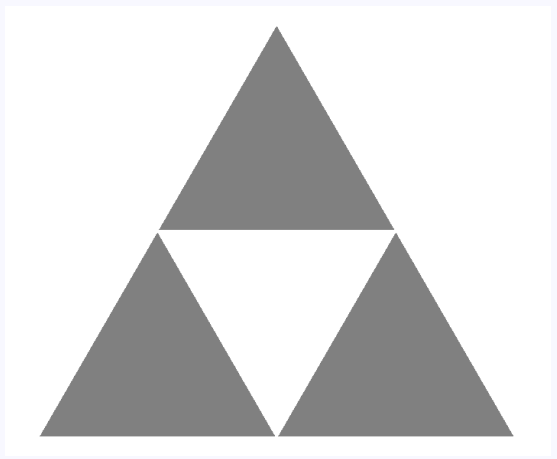
Model 1 Sierpiński Triangle

Open *Triangles.java* and run the program. Then change `LEVELS` to 1 and run the program again. Observe other `LEVELS` from 2 to 5. Adjust the `DELAY` in *Drawing.java*, as needed, to see the full drawing. Then answer the questions below to explore and discuss the source code as a team.

Drawing (cropped)



Console output

```
Starting tri (88, 570), (400, 30), (712, 570)
    Starting tri (88, 570), (244, 300), (400, 570)
    Finished tri (88, 570), (244, 300), (400, 570)
    Starting tri (244, 300), (400, 30), (556, 300)
    Finished tri (244, 300), (400, 30), (556, 300)
    Starting tri (400, 570), (556, 300), (712, 570)
    Finished tri (400, 570), (556, 300), (712, 570)
Finished tri (88, 570), (400, 30), (712, 570)
```

Questions (15 min) Start time:

1. How many times is the `tri()` method called...

a) in the source code?

b) when `LEVELS = 0`?

c) when `LEVELS = 1`?

d) when `LEVELS = 2`?

2. Consider the vertices in the drawing above (when `LEVELS == 1`). Using the boxes below, indicate the location of `p1`, `p2`, `p3`, `p4`, `p5`, and `p6`. *Hint:* see Lines 49–51 and 71–73 of the code.

3. When the `tri()` method calls itself, what value does it pass for `level`?

4. What prevents the recursive process from continuing forever?

5. When starting the drawing with higher `LEVELS`, what do the blue outlines represent?

6. Compare the `VeeTree` program from **??** with `Triangles`. In terms of recursion, what do they do in common? How are they different?