Spezielle Objektmethoden

Seit C++11 gibt es 6 spezielle Objektmethoden (vorher nur 1-4), und zwar:

1) Der (die) Konstruktor(en): Es können beliebig viele existieren (constructor, ctor)
2) Der Destruktor: den gibt es nur einmal (destructor, dtor)
3) Der Kopier-Konstruktor (copy constructor)
4) Der Kopier-Zuweisungsoperator `operator=()` (Copy-Assignment-Operator)
5) Der Move-Konstruktor (move constructor)
6) Der Move-Zuweisungsoperator `operator=()` (Move Assignment Operator)

Diese Methoden sind speziell, weil sie sowohl vom Programmierer **als auch vom Compiler** generiert werden können. Der Compiler erzeugt diese Methoden nur, wenn sie noch nicht vom Programmierer erstellt wurden und wenn bestimmte Voraussetzungen zutreffen.

Alle Konstruktoren (Punkte 1, 3, 5) heißen *`Objekttyp()`* (z.B. `Bruch()`). **Sie haben keinen Rückgabetyp (auch nicht void!).**

Der Destruktor heißt *`~Objekttyp()`* (z.B. `~Bruch()`). **Er hat keinen Rückgabetyp (auch nicht void!).**

Alle Zuweisungsoperatoren (Punkte 4 und 6) heißen `operator=()` und haben den Rückgabetyp `Objekttyp&` (z.B. `Bruch&`).

Nicht alle Objekte besitzen alle speziellen Methoden. Insbesondere haben alle Datentypen, die in C möglich sind, keine dieser Methoden (weil es diese in C nicht geben kann). Solche Datentypen nennt man manchmal POD: Plain Old Data. Sie dürfen u.a. nur public Datenattribute und keinen Konstruktor haben.

Im Gegensatz dazu spreche ich von „echten“ Objekten, wenn es sich nicht um POD handelt.

Beispiel:

```
struct Komplexe_Zahl {
     double re, im;
};    // ist POD

struct Komplexe_Zahl {
private:
     double re, im;
};    // ist kein POD (wegen des private), ist aber in dieser Form unbenutzbar
```