Aufgabe 1

Die Methode baumRekursiv() zeichnet einen Baum. Beantworte dazu die folgenden Fragen:

```
01  void baumRekursiv(double laenge)
02  {
03     t.forward(laenge);
04     if (laenge > 2)
05     {
06        t.left(45);
07        baumRekursiv(laenge / 2);
08        t.right(90);
09        baumRekursiv(laenge / 2);
10        t.left(45);
11     }
12     t.back(laenge);
13  }
```

a) Was ist das Besondere bei einer rekursiven Methode?

b) Was muss man beachten, damit die Rekursion irgendwann endet („terminiert“) ?

c) Wenn man die Methode ohne den rekursiven Aufruf betrachtet:
Welche Position hat die Turtle am Ende der Methode?

```
void baumRekursiv(double länge)
{
   t.forward(länge);
   if (länge > 2)
   {
      t.left(45);
      // hier käme der rekursive Aufruf
      t.right(90);
      // hier käme der rekursive Aufruf
      t.left(45);
   }
   t.back(laenge);
}
```

d) Warum ist es wichtig, wo die Turtle am Ende der Methode steht?
Zeichne als Gegenbeispiel den Verlauf der Turtle für den rekursiven Aufruf `baumRekursiv(4)`, wenn die letzte Zeile `t.back(laenge)` fehlt.

Zu den Programmieraufgaben

Verwende die ausgeteilte BlueJ-Vorlage. Für jede Zeichnung gibt es eine Klasse mit einer rekursiven Methode. Die main-Methode ruft diese Methode auf und startet so die Rekursion.

Aufgabe 2 – Baum

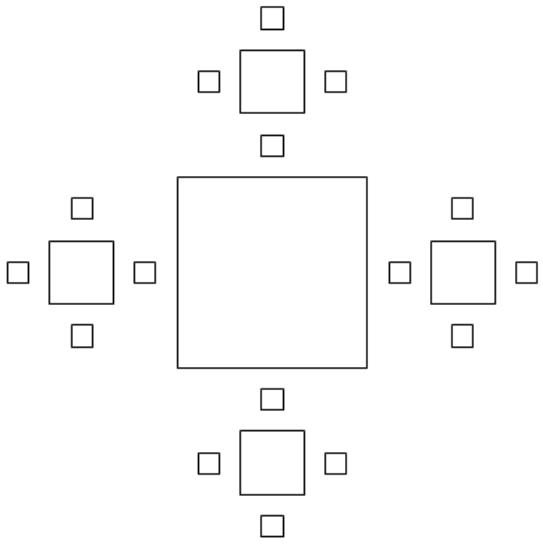
Implementiere die Methode, die den rekursiven Baum mit der Turtle zeichnet.

Tipp
Kürze die Länge der Äste in der Rekursion nicht um die Hälfte, sondern nur um ein Drittel, dann sieht der Baum wesentlich „üppiger“ aus.

Erweiterung
Wenn dein Programm funktioniert, füge eigene Ideen hinzu:
Unterschiedlich dicke Äste (z.B. unten dick, oben dünn), andere Winkel, mehr Äste, Farben, mehrere Bäume nebeneinander, zufällige Winkel oder Längen, usw.

Aufgabe 3 – Sierpinski-Teppich

Implementiere die Methode, die eine vereinfachte Form des Sierpinski-Teppichs zeichnet, wie abgebildet.

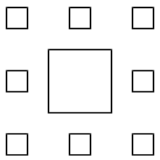


Rekursionsvorschrift
Die Grundform ist ein Quadrat. Es wird rekursiv in den vier Himmelsrichtungen die Grundform (Quadrat) mit ein Drittel Seitenlänge, im Abstand von einem Drittel der Seitenlänge zum größeren Quadrat gezeichnet.

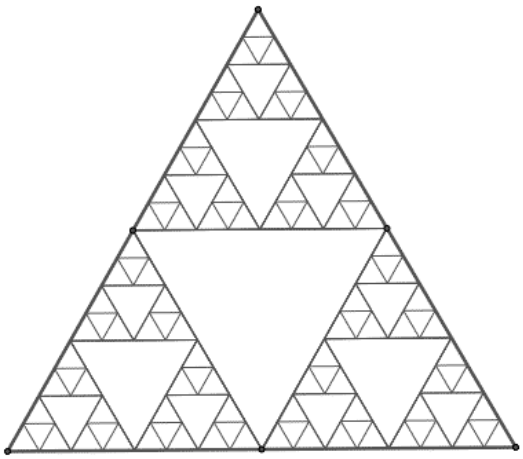
Tipp 1
Implementiere zunächst eine Hilfs-Methode, die ein Quadrat zeichnet. Am Ende der Methode sollte die Turtle wieder am Ausgangspunkt stehen.

Tipp 2
Zeichne zunächst auf kariertem Papier den Weg der Turtle zu den vier Stellen, wo die Rekursion beginnt, und wieder zurück zum Ausgangspunkt. Das macht die Programmierung viel einfacher.

Erweiterung
In jedem Rekursionsschritt werden nicht nur vier, sondern acht Quadrate gezeichnet, also auch in die vier Ecken.



Aufgabe 4 – Sierpinski-Dreieck

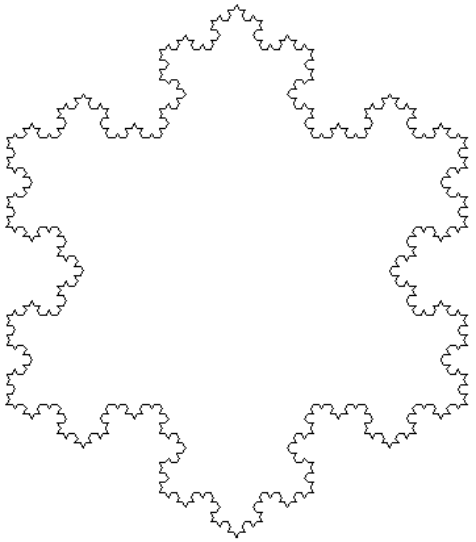


Die Methode für das Sierpinski-Dreieck benötigt Parameter für die drei Eckpunkte des Dreiecks. Außerdem ist hier ein Parameter für die „Tiefe“ der Rekursion hilfreich, um die Rekursion zu beenden.

Rekursionsvorschrift
Die Grundform ist ein Dreieck (es braucht nur im ersten Rekursionsschritt gezeichnet zu werden).
Bilde die Mittelpunkte der Seiten und zeichne ein Dreieck, das diese verbindet.
Wiederhole rekursiv für die drei neu entstandenen Dreiecke (links / oben / rechts).
Das Dreieck in der Mitte bleibt frei.

Tipp
Die Turtle-Methoden setPos(x,y) und moveTo(x,y) leisten bei dieser Aufgabe gute Dienste.

Aufgabe 5 – Koch-Kurve



Die Methode für die Koch-Kurve benötigt nur einen Parameter für die Länge der Grundlinie.

Rekursionsvorschrift
Die Grundform ist eine Linie. Sie wird in drei Teile geteilt: links, mitte, rechts. Zum mittleren Teil werden zwei Linien hinzugefügt, die mit dem mittleren Teil ein gleichseitiges Dreieck ergeben (d.h. die Linien sind ein Drittel so lang wie die Grundlinie, im Winkel von 60°).
Wiederhole viermal rekursiv für den linken und rechten Teil der Linie sowie für die beiden neu hinzugekommenen Linien (also nicht für den mittleren Teil der Linie).
Es wird nur auf der untersten Rekursionstiefe gezeichnet, also die ganz kurzen Linien.

Hinweis
Zeichne die entstandene Kurve dreimal, um die Form der Schneeflocke zu erhalten.
Drehe dazu einfach die Turtle nach der ersten Seite um 120° nach rechts und beginne mit der zweiten Seite, genauso dann für die dritte Seite.

Tipps zum Einsatz von Farben

Die Stiftfarbe der Turtle setzt man mit der Methode `setPenColor()`.
Farben können als Objekte der Klasse **Color** (d.h. im RGB-Format) angegeben werden:

```
Color c = new Color(1.0f, 0.0f, 0.0f);    // die Farbe rot
t.setPenColor(c);
```

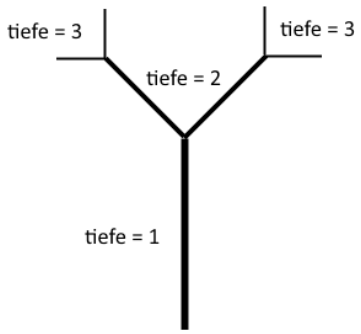
Statt die Farben einzeln zu setzen kannst du auch ein **Array von Farben** (eine „Palette“) erzeugen. Das hat mehrere Vorteile:

- Du kannst viele Farben auf einmal definieren, und du kannst Farben langsam von einer Farbe in eine andere übergehen lassen, z.B. von rot nach blau.
- Ein Array lässt sich in der Rekursion praktisch einsetzen.

Das folgende Beispiel zeigt, wie man eine Palette in einem rekursiven Programm einsetzt:

```
public class FarbBaum
{
   Turtle t;
   private Color[] palette;           // Array von Farben
   private int maxtiefe = 11;         // Die Anzahl der Farben ist auch
                                      // die maximale Rekursionstiefe
   public FarbBaum()
   {
      t = new Turtle();
      palette = new Color[maxtiefe];
      float r = 1.0f;
      float b = 0.0f;
      for (int i = 0; i < maxtiefe; i++)       // Erzeugt Farbverlauf
      {                                        // von rot nach blau
         palette[i] = new Color(r, 0.0f, b);
         r -= 0.1f;
         b += 0.1f;
      }
   }

   public void baumRekursiv(double länge, int tiefe)
   {
      t.setPenColor(palette[tiefe]);
      t.forward(laenge);
      if (tiefe < maxtiefe - 1)
      {
         t.left(45);
         baum(laenge/2, tiefe + 1);
         ...
      }
   }
}
```



Der Parameter „tiefe“ gibt an, wie weit der Ablauf in die Rekursion hineingegangen ist.
Zum Start wird baumRekursiv() mit tiefe = 0 aufgerufen. tiefe wird in jedem Rekursionsschritt um eins erhöht, bis zur maximalen Tiefe – 1 (die Farbindizes reichen von 0 bis maxtiefe – 1).