Der Punkt-Operator

Um eine Methode aufzurufen, sind wir bislang folgendermaßen vorgegangen:

```
Bruch meinBruch = new Bruch(2,3); // Objekt anlegen
Bruch deinBruch = new Bruch(5,6); // Objekt anlegen
Bruch ergebnis = new Bruch(meinBruch);
ergebnis.multipliziere(deinBruch);
```

Hinter dem Punkt steht ein Methodenname, vor dem Punkt ein Objekt (oder ein Klassenname - siehe weiter unten).
Wenn die im Beispiel benutzte **`multipliziere`**-Methode nichts anderes tut, als das Objekt zu verändern, auf dem sie angewendet wurde (im Beispiel hatte **`ergebnis`** erst den Wert von **`meinBruch`**, nach **`multipliziere`** den Wert des Ergebnisses), werden Rechnungen unangenehm unübersichtlich.
Eleganter wäre: **`ergebnis = meinBruch · deinBruch`**

Leider geht das nicht so ohne Weiteres. Aber dass die Methode **`multipliziere`** das „eigene" Objekt verändert, ist eine unschöne Designentscheidung. Viel eleganter ist folgende Lösung:

```
public Bruch multipliziere2(Bruch b) {
   int erg_z = z * b.getZaehler();
   int erg_n = n * b.getNenner();
   return new Bruch(erg_z,erg_n);
}
```

Diese neue Methode verändert nicht die Eigenschaften **`z`** und **`n`** des „eigenen" Objektes. Stattdessen wird ein neues Ergebnisobjekt erstellt und als Rückgabewert zurückgegeben. Damit ist nun folgender Aufruf möglich:

```
Bruch meinBruch = new Bruch(2,3); // Objekt anlegen
Bruch deinBruch = new Bruch(5,6); // Objekt anlegen
Bruch ergebnis = meinBruch.multipliziere2(deinBruch);
```

Und was noch viel besser ist: Aufrufe dieser Art kann man verketten! Mit entsprechenden Methoden ist nun folgendes möglich:

```
Bruch d = a.addiere2(b).multipliziere2(c);
```

Hier wird also zunächst a + b gerechnet. Da das Ergebnis wieder ein Bruch-Objekt ist, kann darauf auch wieder eine Methode angewandt werden. Im Objekt d ist nun also das Ergebnis der Rechnung (a+b) * c enthalten!

Statische Methoden

Bestimmte Methoden können aufgerufen werden, obwohl kein Objekt der entsprechenden Klasse existiert. Z.B. ist die Methode **`ggT`** in der Bruch-Klasse auch ohne konkretes Bruch-Objekt sinnvoll. Deswegen taucht bei der Methodendefinition das Wort **`static`** auf. Um diese Methode von außen ohne existierendes Bruch-Objekt aufzurufen, schreibt man:

```
int c = Bruch.ggt(6,8);
```

Achtung: Statische Methoden dürfen nicht auf Objekteigenschaften zugreifen!