Lab 09: Recursive Triangles

Lab exercises

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?**

   In the beginning, we took a different approach, but after talking to Prof. Weber-Wulff in the Lab, we realized that we need to use Window in Java.

   We started by creating a new class that extends JPanel and then added two fields for JFrame and JWindow. To create our constructor, we searched JWindow in the Java API.
   https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/JWindow.html

   Constructor Summary

   Constructors

Constructor	Description
`JWindow()`	Creates a window with no specified owner.
`JWindow(Frame owner)`	Creates a window with the specified owner frame.

   We then created the constructor and passed our frame to JWindow.

```
26    public Window2() {
27        frame = new JFrame();
28        window = new JWindow(frame);
```

   To find our screen size, we found the following solution:
   https://stackoverflow.com/questions/6777135/java-jframe-size-according-to-screen-resolution

```
static Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
```

   With this Dimension d we could now, set our window size and also calculate the largest equilateral triangle.
   After that we created a new method makeWindow, which we will use in our main method to create the window where we can draw. We got the idea for this method from this example:

http://www.java2s.com/Code/JavaAPI/java.awt/GraphicsdrawLineintx1inty1intx2inty2.htm

```
38    public void makeWindow() {
39        frame.getContentPane().add(new Window2());
40        frame.setPreferredSize(d);
41        frame.setVisible(true);
42        frame.pack();
43    }
```

In the main method we created a new Window object and then call the method makeWindow.

```
21    public static void main(String[] args) {
22        Window2 window = new Window2();
23        window.makeWindow();
24    }
```

We ran the program to see if it works and a new window opened.

To find the largest equilateral triangle we can draw, we need to find the side length. We used that formula to calculate the side, where h is our dimension height.

$$h = \frac{\sqrt{3}}{2} \cdot a$$

https://www.mathportal.org/calculators/plane-geometry-calculators/equilateral-triangle-calculator.php

Since we already set our dimension to the screen size, we can call d.height. In our program we then wrote the following to calculate the side length.

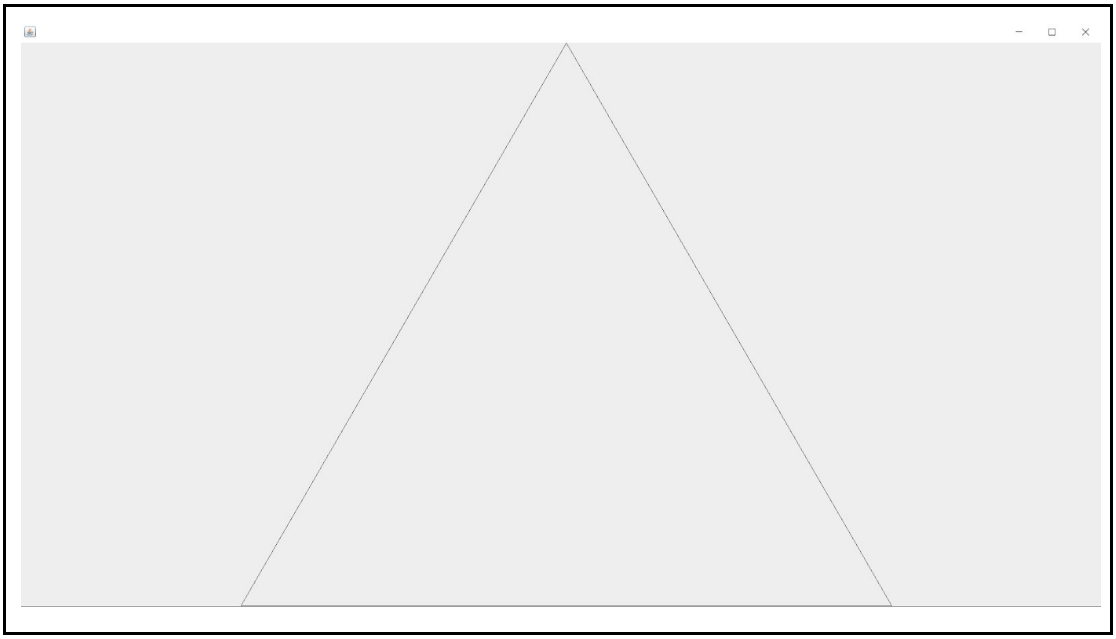
```
46          sidelength = (int) (d.height / (Math.sqrt(3) / 2));
```

To be able to draw in our window we looked at this example again: http://www.java2s.com/Code/JavaAPI/java.awt/GraphicsdrawLineintx1inty1intx2inty2.htm

We created a new method paintComponent and passed Graphics g as parameter. We also created a new int variable called side.

```
45      public void paintComponent(Graphics g) {
46          sidelength = (int) (d.height / (Math.sqrt(3) / 2));
47          int side = (int) (sidelength / 2 * Math.sqrt(3));
48          int[] point1 = { d.width / 2, 0 };
49          int[] point2 = { d.width / 2 - sidelength / 2, side };
50          int[] point3 = { d.width / 2 + sidelength / 2, side };
56  //      Left side, but starts from top to bottom
57          g.drawLine(point1[0], point1[1], point2[0], point2[1]);
58
59  //      Bottom side, starts from left to right
60          g.drawLine(point2[0], point2[1], point3[0], point3[1]);
61
62  //      Right side, starts from top to bottom right
63          g.drawLine(point1[0], point1[1], point3[0], point3[1]);
```

To draw our triangle we used drawLine method and drew three lines. We tested our program and got the following result.



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?**

   First of all we had to figure out how to calculate the midpoint of a line. We found an equation on this site:
   https://www.dummies.com/education/math/trigonometry/how-to-find-the-midpoint-of-a-line-segment/

   We decided to write a method that takes four ints, which calculate the midpoint of the line, using the formula from the linked page above. The return of the method is an array of ints, with the first position being the x-coordinate and the second position being the y-coordinate.

```
private int[] findMidpoint(int x1, int y1, int x2, int y2) {
    int[] midpoint = new int[2];
    midpoint[0] = (x1 + x2) / 2;
    midpoint[1] = (y1 + y2) / 2;
    return midpoint;
}
```

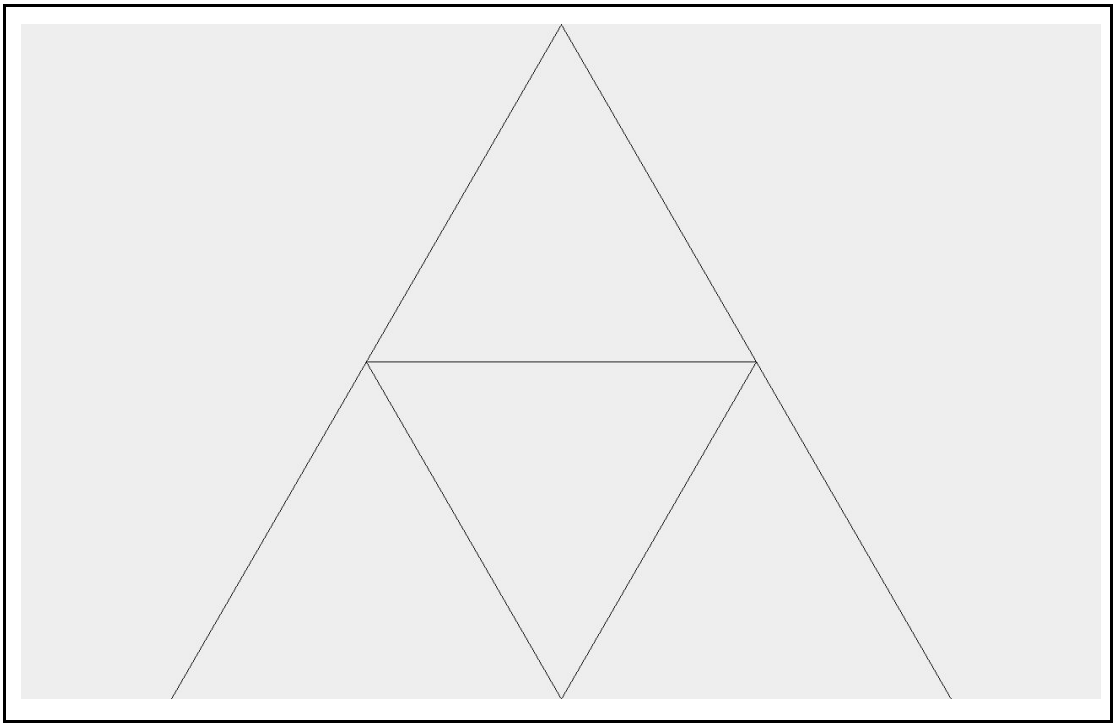
   Then we decided to write a method, that draws a triangle from the midpoints of a larger triangle, whose coordinates are given as a parameter.

```
public void makeTriangle(Graphics g, int[] pointA, int[] pointB, int[] pointC, int counter) {
    can sidelength be more than 800?

int[] newPointA = findMidpoint(pointA[0], pointA[1], pointB[0], pointB[1]);
int[] newPointB = findMidpoint(pointB[0], pointB[1], pointC[0], pointC[1]);
int[] newPointC = findMidpoint(pointA[0], pointA[1], pointC[0], pointC[1]);

g.drawLine(newPointA[0], newPointA[1], newPointB[0], newPointB[1]);
g.drawLine(newPointB[0], newPointB[1], newPointC[0], newPointC[1]);
g.drawLine(newPointA[0], newPointA[1], newPointC[0], newPointC[1]);
```

This was the result:



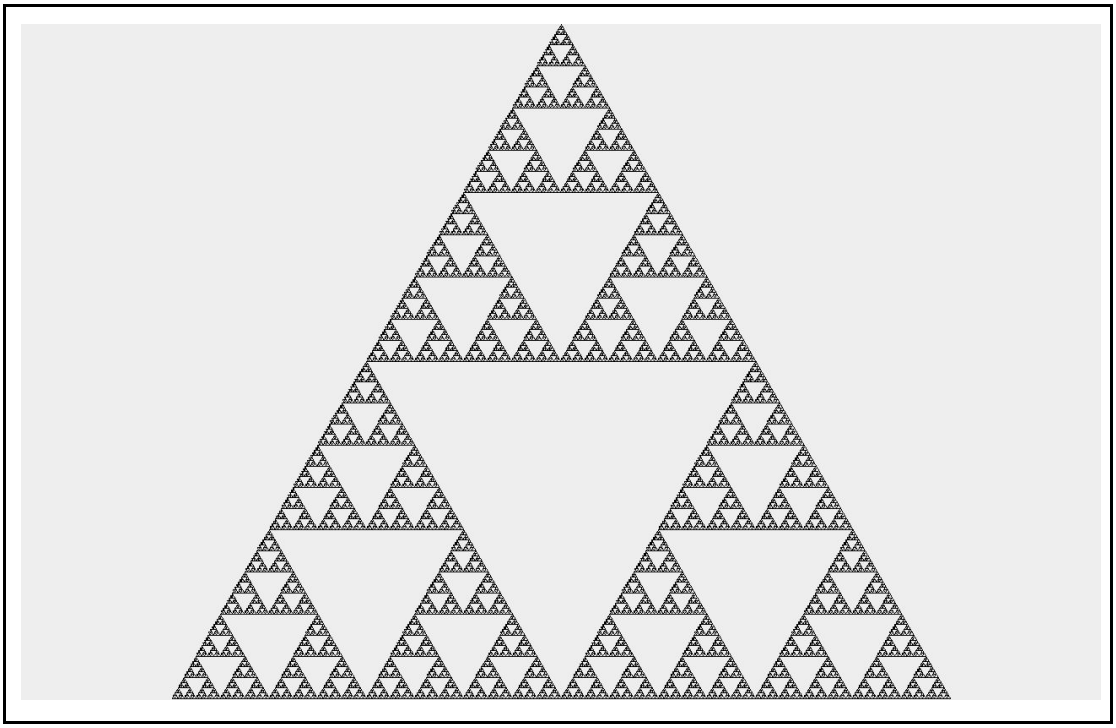
To make this method recursive, we had to decide on a termination condition and a measure. We chose the sidelength of the triangle as our measure and said that if the length is smaller than 10, we would terminate our recursive method. For this we had to add a statement to our method makeTriangle that calculates the sidelength of the current triangle, given in the parameters. The distance between two points is calculated like this:

$$\sqrt{(x2 - x1)\hat{}2 + (y2 - y1)\hat{}2}$$

```
sidelength = (int) Math.sqrt(Math.pow((pointA[0] - pointB[0]), 2) + Math.pow((pointA[1] - pointB[1]), 2));
```

Now we had a measure for our termination condition and we could write our recursive calls. So we wrote an if-statement, that said if the sidelength is larger than 10, then we call makeTriangle. We have to call makeTriangle three times, as we want to create three new triangles, one in each outer triangle. It took us some tries, to find out the right combination of points for makeTriangle, to make the correct pattern. We give the coordinates for each of the outer triangles and makeTriangle then draws a new triangle from the midpoints of the given triangle.

```
if (sidelength > 10) {
    makeTriangle(g, pointA, newPointA, newPointC, counter);
    makeTriangle(g, pointB, newPointB, newPointA, counter);
    makeTriangle(g, pointC, newPointC, newPointB, counter);
}
```



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

   We first looked up the Java API and found java.awt Color. After reading it, we now understood the different ways we can create and use colors in Java.
   https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/Color.html

   We created a new array of colors and used RGB values for the colors.

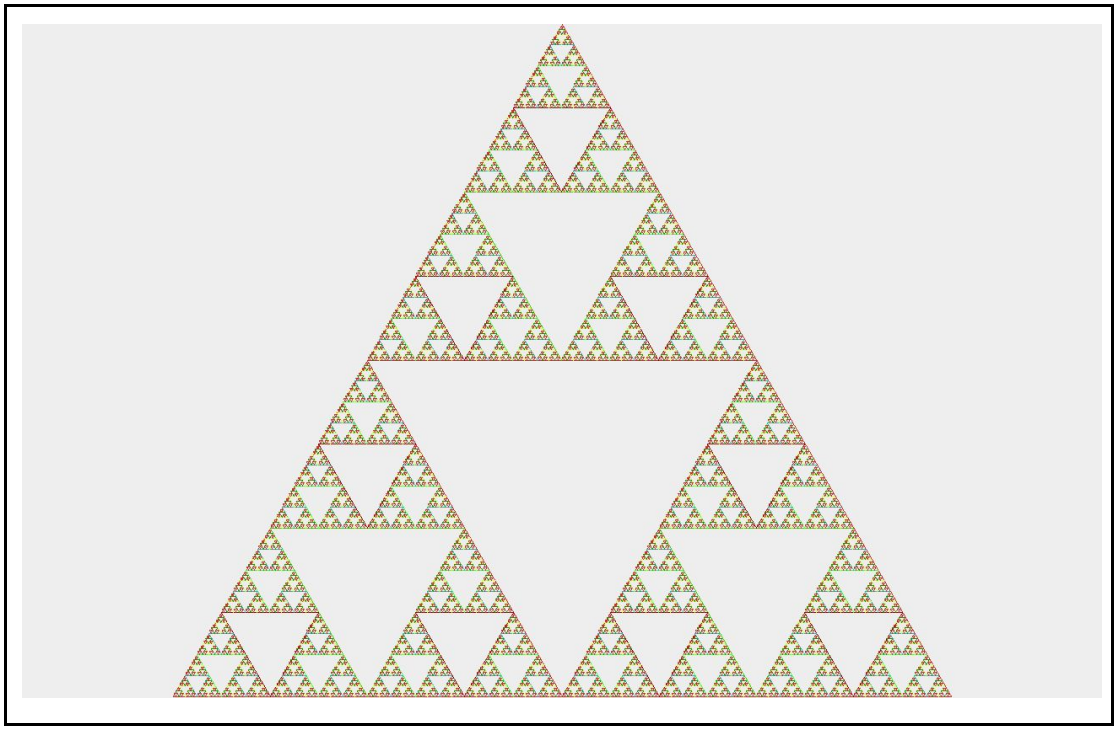
```
16
17  Color[] colorArray = { new Color(205,50,50), new Color(205,38,38), new Color(50,205,50), new Color(139,0,0),
18          new Color(0,255,100), new Color(32,178,170), new Color(205,255,112), new Color(34,139,34) };
19
```

In our makeTriangle method, we added the following code.

```
73    public void makeTriangle(Graphics g, int[] pointA, int[] pointB, int[] pointC, int counter) {
74
75        if (counter < colorArray.length - 1) {
76            counter++;
77        } else {
78            counter = 0;
79        }
80        g.setColor(colorArray[counter]);
```

g.setColor sets the color for the Graphic, any object drawn in that graphic will have the color g was set to.
Now whenever makeTriangle is called we set g.setColor to the next color in the array, by using a counter that is incremented each time. If counter gets bigger than colorArray.length-1, then we set counter back to 0.

Now that we had different colors in our array, we wanted to see what our triangle would look like.



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 used g.fillPolygon with the three points of the triangle being our new points. fillPolygon fills the given polygon with the colour g is currently set

to. As we already set the colours for the lines of each level, we didn't need any further code, to fill the polygon.

https://stackoverflow.com/questions/10839988/i-am-trying-to-fill-a-triangle

```
g.fillPolygon(new int [] {newPointA [0], newPointB [0], newPointC [0]},
        new int [] {newPointA [1], newPointB [1], newPointC [1]}, 3);
```

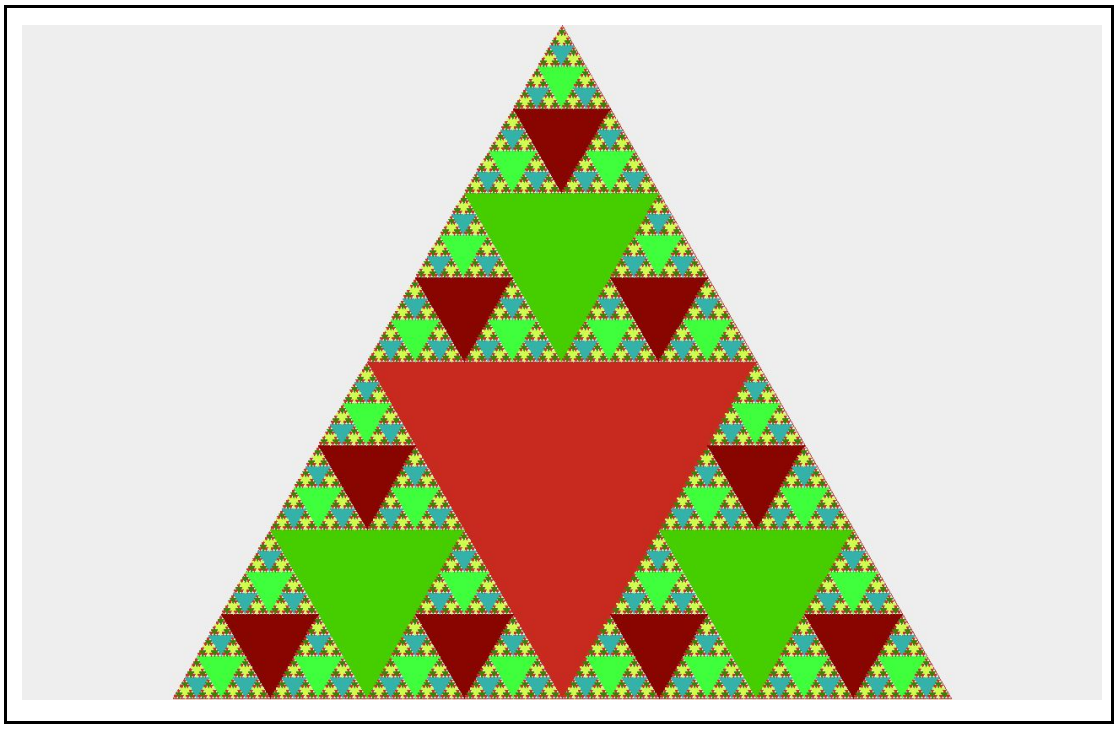
As we had already set our original triangle to the size of our window with a Dimension d, we now only had to figure out how to catch a resizing event for our window and how then to redraw the triangle with our new window size.

To resize the window, and to set the position and the size of the triangle accordingly, we needed an EventListener. After some research we found this and thought it would be the best solution:
https://stackoverflow.com/questions/2303305/window-resize-event

Now we only had to decide what componentResized() was supposed to do. As we said before, we needed a way to get the current window size. After some research in the Java API, we found out that Component has a method getSize(), which returns the size of the component as a Dimension. As JFrame extends Component, we could use this method for our JFrame. So we said, that when the window is resized, our Dimension d is set to frame.getSize(). And then we call repaint(), to make a new triangle.

```
frame.addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent e) {
        d = frame.getSize();
        System.out.println("new height: " + d.height + ", new width: " + d.width);
        repaint();
    }
});
```

This worked, but we noticed that the full triangle is only shown if the width of our window is larger than the height of our window, as we calculate the size of triangle only depending on the height of our window and not its width.



Evaluation

Marie:

I think for me the hardest part about this exercise was to figure out how to make a window in which we can draw a triangle. After we had figured this out, I thought it was much easier to calculate the triangle.

Pavel:

In this exercise i learned about the Window in Java and the getScreenSize method. It was interesting to see how the triangles were drawn when we used recursion in our method. After that we could also play around and see how it looks with more or less triangles and set different colors. In the last exercise I also learned about the componentResized method.

Pre Lab

Marie:

1. What exactly is an equilateral triangle? Can you write a class that draws a triangle? What data do you need to know in order to put a triangle at a particular position on the screen?

   → equilateral triangle: all sides have the same length, angles are 60°

   → triangle class: make a JFrame, create a Panel that has a canvas, draw triangle by creating a polygon with point for the middle of the horizontal line and the endpoint of the horizontal line, also y point

   → also possible to draw lines with drawLine (Graphics)

   → need to know the x and y position on the screen

2. What is the mathematical formula for finding the midpoint of a line segment that connects two Points?

   → new xPosition: xPosition1 + ((xPosition2 - xPosition1)/2)

   → new yPosition: yPosition1 + ((yPosition2 - yPosition1)/2)

3. What is the resolution of your computer screen? How can you find out? What is the largest ~~isosceles~~ equilateral triangle that you can show on a screen with this resolution?

   → 1280x800, checked about mac and internet apple support

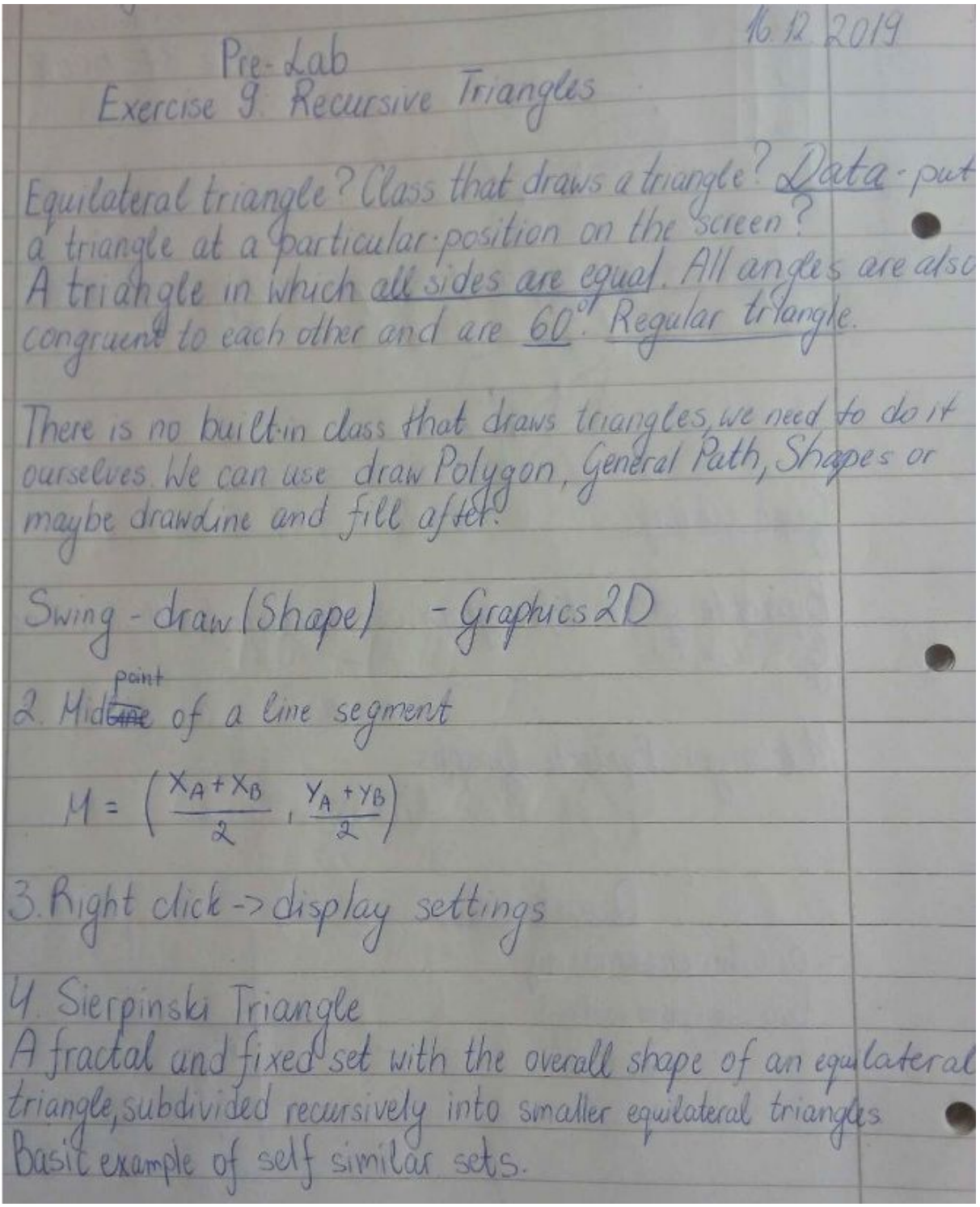
   → 800x800x800

4. Briefly describe what a **Sierpinski Triangle** is.

   → wikipedia: https://en.wikipedia.org/wiki/Sierpiński_triangle

   → a triangle that consists of four smaller triangles, with the three outer triangles, consisting of four smaller triangles and so on...

Pavel:



16.12.2019

Pre-Lab
Exercise 9: Recursive Triangles

Equilateral triangle? Class that draws a triangle? Data - put a triangle at a particular position on the screen?
A triangle in which all sides are equal. All angles are also congruent to each other and are 60°. Regular triangle.

There is no built-in class that draws triangles, we need to do it ourselves. We can use drawPolygon, GeneralPath, Shapes or maybe drawLine and fill after.

Swing - draw(Shape) - Graphics2D

2. Mid~~line~~ paint of a line segment

$$M = \left(\frac{X_A + X_B}{2}, \frac{Y_A + Y_B}{2}\right)$$

3. Right click -> display settings

4. Sierpinski Triangle
A fractal and fixed set with the overall shape of an equilateral triangle, subdivided recursively into smaller equilateral triangles
Basic example of self similar sets.

Code

```
import java.awt.Color;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Toolkit;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JWindow;

public class Window2 extends JPanel {
	static Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
	int sidelength;
	Color[] colorArray = { new Color(205,50,50), new Color(205,38,38), new
Color(50,205,50), new Color(139,0,0),
			new Color(0,255,100), new Color(32,178,170), new
Color(205,255,112), new Color(34,139,34) };
	JFrame frame;
	JWindow window;

	public static void main(String[] args) {
		Window2 window = new Window2();
		window.makeWindow();
	}

	public Window2() {
		frame = new JFrame();
		window = new JWindow(frame);
		frame.addComponentListener(new ComponentAdapter() {
			public void componentResized(ComponentEvent e) {
				d = frame.getSize();
//				System.out.println("new height: " + d.height + ", new
width: " + d.width);
				repaint();
			}
		});
	}

	public void makeWindow() {
		frame.getContentPane().add(new Window2());
		frame.setPreferredSize(d);
		frame.setVisible(true);
		frame.pack();
```

```
	}

	public void paintComponent(Graphics g) {
		sidelength = (int) (d.height / (Math.sqrt(3) / 2));
		int side = (int) (sidelength / 2 * Math.sqrt(3));
		int[] point1 = { d.width / 2, 0 };
		int[] point2 = { d.width / 2 - sidelength / 2, side };
		int[] point3 = { d.width / 2 + sidelength / 2, side };
		g.setColor(colorArray[0]);

//		int side = (int) (sidelength/2 * Math.sqrt(3)); // 692 , didn't know how
to
//		name this variable lol

//		Left side, but starts from top to bottom
		g.drawLine(point1[0], point1[1], point2[0], point2[1]);

//		Bottom side, starts from left to right
		g.drawLine(point2[0], point2[1], point3[0], point3[1]);

//		Right side, starts from top to bottom right
		g.drawLine(point1[0], point1[1], point3[0], point3[1]);

//		g.fillPolygon(new int [] {point1 [0], point2 [0], point3 [0]}, new int []
{point1 [1], point2 [1], point3 [1]}, 3);

		makeTriangle(g, point1, point2, point3, 0);

	}

	public void makeTriangle(Graphics g, int[] pointA, int[] pointB, int[]
pointC, int counter) {
//		can sidelength be more than 800?

		if (counter < colorArray.length - 1) {
			counter++;
		} else {
			counter = 0;
		}
		g.setColor(colorArray[counter]);
		sidelength = (int) Math.sqrt(Math.pow((pointA[0] - pointB[0]), 2) +
Math.pow((pointA[1] - pointB[1]), 2));

		int[] newPointA = findMidpoint(pointA[0], pointA[1], pointB[0],
pointB[1]);
		int[] newPointB = findMidpoint(pointB[0], pointB[1], pointC[0],
pointC[1]);
		int[] newPointC = findMidpoint(pointA[0], pointA[1], pointC[0],
pointC[1]);
```

```
                g.drawLine(newPointA[0], newPointA[1], newPointB[0],
newPointB[1]);
                g.drawLine(newPointB[0], newPointB[1], newPointC[0],
newPointC[1]);
                g.drawLine(newPointA[0], newPointA[1], newPointC[0],
newPointC[1]);

                g.fillPolygon(new int [] {newPointA [0], newPointB [0], newPointC
[0]},
                                new int [] {newPointA [1], newPointB [1], newPointC
[1]}, 3);

                if (sidelength > 10) {
                        makeTriangle(g, pointA, newPointA, newPointC, counter);
                        makeTriangle(g, pointB, newPointB, newPointA, counter);
                        makeTriangle(g, pointC, newPointC, newPointB, counter);
                }

        }

        private int[] findMidpoint(int x1, int y1, int x2, int y2) {
                int[] midpoint = new int[2];
                midpoint[0] = (x1 + x2) / 2;
                midpoint[1] = (y1 + y2) / 2;
                return midpoint;
        }

}
```