1 **import** edu.sjcny.gpv1.\*;

2 **import** java.awt.\*;

3

4 **public** **class** RecursiveFractal **extends** DrawableAdapter

5 {

6 **static** RecursiveFractal ge = **new** RecursiveFractal();

7 **static** GameBoard gb = **new** GameBoard(ge, "RECURSIVE FRACTAL");

8 **static** Point p1 = **new** Point(250, 70); **//vertices of the 1st triangle**

9 **static** Point p2 = **new** Point(25, 460);

10 **static** Point p3 = **new** Point(475, 460);

11

12 **public** **static** **void** main(String args[])

13 {

14 showGameBoard(gb);

15 }

16

17 **public** **void** draw(Graphics g)

18 {

19 g.setColor(Color.BLUE);

20 drawSierpinsky(8, p1, p2, p3, g);

21 }

22

23 **public** **static** Point midPoint(Point p1, Point p2)

24 {

25 Point midPoint = **new** Point();

26 midPoint.y = p1.y + (p2.y - p1.y)/2;

27 midPoint.x = p1.x + (p2.x - p1.x)/2;

28 **return** midPoint;

29 }

30

31 **public** **static** **void** drawSierpinsky(**int** iterations, Point p1, Point p2,

32 Point p3, Graphics g)

33 {

34 **if**(iterations == 0) **//base case**

35 {

36 **return**;

37 }

**38 //general solution**

39 g.drawLine(p1.x, p1.y, p2.x, p2.y); //draw a triangle

40 g.drawLine(p2.x, p2.y, p3.x, p3.y);

41 g.drawLine(p3.x, p3.y, p1.x, p1.y);

42 iterations--;

43 **//reduced problems to draw top, left & right side triangles recursively**

44 drawSierpinsky(iterations, p1, midPoint(p1,p2), midPoint(p1,p3), g);

45 drawSierpinsky(iterations, p2, midPoint(p2,p1), midPoint(p2,p3), g);

46 drawSierpinsky(iterations, p3, midPoint(p3,p1), midPoint(p3,p2), g);

47 }

48 }

**Figure 9.17 The application RecursiveFractal.**