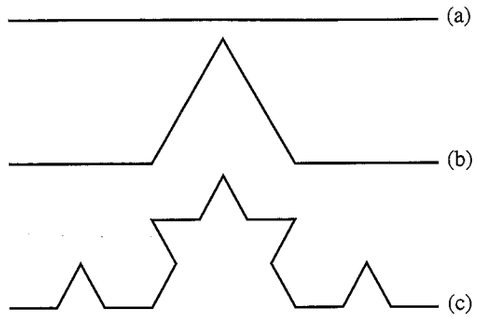

Figure 13.5 The first three stages (a)–(c) of the generation of a self-similar Koch curve. At each stage the displacement of the middle third of each segment is in the direction that increases the area under the curve. The curves were generated using `KochApp`. The Koch curve is an example of a continuous curve for which there is no tangent defined at any of its points. The Koch curve is self-similar on each length scale.

```
        frame.setVisible(true);
    }

    public void iterate(double x1, double y1, double x2, double y2,
            int n, DrawingPanel panel, Graphics g) {
        // draw Koch curve using recursion
        if(n>0) {
            double dx = (x2-x1)/3;
            double dy = (y2-y1)/3;
            double xOneThird = x1+dx; // new end at 1/3 of line segment
            double yOneThird = y1+dy;
            double xTwoThird = x1+2*dx; // new end at 2/3 of line segment
            double yTwoThird = y1+2*dy;
            // rotates line segment (dx, dy) by 60 degrees and adds to
            // (xOneThird,yOneThird)
            double xMidPoint = (0.5*dx-0.866*dy+xOneThird);
            double yMidPoint = (0.5*dy+0.866*dx+yOneThird);
            // each line segment generates 4 new ones
            iterate(x1, y1, xOneThird, yOneThird, n-1, panel, g);
            iterate(xOneThird, yOneThird, xMidPoint, yMidPoint, n-1,
                panel, g);
            iterate(xMidPoint, yMidPoint, xTwoThird, yTwoThird, n-1,
                panel, g);
            iterate(xTwoThird, yTwoThird, x2, y2, n-1, panel, g);
        } else {
            int ix1 = panel.xToPix(x1);
            int iy1 = panel.yToPix(y1);
            int ix2 = panel.xToPix(x2);
            int iy2 = panel.yToPix(y2);
            g.drawLine(ix1, iy1, ix2, iy2);
        }
    }

    public void draw(DrawingPanel panel, Graphics g) {
        iterate(0, 0, 500, 0, n, panel, g);
    }
```


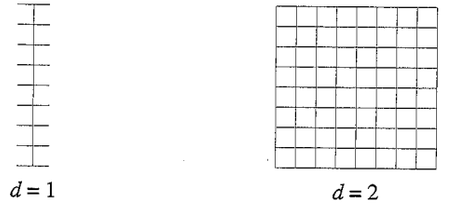

Figure 13.6 Examples of one-dimensional and two-dimensional objects.

```
    public void reset() {
        control.setValue("Number of iterations", 3);
    }

    public static void main(String args[]) {
        CalculationControl.createApp(new KochApp());
    }
}
```

How can we determine the fractal dimension of the Koch and similar mathematical objects? There are several generalizations of the Euclidean dimension that lead naturally to a definition of the fractal dimension (see Section 13.5). Here we consider a definition based on counting boxes. Consider a one-dimensional curve of unit length that has been divided into N equal segments of length ℓ so that $N = 1/\ell$ (see Figure 13.6). As ℓ decreases, N increases linearly, which is the expected result for a one-dimensional curve. Similarly, if we divide a two-dimensional square of unit area into N equal subsquares of length ℓ, we have $N = 1/\ell^2$, the expected result for a two-dimensional object (see Figure 13.6). In general, we have $N = 1/\ell^D$, where D is the fractal dimension of the object. If we take the logarithm of both sides of this relation, we can express the fractal dimension as

$$D = \frac{\log N}{\log(1/\ell)} \quad \text{(box dimension)}. \tag{13.8}$$

Now let us apply this definition to the Koch curve. Each time the length ℓ of our measuring unit is reduced by a factor of 3, the number of segments is increased by a factor of 4. If we use the size of each segment as the size of our measuring unit, then at the nth iteration we have $N = 4^n$ and $\ell = (1/3)^n$, and the fractal dimension of the triadic Koch curve is given by

$$D = \frac{\log 4^n}{\log 3^n} = \frac{n \log 4}{n \log 3} \approx 1.2619 \quad \text{(triadic Koch curve)}. \tag{13.9}$$

From (13.9) we see that the Koch curve has a fractal dimension between that of a line and a plane. Is this statement consistent with your visual interpretation of the degree to which the triadic Koch curve fills space?

Problem 13.4 The recursive generation of regular fractals

(a) Recursion is used in method `iterate` in `KochApp` and is one of the more difficult programming concepts. Explain the nature of recursion and the way it is implemented.