Übung 2: Methoden und Transformationen

In dieser Übung lernen Sie die Verwendung von Methoden in Processing kennen, insbesondere die beiden vorgegebenen Methoden `setup()` und `draw()`. Anschließend manipulieren Sie das Koordinatensystem mit Hilfe von Transformationen.

Funktion	Beschreibung
`setup()`	„Initialisierungscode“: Wird einmalig beim Start ausgeführt
`draw()`	Wird endlos wiederholt aufgerufen. Zeichenbefehle gehören hier hinein.

Aufgabe 1: Zeichnen mit Methode

Im ersten Übungsblatt haben Sie eine Figur erstellt, die sich aus einer Abfolge verschiedener Zeichen- und Farbbefehle ergibt. Um in längeren Programmen übersichtlicher und strukturierter vorzugehen, ist es hilfreich solche zusammengehörenden Befehle in eine einzige Methode zusammenzufassen bzw. zu kapseln.

So könnte *Frosty, der Schneemann* mit einem Aufruf von `drawFrosty();` erstellt werden, wie im Bild zu sehen ist.

Beachten Sie, dass einmalige Befehle wie das Setzen der Größe der Zeichenfläche in der `setup`-Methode stehen sollten.

→ Kapseln Sie die Figur aus dem ersten Übungsblatt in eine Methode und rufen Sie diese Methode innerhalb der `draw`-Methode auf.

```
void setup() {

  size(100, 160);
  smooth();
}

void draw() {

  drawFrosty();
}

void drawFrosty() {

  //Zeichenbefehle:
  ...
}
```

→ Führen Sie den Sketch aus und betrachten Sie das Ergebnis. Hat sich das Bild verändert? Wenn ja, können Sie feststellen warum das so ist?

Aufgabe 2: Transformationen

Mit der Methode aus der ersten Aufgabe können Sie nun Ihre Figur durch einen einfachen Aufruf zeichnen lassen. Das können Sie sich zunutze machen, um die Figur zu „klonen“ und zu manipulieren. Denken Sie an einen Stempel: Sie könnten ihn verschieben und sogar drehen. Dies soll die Methode nun auch ermöglichen.

Betrachten Sie hierzu die folgenden Befehle, die Processing zur Verfügung stellt:

Funktion	Beschreibung
`translate(deltaX, deltaY)`	Verschiebt den Ursprung um `deltaX` Einheiten nach links/rechts und `deltaY` Einheiten nach oben/unten.
`rotate(winkel)`	Dreht die Zeichenfläche um `winkel (Einheit: rad)` um den Ursprung.
`radians(grad)`	Rechnet `grad` von deg in rad um, z.B. `radians(180)` ergibt Pi.
`scale(faktor), scale(x, y)`	Skaliert das Koordinatensystem um den (die) angegebenen Faktor(en).

→ Fügen Sie der Methode aus der ersten Aufgabe mehrere Parameter hinzu (x, y, rotation), mit denen die Position und Drehung Ihrer Figur einfach verändert werden kann. Es soll zuerst verschoben und dann gedreht werden.

Eventuell müssen Sie Ihre Zeichenfläche vergrößern um genug Platz zum Verschieben zu haben.

Rufen Sie Ihre Methode nun in `draw` mehrmals auf (wobei Sie jeweils verschiedene Parameterwerte für Verschiebung und Drehung verwenden können), um somit mehrere Figuren zu erzeugen.

→ Entspricht das Ergebnis Ihren Erwartungen? Falls nein, warum nicht?

Aufgabe 3: Matrix-Stack

Um den unerwünschten Effekt zu vermeiden, dürfen die Transformationen jeweils nur für einen Methodenaufruf gelten. Dies kann man erreichen, in dem man sich den aktuellen Status des Koordinatensystems zu Beginn der Methode merkt und ihn am Ende wieder herstellt. Dazu ist der Matrix-Stack da.

Betrachten Sie die folgenden Stack-typischen Methoden um den Matrix-Stack zu verwenden:

Funktion	Beschreibung
`pushMatrix()`	Speichert den aktuellen Zustand des Koordinatensystems.
`popMatrix()`	Lädt den letzten Zustand des Koordinatensystems.

→ Binden sie `pushMatrix` als ersten Befehl und `popMatrix` als letzten Befehl in Ihre Methode ein. Führen Sie den Sketch erneut aus und überprüfen Sie, dass das Ergebnis nun den Erwartungen entspricht.

→ Probieren Sie nun auch die andere Reihenfolge der Transformationen in Ihrer Methode aus (erst drehen, dann verschieben) - was fällt Ihnen im Vergleich zu vorher auf? Warum ist das so?