

Grundlagen

<code>import ch.aplu.turtle.*;</code>	Bindet die Turtle-Bibliothek ein.
<code>Turtle tom;</code> <code>tom = new Turtle();</code>	Erzeugt ein Objekt der Klasse Turtle in einem neuen Fenster mit der Standardgröße 400 x 400 Pixel.
Um die Größe des Turtle-Fensters selbst anzugeben, erzeuge das Turtle-Fenster selbst: <code>TurtleFrame frame;</code> <code>Turtle tom;</code> <code>frame = new TurtleFrame("Titel", 800, 600);</code> <code>tom = new Turtle(frame);</code>	
Es können auch mehrere Turtle-Objekte für ein Fenster erzeugt werden: <code>tom = new Turtle(frame);</code> <code>tea = new Turtle(frame);</code>	
<code>void hideTurtle()</code>	Macht Turtle unsichtbar (Turtle zeichnet schneller).
<code>void showTurtle()</code>	Zeigt die Turtle wieder an.
<code>void penUp()</code>	Hebt den Zeichenstift (Turtle bewegt sich, ohne zu zeichnen).
<code>void penDown()</code>	Setzt Zeichenstift ab (Turtle zeichnet).
<code>void penWidth(int pWidth)</code>	Setzt die Breite des Zeichenstifts.
<code>void speed(double pSpeed)</code>	Setzt die Geschwindigkeit der Turtle. 1: langsam; > 1: schneller; -1: „unendlich“ Hat nur einen Effekt wenn die Turtle sichtbar ist.
<code>void setPenColor(String pColor)</code>	Legt die Stiftfarbe (d.h. die Farbe der Zeichnung) fest.
<code>void setColor(String pColor)</code>	Setzt die Farbe der Turtle (keine Wirkung auf die Zeichnung).
<code>void clear()</code>	Löscht die Zeichnung (Fenster ist anschließend weiß).
<code>void clear(String pColor)</code> <code>void clear(Color pColor)</code>	Übermalt das ganze Fenster mit der gegebenen Farbe.
Farben können als String angegeben werden, d.h. als Text in Anführungsstrichen: <code>teo.setPenColor("red");</code> Beispiele: red, blue, yellow, green, orange, purple, brown, magenta, cyan, navy, silver, gold, white, gray, black, ... Zur vollständigen Liste der Farbcodes siehe http://en.wikipedia.org/wiki/X11_color_names	
Farben können stattdessen auch mithilfe von der Klasse Color angegeben werden. So kann prinzipiell jede beliebige Farbe (im RGB-Format) definiert werden. Dazu muss man das Java-Package „awt“ (advanced window toolkit) einbinden: <code>import java.awt.*;</code> Zur Dokumentation siehe http://docs.oracle.com/javase/8/docs/api/java/awt/Color.html	

Zeichnen und Bewegung

Koordinatensystem der Turtle Der Mittelpunkt ist die Mitte des Turtle-Fensters Standardmaße: -200 bis 200 für x und y Anfangsposition der Turtle in der Mitte (0 / 0) Richtungswinkel 0° = oben, 90° = rechts, 180° = unten, 270° = links Zu Anfang schaut die Turtle nach oben (0°).	
<code>void forward(double distance)</code>	Bewegt Turtle vorwärts.
<code>void back(double distance)</code>	Bewegt Turtle rückwärts. Die Blickrichtung ändert sich nicht.
<code>void moveTo(double x, double y)</code>	Bewegt die Turtle an die Position (x, y) und zeichnet dabei. Dreht die Turtle dabei in die entsprechende Richtung.
<code>void setPos(double x, double y)</code>	Setzt Turtle auf die Position (x, y), ohne zu zeichnen. Die Richtung der Turtle ändert dabei sich nicht.
<code>void double getX() / getY()</code>	Gibt die Turtleposition zurück.
<code>void left(double winkel)</code> <code>void right(double winkel)</code>	Dreht Turtle um den gegebenen Winkel (in Grad) nach links, bzw. rechts (von der aktuellen Blickrichtung aus).
<code>void heading(double winkel)</code>	Setzt die Richtung der Turtle (egal wo sie vorher hinschaut).
<code>double heading()</code>	Gibt die aktuelle Richtung der Turtle zurück (in Grad).

Kreise / Bögen

<code>void dot(double durchmesser)</code>	Zeichnet einen gefüllten Kreis. Mittelpunkt ist der aktuelle Standort der Turtle.
<code>void openDot(double durchmesser)</code>	Zeichnet einen Kreis ohne Füllung.
<code>void leftCircle (double radius)</code> <code>void rightCircle(double radius)</code>	Zeichnet einen Kreis von der Position der Turtle aus (nach links bzw. nach rechts)
<code>void leftArc (double r, double w)</code> <code>void rightArc(double r, double w)</code>	Zeichnet einen Kreisbogen von der Position der Turtle aus (nach links bzw. nach rechts). Der Radius des Kreisbogens ist r, und die Turtle zeichnet einen Ausschnitt des Kreises von w Grad.

Flächen füllen

<code>void setFillColor(String color)</code>	Legt die Füllfarbe fest .
<code>void fill()</code>	Füllt eine geschlossene Figur, in der sich die Turtle befindet. Die Turtle darf nicht auf dem Rand, sondern muss innerhalb der Form stehen (also Stift anheben und hinein bewegen). Die Füllfarbe muss eine andere sein als die Hintergrundfarbe.
<code>void fill(double x, double y)</code>	Füllt eine geschlossene Form, in der sich (x, y) befindet.
<code>void fillToPoint(double x, double y)</code>	Füllt fortlaufend die gezeichnete Form vom Punkt (x, y) aus.
<code>void fillToHorizontal(double y)</code>	Füllt eine geschlossene Form bis zur einer horizontalen Linie.
<code>void fillToVertical(double x)</code>	Füllt eine geschlossene Form bis zur einer vertikalen Linie.

Sonstiges

<code>double distance(double x, double y)</code>	Gibt die Entfernung der Turtle zum Punkt (x, y) zurück.
<code>Color getPixelColor()</code>	Gibt die Farbe des Fensters an der Position der Turtle zurück.
<code>void label(String text)</code>	Schreibt einen Text an der aktuellen Position der Turtle.
<code>void setFont(Font font)</code>	Setzt die Schriftart für <code>label()</code> (frage ggf. die Lehrkraft nach einem Beispiel).
<code>void setFontSize(int size)</code>	Setzt die Schriftgröße für <code>label()</code> .
<code>void setTitle(String text)</code>	Setzt den Titel des Turtle-Fensters.
<code>void addStatusBar(int height)</code>	Erzeugt Statusleiste mit der geg. Höhe unten am Turtlefenster.
<code>void setStatusText(String text)</code>	Schreibt einen Text in die Statusleiste.

Vollständige Turtle-Dokumentation

<http://www.aplu.ch/classdoc/turtle/index.html>

Vollständige Dokumentation aller Java-Klassen

z.B. für „Color“ oder „Font“

<http://docs.oracle.com/javase/8/docs/api/>