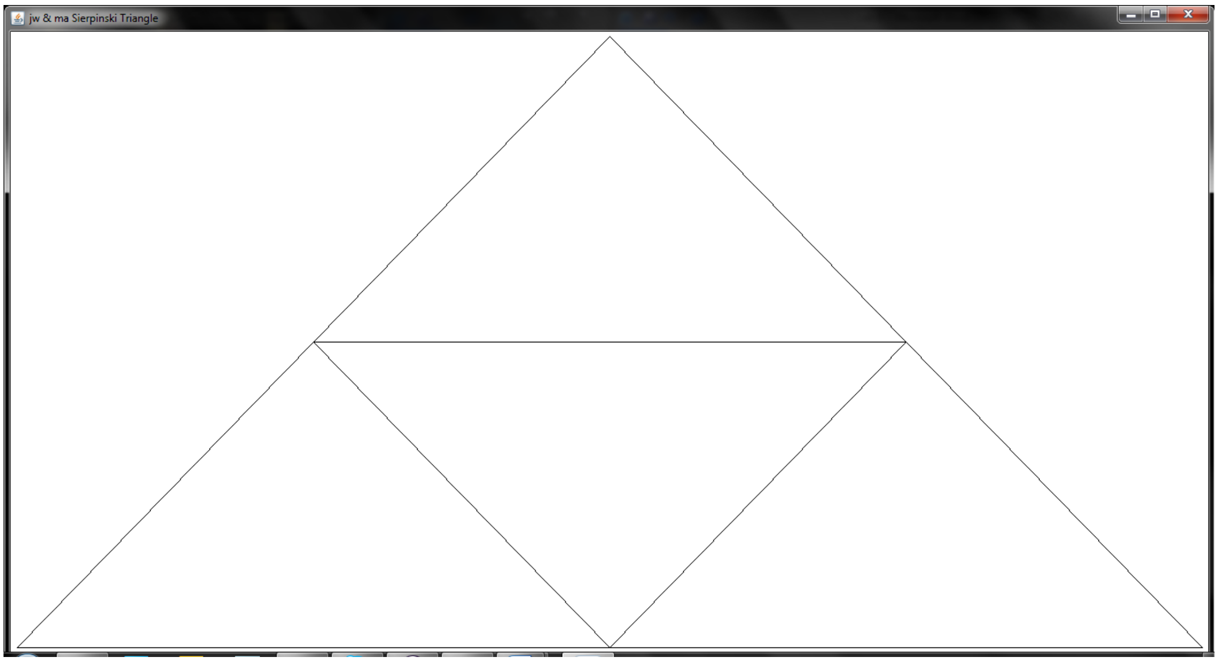
## 2. Once you can draw the triangle, now draw a triangle that connects the midpoints of each of the lines. You now have 4 triangles. For each of the three outer triangles, recursively draw a triangle that connects the midpoints. What is your termination condition, what is the measure?

We simply create 3 new Points p4, p5 and p6 in the middle of the existing Points p1, p2 and p3 (added x coordinates /2 and added y coordinates /2). Then method drawTriangle() draws another Polygon which connects the new points:



```
public void paint(Graphics g) {
    Dimension size = this.getSize();
    int top = 35;
    int bottom = -45;
    int left = 15;
    int right= -15;

    Point p1 = new Point (0 + left, size.height + bottom);
    Point p2 = new Point(size.width / 2, 0 + top);
    Point p3 = new Point(size.width + right, size.height + bottom);
    Point p4 = new Point ((p1.x + p2.x)/2, (p1.y + p2.y)/2);
    Point p5 = new Point ((p2.x + p3.x)/2, (p2.y + p3.y)/2);
    Point p6 = new Point ((p3.x + p1.x)/2, (p3.y + p1.y)/2);

    drawTriangle(p1, p2, p3, g);
    drawTriangle(p4, p5, p6, g);
}

public void drawTriangle(Point p1, Point p2, Point p3, Graphics g) {
    int[] x1 = { p1.x, p2.x, p3.x };
    int[] y1 = { p1.y, p2.y, p3.y };

    g.drawPolygon(x1, y1, x1.length);
}
```

Now the test:

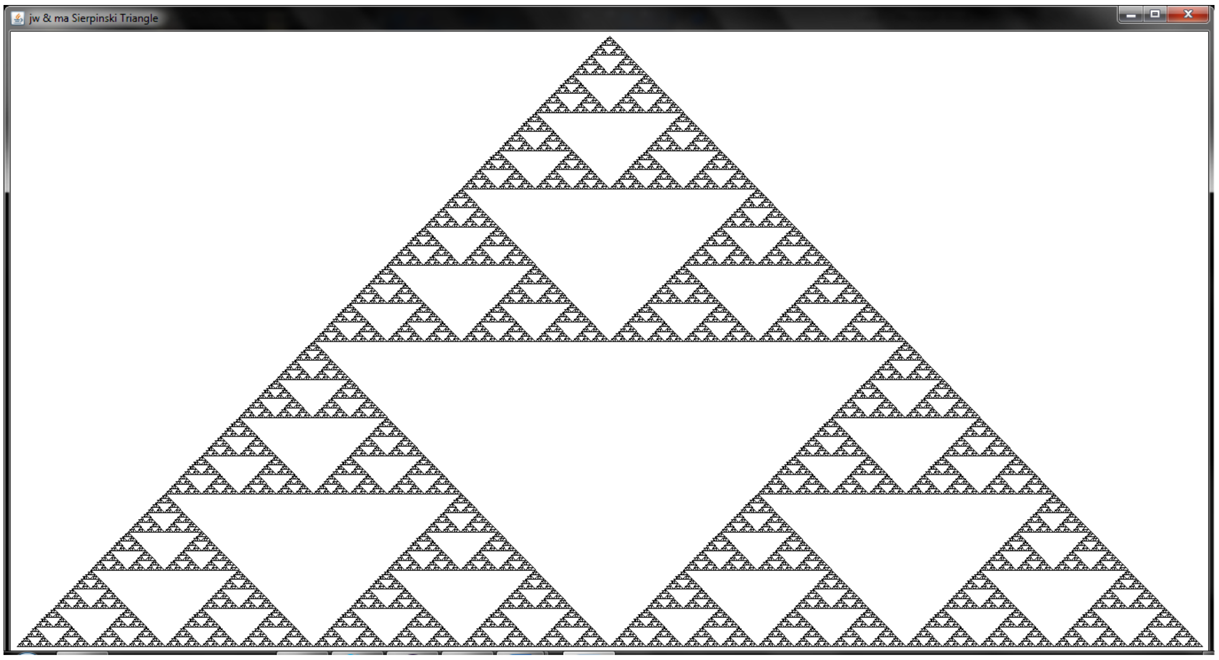


It works!

Now the recursion: we create a new method drawTriangleRecursive()

```
public void drawTriangleRecursive(Point p1, Point p2, Point p3, Graphics g) {
    //Wenn die Seite des Dreiecks kuerzer ist als 5 Pixel -> Abbruch
    if(Math.sqrt( (p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y) ) < 5) {
        return;
    }

    //Mittelpunkte des großen Dreiecks (P1, P2, P3)
    Point p4 = new Point ((p1.x + p2.x)/2, (p1.y + p2.y)/2);
    Point p5 = new Point ((p2.x + p3.x)/2, (p2.y + p3.y)/2);
    Point p6 = new Point ((p3.x + p1.x)/2, (p3.y + p1.y)/2);

    int[] x2 = { p4.x, p5.x, p6.x };
    int[] y2 = { p4.y, p5.y, p6.y };

    g.drawPolygon(x2, y2, x2.length);
    //Für jedes der drei neu entstandenen Dreiecke:

    //unten links im Eck zeichnen
    drawTriangleRecursive(p1, p4, p6, g);
    //oben im Eck zeichnen
    drawTriangleRecursive(p4, p2, p5, g);
    //unten rechts im Eck zeichnen
    drawTriangleRecursive(p6, p5, p3, g);
}
```

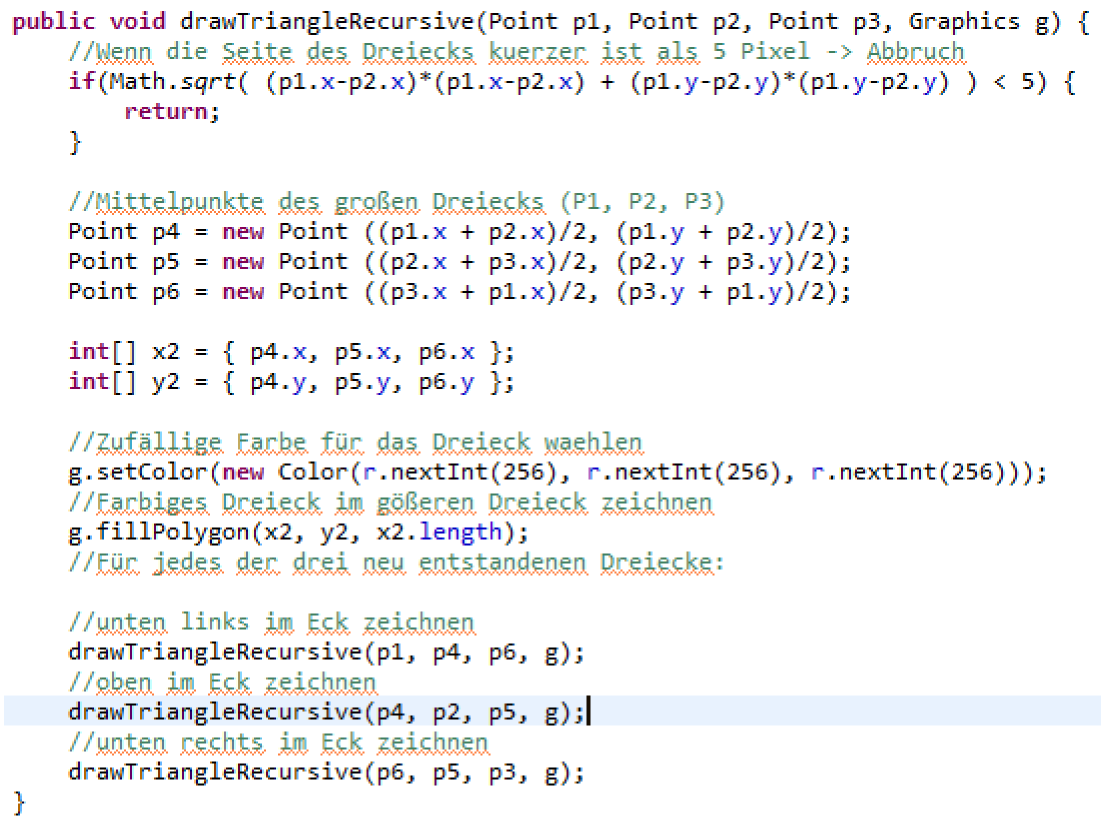
This recursive method calls itself until the smallest lines of the new triangles are smaller than 5px, which is the termination condition (base case). The measure would be the decreasing length of the triangle edges each recursive call.



It works! The amount of triangles can be controlled in the if-statement (here: 5).

## 3. Expand your triangle drawing algorithm to draw in a specific color. Choose a different color for every level of the algorithm.

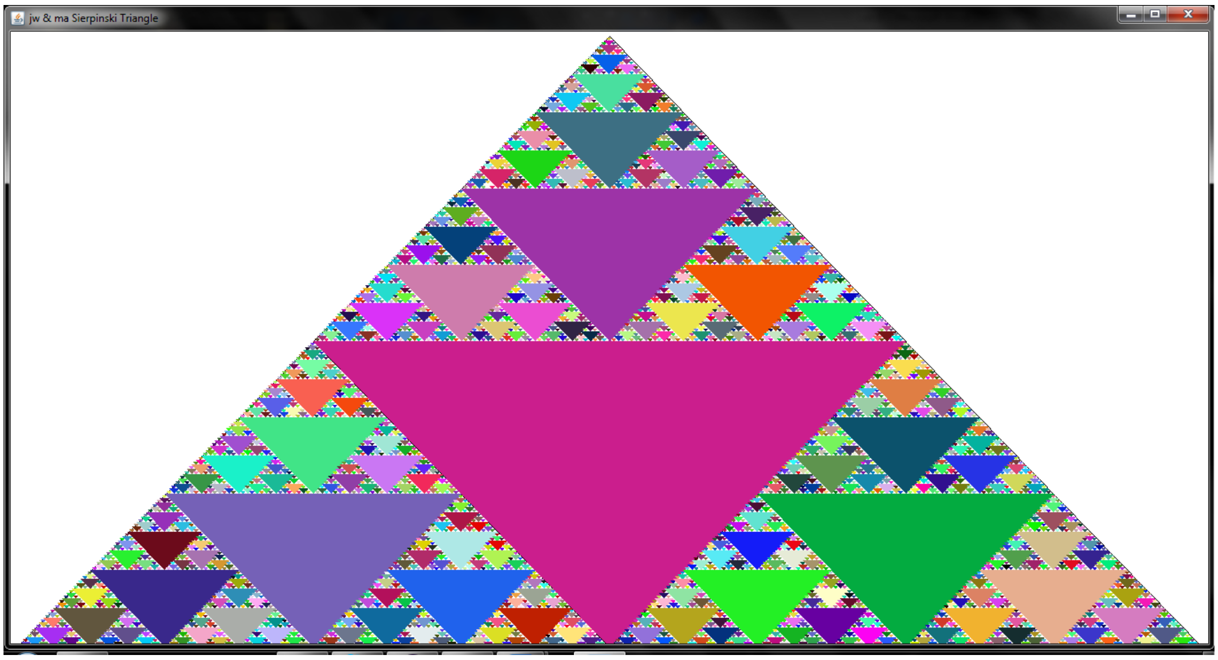
We wanted to have more fun with the many many little triangles, so we decided to choose the color of each triangle randomly. We add `g.setColor(new Color(r.nextInt(256), r.nextInt(256), r.nextInt(256)));` to our method drawTriangleRecursive and change drawPolygon to fillPolygon. Now each triangle gets filled with a random color on each recursive step.



```
public void drawTriangleRecursive(Point p1, Point p2, Point p3, Graphics g) {
    //Wenn die Seite des Dreiecks kuerzer ist als 5 Pixel -> Abbruch
    if(Math.sqrt( (p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y) ) < 5) {
        return;
    }

    //Mittelpunkte des großen Dreiecks (P1, P2, P3)
    Point p4 = new Point ((p1.x + p2.x)/2, (p1.y + p2.y)/2);
    Point p5 = new Point ((p2.x + p3.x)/2, (p2.y + p3.y)/2);
    Point p6 = new Point ((p3.x + p1.x)/2, (p3.y + p1.y)/2);

    int[] x2 = { p4.x, p5.x, p6.x };
    int[] y2 = { p4.y, p5.y, p6.y };

    //Zufällige Farbe für das Dreieck waehlen
    g.setColor(new Color(r.nextInt(256), r.nextInt(256), r.nextInt(256)));
    //Farbiges Dreieck im gößeren Dreieck zeichnen
    g.fillPolygon(x2, y2, x2.length);
    //Für jedes der drei neu entstandenen Dreiecke:

    //unten links im Eck zeichnen
    drawTriangleRecursive(p1, p4, p6, g);
    //oben im Eck zeichnen
    drawTriangleRecursive(p4, p2, p5, g);
    //unten rechts im Eck zeichnen
    drawTriangleRecursive(p6, p5, p3, g);
}
```

## 4. Fill the middle triangle on each step with an appropriate color. Choose the size of the first triangle depending on what size the window is. Redraw the triangle when the window is resized.

We already did that because of the random colors in task 3. If the user resizes the window all triangles get resized and a new random color. And here it is:



Everything works just fine!

In this lab we learned something about drawing graphics to the screen on and on because of a recursive method.

All in all the lab took us about 5 hours.

## Source code:

### Class Draw:

```
import java.awt.*;
import java.util.Random;

import javax.swing.*;

public class Draw extends JFrame {
	Random r = new Random();
	JFrame frame;

	public Draw() {
		setTitle("jw & ma Sierpinski Triangle");
		setSize(Toolkit.getDefaultToolkit().getScreenSize());
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBackground(Color.WHITE);
		setVisible(true);
	}

	public void paint(Graphics g) {
		Dimension size = this.getSize();
		int top = 35;
		int bottom = -45;
		int left = 15;
		int right= -15;

		Point p1 = new Point (0 + left, size.height + bottom);
		Point p2 = new Point(size.width / 2, 0 + top);
		Point p3 = new Point(size.width + right, size.height + bottom);
		Point p4 = new Point ((p1.x + p2.x)/2, (p1.y + p2.y)/2);
		Point p5 = new Point ((p2.x + p3.x)/2, (p2.y + p3.y)/2);
		Point p6 = new Point ((p3.x + p1.x)/2, (p3.y + p1.y)/2);

		drawTriangle(p1, p2, p3, g);
		drawTriangle(p4, p5, p6, g);
		drawTriangleRecursive(p1, p2, p3, g);
	}

	public void drawTriangle(Point p1, Point p2, Point p3, Graphics g) {
		int[] x1 = { p1.x, p2.x, p3.x };
		int[] y1 = { p1.y, p2.y, p3.y };

		g.drawPolygon(x1, y1, x1.length);
	}

	public void drawTriangleRecursive(Point p1, Point p2, Point p3, Graphics g)
{
		//Wenn die Seite des Dreiecks kuerzer ist als 5 Pixel -> Abbruch
		if(Math.sqrt( (p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-p2.y) ) <
5) {
			return;
		}

		//Mittelpunkte des großen Dreiecks (P1, P2, P3)
		Point p4 = new Point ((p1.x + p2.x)/2, (p1.y + p2.y)/2);
```

```
			Point p5 = new Point ((p2.x + p3.x)/2, (p2.y + p3.y)/2);
			Point p6 = new Point ((p3.x + p1.x)/2, (p3.y + p1.y)/2);

			int[] x2 = { p4.x, p5.x, p6.x };
			int[] y2 = { p4.y, p5.y, p6.y };

			//Zufällige Farbe für das Dreieck waehlen
			g.setColor(new Color(r.nextInt(256), r.nextInt(256),
r.nextInt(256)));
			//Farbiges Dreieck im gößeren Dreieck zeichnen
			g.fillPolygon(x2, y2, x2.length);
			//Für jedes der drei neu entstandenen Dreiecke:

			//unten links im Eck zeichnen
			drawTriangleRecursive(p1, p4, p6, g);
			//oben im Eck zeichnen
			drawTriangleRecursive(p4, p2, p5, g);
			//unten rechts im Eck zeichnen
			drawTriangleRecursive(p6, p5, p3, g);
		}
}
```

Class Main:

```
public class Main {

	public static void main(String[] args) {
		new Draw();
	}
}
```