Einführung

Bis jetzt waren unsere selbsterstellten Klassen nur eine Sammlung verschiedener Datentypen (Ausnahme: Die Klasse `Stoppuhr` enthielt auch Methoden.) Sinn und Zweck der Klassen in der *objektorientierten Programmierung*, einer Standardtechnik des *Software Engineering*, ist es aber, Daten und Methoden zu einer Einheit zusammen zu fassen. Daten existieren nur in kleineren überschaubaren Programmen alleine, d.h. ohne zugehörige Methoden. In größeren Projekten gibt es Daten und Methoden, die auf die Daten zugreifen. Durch Zusammenfassung zu einer Klasse schafft man Ordnung in größeren Programmen und eine bessere Gliederung.

Methoden in Klassen

Ein Beispiel soll das Zusammenwirken von Daten und Methoden in einer Klasse demonstrieren. Ein Programm soll mit Brüchen arbeiten. Die Daten zu einem Bruch sind dann `zaehler` und `nenner`. Sinnvolle Methoden sind `addiere`, `subtrahiere`, `multipliziere` und `dividiere`.

```
class Bruch
{ int zaehler;
  int nenner;

  void multipliziere (Bruch b)
  { this.zaehler = this.zaehler * b.zaehler;
    this.nenner  = this.nenner  * b.nenner;
  }

} // class Bruch
```

Bemerkungen

- Das Objekt, auf das eine Methode angewandt wird, wird in Java durch die Standardvariable `this` bezeichnet. Wenn `multipliziere` auf `zaehler` zugreift, so ist damit `this.zaehler` gemeint. Die Qualifikation mit `this` kann man weg lassen, wenn klar ist, dass `zaehler` ein Feld des Objektes und nicht vielleicht eine lokale Variable ist.

- **Aufruf von Methoden:** Zuerst werden zwei Brüche in einem Hauptprogramm erzeugt und initialisiert:

```
Bruch a = new Bruch();
a.zaehler = 1; a.nenner = 2;
Bruch b = new Bruch();
b.zaehler = 3; b.nenner = 5;
```

Der Aufruf der Methode `multipliziere` zur Multiplikation der beiden Brüche ist nun etwas ungewöhnlich:

```
a.multipliziere(b);
```

Man wählt zunächst ein Objekt aus (hier `a`) und wendet anschließend eine Operation darauf an (hier `multipliziere`) mit der Übergabe eines Parameters (hier `b`). Das Ergebnis steht danach in `a`.

Mit Begriffen der objektorientierten Programmierung ausgedrückt macht man Folgendes:

- Das Objekt `a` bekommt die *Meldung* oder *Nachricht* `multipliziere`, wodurch die `multipliziere`-Methode aufgerufen wird. Meldungen führen also zum Aufruf von Methoden.
- Das Objekt `a` nennt man den *Empfänger* der Meldung `multipliziere`. Der Empfänger ist immer dasjenige Objekt, dem die Meldung geschickt wird.

Konstruktoren

In obigem Beispiel musste man ein Objekt erzeugen und danach konnte man es erst initialisieren. Java stellt spezielle Initialisierungsmethoden zur Verfügung, die bei der Erzeugung eines Objektes automatisch aufgerufen werden und in denen man diverse Einstellungen machen kann. Diese Methoden heißen *Konstruktoren* und haben den gleichen Namen wie die Klasse selbst. Sie werden ohne das Schlüsselwort `void` und ohne Funktionstyp aufgerufen, können aber Parameter haben. Eine Klasse kann sogar mehrere Konstruktoren haben. Sie haben alle denselben Namen wie ihre Klasse, unterscheiden sich aber in ihren Parametern.

```
class Bruch
{ int zaehler;
  int nenner;

  void multipliziere (Bruch b)
  { this.zaehler = this.zaehler * b.zaehler;
    this.nenner  = this.nenner  * b.nenner;
  }
```

```
  Bruch (int z, int n)
  { this.zaehler = z;
    this.nenner  = n;
  }

  Bruch ()
  { this.zaehler = 0;
    this.nenner  = 1;
  }

} // class Bruch
```

Durch

```
Bruch a = new Bruch(1, 2);
Bruch b = new Bruch();
```

wird der Bruch `a` zu $\frac{1}{2}$. Weil beim Erzeugen von `b` keine Parameter angegeben werden, wird der andere Konstruktor aufgerufen und `zaehler` auf 0 und `nenner` auf 1 gesetzt.
Wird in einer Klasse kein Konstruktor angegeben, so fügt Java automatisch einen parameterlosen Konstruktor hinzu, der den Feldern Standardwerte gibt (0, `null`, `false`,...). Deshalb steht bei der Erzeugung eines Objektes immer mindestens ein Klammerpaar `()`.

Aufgabe

Eine *komplexe Zahl* (z.B. $2 + 3i$) besteht aus einem *Realteil* (hier 2) und einem *Imaginärteil* (hier 3). Dabei gilt $i^2 = -1$ und $a, b, c, d \in \mathbb{R}$. Für die Grundrechenarten gilt:

$$\begin{aligned}
(a+bi) + (c+di) &= (a+c) + (b+d)i \\
(a+bi) - (c+di) &= (a-c) + (b-d)i \\
(a+bi) \cdot (c+di) &= (ac-bd) + (ad+bc)i \\
\frac{a+bi}{c+di} &= \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i
\end{aligned}$$

Implementiere eine Klasse `Komplex` mit geeigneten Konstruktoren.
Zusatzaufgabe 1: Erweitere die Klasse um geeignete Ein- und Ausgabemethoden.
Zusatzaufgabe 2: Stelle die komplexen Zahlen als Ortvektoren im $\mathbb{R}^2$ dar (Turtle.class benutzen!) und veranschauliche damit die Grundrechenarten. Der Realteil einer komplexen Zahl sei die x-Koordinate, der Imaginärteil sei die y-Koordinate.