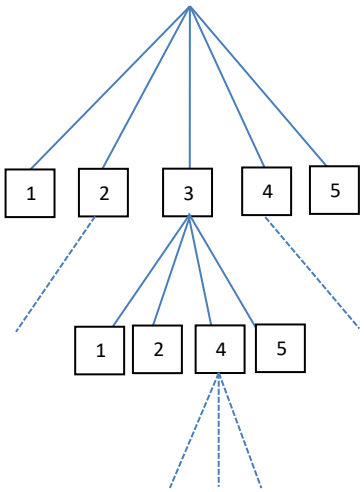
Projekt 1 : Rekursive Programmierung – Auf dem Kindergeburtstag

Aufgabe 1: Auf Jans Party sind vier weitere Jugendliche eingeladen. Am Partystisch stehen fünf Stühle. Wie viele Möglichkeiten gibt es, wie sich die Jugendlichen am Tisch auf die Stühle setzen können. Tipp: Jan setzt sich zuerst, er hat fünf Möglichkeiten. Als nächstes setzt sich Tim, für ihn bleiben noch vier Stühle frei. Wie viele Möglichkeiten gibt es, wie sich die beiden setzen können? Als nächstes kommt Freundin Anna hinzu. Sie hat bei jeder bisher in Frage kommenden öglichkeit jeweils drei Möglichkeiten. Zum Schluss setzen sich Greta und Daniel. Wie viele Möglichkeiten gibt es insgesmt? Gib eine Formel an, mit der man die Möglichkeiten berechnen kann.



Mit einem Baum kann man sich alle Möglichkeiten veranschaulichen, wie sich die Kinder setzen können. Die Zahl aller möglichen Pfade gibt die Zahl der Sitzmöglichkeiten an.

Aufgabe 2: a) Wie viele Sitzmöglichkeiten gibt es, wenn ein sechstes Kind hinzukommt? Verwende das Ergebnis von Aufgabe 1.
b) Unter der Fakultät einer natürlichen Zahl n versteht man das Produkt aller natürlichen Zahlen, die kleiner als diese Zahl sind. Also z. B.

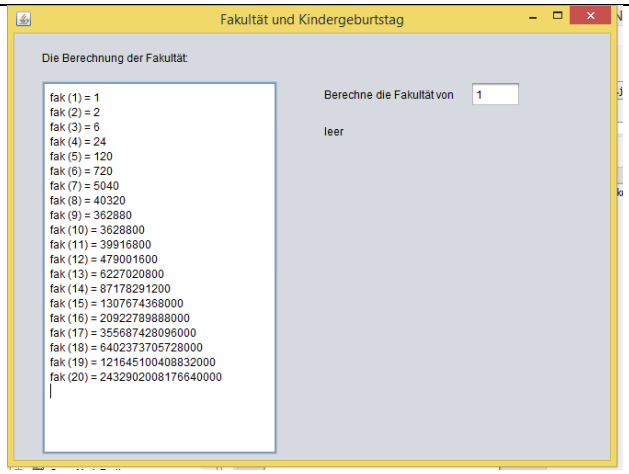
fak (5) = $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. Diese Definition der Fakultät nennt man **iterativ**, da mit einer einzigen Formel die Fakultät berechnet werden kann. In der Mathematik findet man für die Fakultät auch oft die Schreibweise 5 ! = fak (5).

Die Fakultät kann man auch **rekursiv** definieren, d. h. man bezieht sich auf Fakultät der vorhergehenden Zahl. Dies zeigt genau die Aufgabe 2a:
fak (5) = 5 · fak (4) oder allgemein für eine beliebige natürliche Zahl n: **fak (n) · = n · fak (n-1)**
Innerhalb einer Programmmethode spricht man bei dieser Formel auch von der **Rekursionsschleife**, da ähnlich wie in einer Schleife der Wert berechnet wird, indem er von einem vorhergehenden Wert zurückgeholt wird. Bei einer rekursiven Definition muss man zusätzlich einen **Rekursionsanker** (Startwert) definieren. In unserem Fall wäre dies **fak (1) = 1** : Setzt sich eine Person an einen Tisch mit nur einem Stuhl, gibt es dafür nur eine Möglichkeit.
Berechne mit Hilfe der rekursiven Definition fak (6) und fak (10).

Aufgabe 3: Importiere das Projekt Fakulät aus dem Schülerordner in dein persönliches Datenverzeichnis.

Das Programm soll in einer jTextaerea die Fakultäten der ersten 20 natürlichen Zahlen anzeigen. Außerdem kann in einem jTextField eine beliebige natütliche Zahl eingegeben werden, zu der die Fakultät unmittelbar angezeigt wird.
a) Gib an, wo in der folgende Methode, die die Fakultät einer Zahl berechnen soll, der Rekursionsanker und die Rekursionsschleife programiert wurde:

```
public long fak (long n){
    if (n == 1){
      return 1;
    }
    else{
      return n * fak (n-1);
    }
}
```



long ist ein Datentyp ähnlich wie int, mit dem natürliche Zahlen gespeichert werden können, der Bereich darstellbarer Zahlen ist jedoch größer als bei int.

b) Implementiere die angegebene Methode fak (long n) in der einzigen Klasse Berechnung. Mit dem Befehl

textArea.append („Hallo !“ + „\n“); kannst du innerhalb der TextArea eine Zeile einfügen, das Kommando **+ „\n“** bewirkt dabei einen Zeilensprung, so dass mit einem weiteren Befehl die nächste Zeile geschrieben werden kann. Implementiere mit Hilfe einer for-Schleife die Methode, die die Fakultäten der ersten 20 natürlichen Zahlen beechnet.

c) Die Methode **zahlenFeldKeyReleased** soll auf jede Veränderung innerhalb des jtextField **Zahlenfeld** reagieren. Der Methodenaufruf ist als Event-Methode bereits implementiert, in der ersten Zeile wird die aktueller Wert des Textfeldes umgeandelt in den Datentyp int. Vervollständige die Methode, so dass die Fakultät zur eingegebenen Zahl berechnet werden kann. Mit dem Befehl **ausgabeLab.setText („Hallo“);** kann im Label **ausgabeLab** ein beliebiger Text ausgegeben werden.

Zusatzaufgabe: Berechne mit Hilfe des Programm fak (60). Der vom Programm zurückgegebene Wert ist offenbar falsch, warum? Suche die kleinste natürliche Zahl, die zum ersten Mal einen falschen Wert liefert. Ändere in der Methode **fak (long n)** den Datentyp von **long** nach **int**. Untersuche, ab welcher natürlichen Zahl jetzt fehlerhafte Werte auftreten. Finde heraus, in welchen Zahlenbereichen natürliche Zahlen in den Datentypen **long** und **int** dargestellt werden können.

Projekt 2: Die Vermehrung von Kaninchen – Die Fibonacci-Zahlen

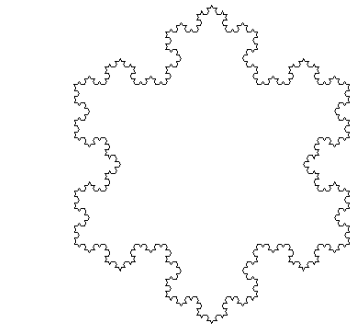
Auf einer einsamen Insel wird ein Kaninchenpaar ausgesetzt. Mit den Fibonacci-Zahlen kann man die Vermehrung der Kaninchen bei guten Nahrungsbedingungen simulieren. Die Fibonacci-Zahlen werden wie folgt definiert:

Rekursionsanker:	fib (1) = 1 und fib (2) = 1
Rekursionsschleife:	fib (n) = fib (n-1) + fib (n-2)

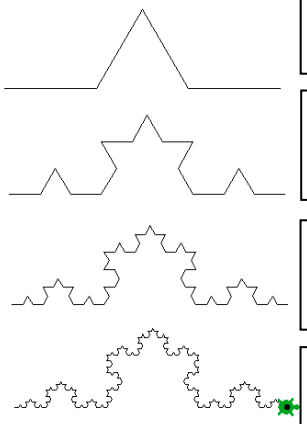
Aufgabe 1: Implementiere das Projekt Fibonacci aus dem Schülerverzeichnis in dein persönliches Datenverzeichnis. Das Projekt ist vollkommen analog aufgebaut zum Projekt Fakultät. Implementiere die neue Methode fib (long n) und sorge dafür, dass die Tabelle mit den ersten 20 Werten sowie die Reaktion auf die Eingabe einer Zahl im Textfeld wieder funktiomiert.

Projekt 3: Rekursive Grafiken mit Greenfoot – Die Schneeflocke

In diesem Projekt lernst du am Beispiel einer Schneeflocke, wie man mit Greenfoot rekursive Grafiken erzeugen kann.



Die fertige Schneeflocke



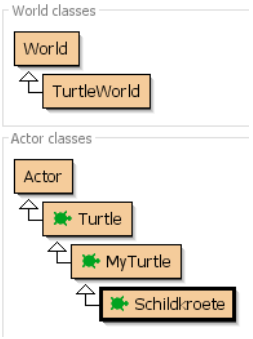
Grundlage der Schneeflocke:
Die Zackenkurve mit Länge 300, Stufe 1

Die Zackenkurve mit Länge 300, Stufe 2:
Die Zackenkurve wird 4 Mal mit Länge 100 gezeichnet, zwischendurch wird gedreht

Die Zackenkurve mit Länge 300, Stufe 3:
Die Zackenkurve wird 16 Mal mit Länge 33,3 (gerundet) gezeichnet

Die Zackenkurve mit Länge 300, Stufe 4:
Die Zackenkurve wird 64 Mal mit Länge 11,1 (gerundet) gezeichnet

Aufgabe 1: Kopiere in dein persönliches Verzeichnis aus dem Schülerverzeichnis das Greenfoot-Projekt Schneeflocke und starte es. Du kannst dazu im Greenfoot-Projekt auf die Projektdatei doppelklicken oder Greenfoot starten und über das Menü mit dem Befehl Scenario / Open das gesamte Projekt öffnen. Das Projekt enthält unter **Actor classes** eine sogenannte Turtle-Grafik. Damit meint man eine Schildkröte, die man auf dem Bildschirm positionieren und bewegen kann. Die Schildkröte hinterlässt entlang ihrer Bewegung eine Spur. Erzeuge durch Rechtsklick auf die Unterklasse Schildkröte durch Auswahl des Befehls **new Schildkroete()** eine Schildkröte und positioniere diese etwa in der Mitte des Bildschirms.



Klicke mit der rechten Maustaste auf die Schildkröte, es wird die Methode **quadrat (double laenge)** angezeigt. Führe diese Methode per Mausklick aus, gebe als Länge die Zahl 100 ein und beobachte, was gezeichnet wird. Durch Rechtsklick auf die Klasse Schildkröte kannst du dir den Quellcode ansehen.
Ergänze den Quellcode durch eine neue Methode Zackenkurve (double laenge), die die Zackenkurve der Stufe 1 (s. Grafik oben rechts) zeichnet, wobei die gesamte horizontale Länge der Zackenkurve über den Parameter laenge angegeben werden soll. **Hinweis 1:** Man teilt den Parameterwert laenge durch 3 und zeichnet insgesamt vier Teilstriche mit dieser Länge. Das Dreieck, von dem zwei Seitenlängen gezeichnet werden, ist gleichseitig, was den Winkel bestimmt. Drehungen im Uhrzeigersinn werden positivem Parameter, Drehungen gegen den Uhrzeigersinn mit negativem Parameter durchgeführt.
Probiere deine neue Methode aus. **Hinweis 2:** Die Schildkröte kann mit der Maus beliebig verschoben werden, die gesamte Zeichenfläche kann mit der Schaltfläche Reset neu gezeichnet werden.

Aufgabe 2: Die Zackenkurve der Stufe 2 kann man relativ einfach zeichnen, indem man die Methode Zackenkurve (double laenge) insgesamt viermal aufruft und dazwischen die Schildkröte jeweils um den richtigen Winkel dreht. Überlege, mit welcher Länge die Zackenkurve aufgerufen werden muss. Die neue Methode zackenkurve2 (double laenge) soll wiederum die Zackenkurve der Stufe 2 mit der gesamten horizontalen Länge zeichnen, die dem übergebenen Parameter laenge entspricht. Probiere deine neue Methode aus.

Aufgabe 3: Die Zackenkurve der Stufe 3 könnte man so zeichnen, dass man die Methode zackenkurve2 wiederum viermal nacheinander aufruft. Genau das kann man auch tun, indem man die Methode zackenkurve (double laenge) rekursiv aufruft. Dies hat den Vorteil, dass man mit einer einzigen Methode beliebig viele Stufen zeichnen kann. Die Abbruchbedingung für die Rekursion (Rekursionsanker) setzt man, indem man ab einer bestimmten Wert des übergebenen Parameters laenge keine weiteren rekursiven Aufrufe startet, sondern einfach eine Zackenkurve mit der richtigen Länge (laenge / 3) zeichnet. Die folgende Abbildung zeigt den grundlegenden Aufbau der rekursiven Methode superZackenkurve (double laenge), die die Zackenkurve mit der Stufe 5 zeichnet, es werden dabei die Längen (gerundet) 300, 100, 33.3, 11.1 und 3.7 benutzt:

```
public void superZackenkurve (double laenge){
  if (laenge > 10){
    move (laenge / 3); turn (-60);  ….
  }
  else{
     superZackenkurve (laenge/3); turn (-60);  ….
  }
}
```

a) Vervollständige die Methode, so dass die Zackenkurve der Stufe 5 rekursiv gezeichnet wird.
b) Wie muss die Zahl 10 verändert werden, damit eine Zackenkurve der Stufe 1, 2 oder 3 gezeichnet wird. Schreibe jeweils auf, in welchem Bereich die Zahl dazu liegen muss und erkläre, warum dies so ist.

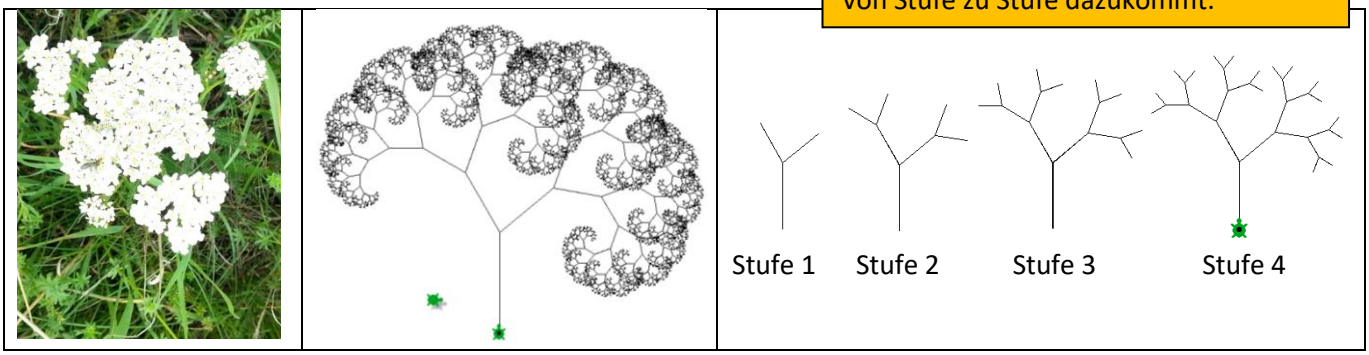
Zusatzaufgabe 1: Schreibe eine neue rekursive Methode **zackenkurveStufe (int Stufe, double laenge)**, die ein Zackenkurve der Stufe stufe mit der gesamten horizontalen Länge laenge zeichnet.

Zusatzaufgabe 2: Schreibe eine Methode schneeflocke (double laenge), die die gesamte Schneeflocke (s. Bild am Anfang des Kapitels) der Stufe 5 zeichnet. **Hinweis:** Die Methode superZackenkurve muss dazu dreimal aufgerufen werden.

Zusatzaufgabe 3: Zeichen 4 Schneeflocken um denselben Mittelpunkt mit verschiedenen Größen innerhalb einer neuen Methode. Finde dazu heraus, wie man den Zeichenstift unsichtbar machen kann. Du findest diese Information im Quellcode der Klasse Turtle.

Projekt 4: Die Scharfgabe

Tipp: Wahrscheinlich hilft es dir, wenn du jeweils farbig einzeichnest, was hier von Stufe zu Stufe dazukommt.



Die Schafgabe ist eine Heilpflanze, die ähnlich wie Kamille äußerlich angewendet werden kann. Sie wächst auf Wiesen und an Wegrändern und ist eigentlich einem jedem schon aus der frühen Kindheit bekannt.

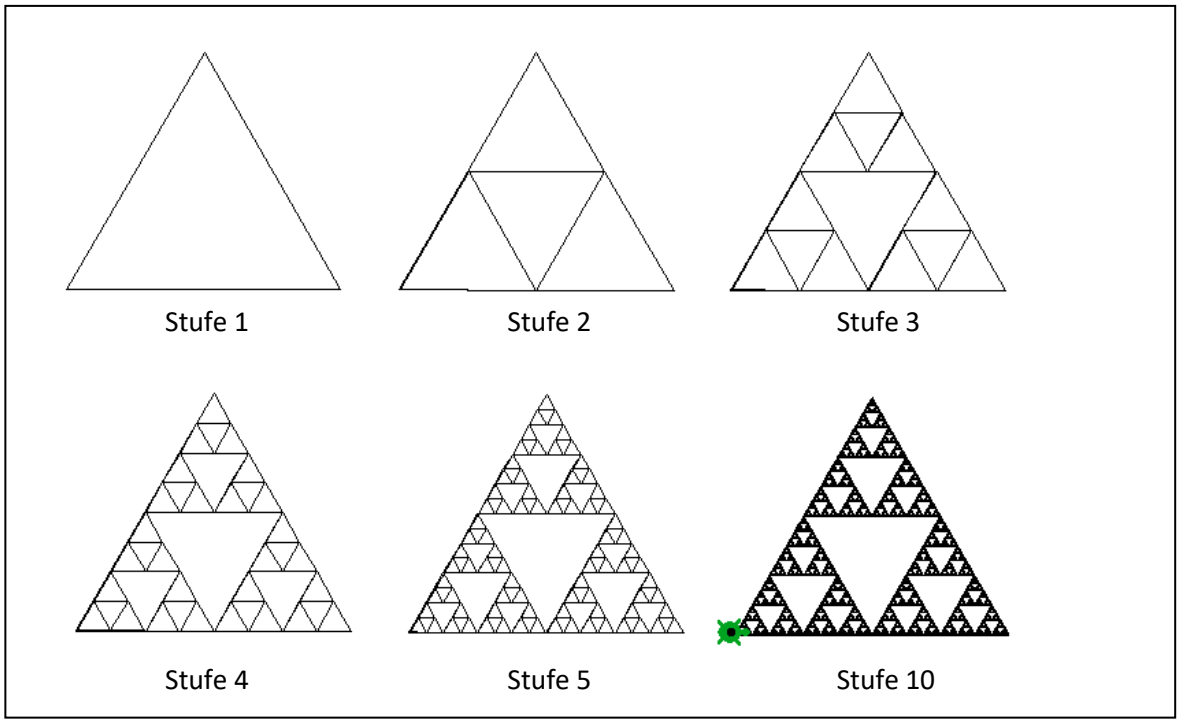
Aufgabe 1: Impementiere eine Methode **schafgabe (double laenge)**, die eine Schafgabe der Stufe 1 zeichnet. Winkel und Längen kannst du selber schätzen. Der erste vertikale Strich sollte die Länge laenge haben, die als Parameter übergeben wird. Die Schafgabe sollte mit der Anfangsrichtung Oben gezeichnet werden. **Hinweis1:** Da die Schildkröte nach ihrer Erzeugung stets nach rechts schaut, lohnt sich bei diesem Projekt die Implementierung der Methode **dreheLinks()**, die die Schildkröte um 90° gegen den Uhrzeigersinn dreht. **Hinweis 2:** Bei dieser Grafik ist es wichtig, dass die Schildkröte am Ende der Methode **schafgabe** wieder zum Ausgangspunkt zurückkehrt und in die Anfangsrichtung schaut, sonst werden die folgenden Aufgaben sehr schwierig bis unlösbar. Der Befehl **move (laenge)** kann auch mit negativem Parameter aufgerufen werden.

Aufgabe 2: Impementiere eine Methode **schafgabe2 (double laenge)**, die eine Schafgabe der Stufe 2 zeichnet. Die Methode **schafgabe (double laenge)** soll dabei zweimal aufgerufen werden.

Aufgabe 3: Implemetiere eine Methode **superSchafgabe (double laenge)**, die die gesamte Schafgabe rekursiv zeichnet. Denke an Rekursionsaker und Rekursinsschleife.

Zusatzaufgabe: Implemetiere die neue Methode **andereSchafgabe (int Stufe, double laenge)**, die die Pflanze so zeichnet, dass es nicht zu Überdeckung einzelner Äste kommt und die Schafgabe zur anderen Seite hin geneigt ist. Außerdem soll als weiterer Parameter die Stufe hinzukommen, bis zu der gezeichnet wird.

Projekt 5 : Dreiecke, Dreiecke, Superdreiecke …



Die Grafik zeigt eine rekursive Programmierung mit Dreiecken.

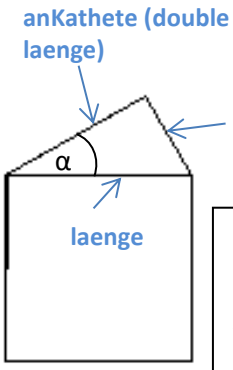
Aufgabe 1: Implementiere eine Methode **dreieck (double laenge)**, die ein gleichseitiges Dreieck der Kantenlänge **laenge** zeichnet. Die Schildkröte soll sich zum Beginn und am Ende jeweils in der unteren linken Ecke des Dreiecks befinden und nach Links ausgerichtet sein.

Aufgabe 2: Implementiere eine Methode **dreieck2 (double laenge)**, die obiges Dreieck der Stufe 2 mit der äußeren Kantenlänge **laenge** zeichnet. Die Methode soll dreimal die Methode **dreiecke (double laenge)** aufrufen und dabei jeweils ein Teildreieck mit kleinerer Länge zeichnen. Zwischen den Aufrufen muss die Schildkröte jeweils weiterbewegt und in die richtige Richtung gedreht werden. Zum Abschluss muss die Schildkröte wieder am Ausgangspunkt, d. h. in der linken unteren Ecke des Dreiecks sein und nach Links schauen.

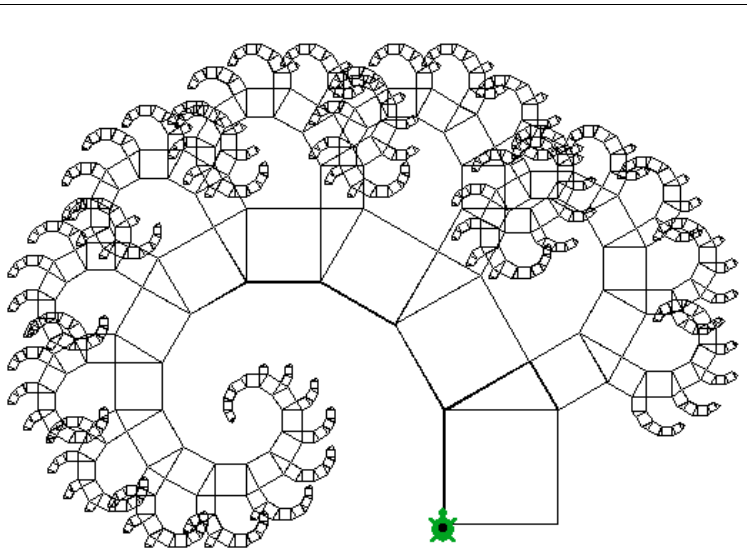
Aufgabe 3: Implementiere eine Methode **superDreieck (double laenge)**, die das fertige Dreieck rekursiv zeichnet. Als Rekursionsanker, reicht es aus, die Methode **dreieck (double laenge)** aufzurufen.

Projekt 6 : Rechtwinklige Dreiecke - Der Pythagorasbaum

Die rechte Grafik zeigt eine rekursive Grafik, deren Grundelement ein Quadrat ist, auf dem ein rechtwinkliges Dreieck aufgesetzt ist.

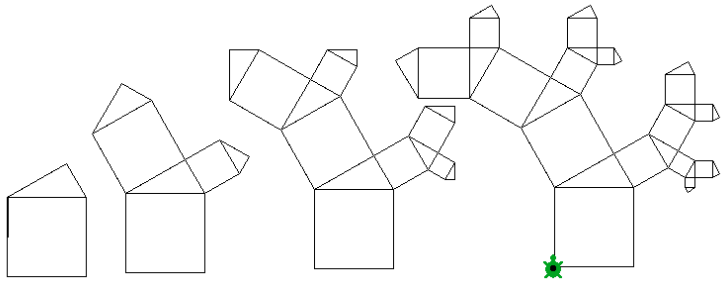


Mit den beiden angegebenen Methoden kann man die Länge von Ankathete und Gegenkathete im rechtwinkligen Dreieck berechnen (α = 30°).



Mit den beiden Methoden **anKathete (double laenge)** und **gegenKathete (double laenge)**, die in der Klasse Schildkroete bereits implementiert sind, kann man die Länge der Ankathete und Gegenkathete im Deieck berechnen, wenn die Länge der Hypothenuse über den Parameter laenge angegeben wird. Der Winkel α beträgt dabei 30° (s. Skizze auf der letzten Seite).

Aufgabe 1: Berechne die Länge von Ankathete und Gegenkathete im Dreieck mit der Hypothenusenlänge 10 und dem Winkel α = 30° und schreibe eine Rechnung dazu auf. Vergleiche mit dem Ergebnis der beiden Methoden. **Hinweis:** Die Methoden **sin (winkel)** und **cos (winkel)** findet man in der Klasse Math, es muss jedoch das Bogenmaß benutzt werden, deshalb wird der Winkel zunächst mit π / 180 multipliziert (z. B. 90° entspricht dem Bogenmaß 90° * π / 180° = π / 2.



Stufe 1 Stufe 2 Stufe 3 Stufe 4

Aufgabe 2: Implementiere die Methode **rechtwinkligesDreieck (double laenge)**, die ein rechtwinkliges Dreieck mit der Hypothenusenlänge laenge und dem Winkel α = 30° (linke, untere Ecke) zeichnet. Die Schildkröte soll vor und nach dem Zeichnen nach rechts zeigen, die Hypothenuse soll waagerecht liegen.

Aufgabe 3: Implementiere die Methode **quadratmitDreieck (double laenge)**, die zunächst ein Quadrat mit der Seitenlänge laenge und das aufgesetzte Dreieck (s. o. Stufe 1) zeichnet. Zu Beginn und am Ende der Methode soll sich die Schildkröte in der unteren linken Ecke des Quadrats befunden und nach Oben zeigen.

Aufgabe 4: Implementiere die Methode **quadratmitDreieck2 (double laenge)**, die die obige Figur der Stufe 2 zeichnet. Die Seitenlänge des Quadrats soll wieder als Parameter übergeben werden. Dabei soll die Methode quadratmitDreieck insgesamt dreimal aufgerufen werden. Überlege dazu, welchen Wert der Parameter der Seitenlänge jeweils haben muss. **Tipp:** Alle drei Werte sind unterschiedlich!

Aufgabe 5: Von der Methode **quadratmitDreieck2 (double laenge)** zur **Methode SuperQuadratmitDreieck (double laenge)**, die die gesamte Figur rekursiv zeichnen soll ist es nur ein kurzer Weg:

```
public void superQuadratmitdreieck (double laenge){
  if (laenge > 5){
    // zeichne das Grundquadrat: hier kein rekursiver Aufruf!
    …
    // zeichne das QuadratmitDreieck auf der Ankathete mit den weiteren kleineren Figuren darauf
    …   // rekursiver Aufruf der Methode superQuadratmitdreieck
    // zeichne das QuadratmitDreieck auf der Gegenkathete mit den weiteren kleineren Figuren darauf
    …   // rekursiver Aufruf der Methode superQuadratmitdreieck
   // bewege dich zurück zur unteren linken Ecke des Quadrats
  }
 }
```

Wie man sieht, kann man hier den Rekursionsanker weglassen, wenn man dafür sorgt, dass die Rekursionsschleife nur bis zu einer bestimmten Seitenlänge ausgeführt wird. Vervollstädige die Methode und probiere aus, ob der Pythagorasbaum komplett gezeichnet wird.

Zusatzaufgabe: a) Verändere die Methoden so, dass als zusätzlicher Parameter der Winkel des Deiecks α angegeben werden kann.
b) Wie groß ist die Summe aller Quadratflächen, die beim Pythagorasbaum der Stufe 10 gezeichnet werden, wenn als Parameter 80 (cm) übergeben wird.