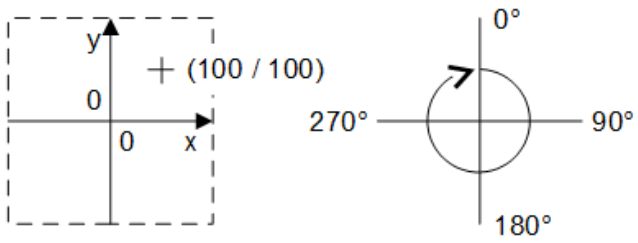
**Grundlagen**

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

### Zeichnen und Bewegung

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

### Kreise / Bögen

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

### Flächen füllen

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

**Sonstiges**

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