Kapitel 18

Konstruktor / Destruktor einer Klasse

Konstruktoren

- Zur Initialisierung der Eigenschaften einer Klasse werden Konstruktoren verwendet.
- Der Konstruktor wird bei der Definition eines Objekts der Klasse automatisch aufgerufen.
- Der Konstruktor besitzt den gleichen Namen wie die Klasse.
- Bis auf Objekte derselben Klasse dürfen in einem Konstruktor auch Objekte definiert werden.

```
CKonto::CKonto(string name)
{
    m_name = name;
}
```

Konstruktoren

- Ein Konstruktor besitzt **keinen** Rückgabewert; auch **nicht** den Typ *void*.
- Er darf nicht *private* sein.
- Für globale Objekte wird der Konstruktor unmittelbar vor der Ausführung der *main()*- Funktion ausgeführt und bei lokalen Objekten an der Stelle, an der sie definiert werden.
- Hat ein Konstruktor Parameter, werden diese bei der Definition des Objekts angegeben.

```
CKonto konto2("Mueller, Anna");
```

Standardkonstruktor

- Ist für eine Klasse kein Konstruktor definiert, so hat sie einen parameterlosen Standardkonstruktor.

```
CKonto();
```

- Auch der Aufruf des Standardkonstruktors erfolgt bei der Definition eines Objekts:

```
CKonto konto1;
```

Konstruktoren mit einem Parameter

- Hat ein Konstruktor nur einen Parameter, kann die Initialisierung bei der Definition auch über eine Zuweisung erfolgen.
- Der Aufruf erfolgt bei folgender Definition:

```
CKonto konto2("Mueller, Anna");
```

- Oder bei folgender Definition:

```
CKonto konto2 = "Mueller, Anna";
```

Konstruktoren mit einem Parameter

- Umgekehrt können auch einfache Datentypen (z.B. *int*) entsprechend der Konstruktor-Schreibweise initialisiert werden.
- Der Aufruf erfolgt bei folgender Definition:

```
int x = 4;
```

- Oder:

```
int x(4);          // Runde Klammern!
```

Konstruktoren mit mehr als einem Parameter

- Bei der Definition eines Objekts können hinter dem Namen des Objekts Anfangswerte in Klammern angegeben werden:

```
CKonto konto4("Maier, Hans", 10002000, 1000.00);
```

- Der Compiler ordnet dieser Initialisierungsliste einem Konstruktor mit gleicher Parameterliste zu:

```
CKonto(string name, int nummer, double stand);
```

=> Bsp: CGirokonto

Elementinitialisierer

- Unter Verwendung von Elementinitialisierern kann statt der bisherigen Schreibweise:

```
CKonto::CKonto(string name)
{
    m_name = name;
}
```

- die Konstruktordefinition auch wie folgt geschrieben werden:

```
CKonto::CKonto(string name) : m_name(name)
{
}
```

Elementinitialisierer

- Ein Elementinitialisierer wird nach der Parameterklammer der Konstruktor-Definition durch einen Doppelpunkt eingeleitet.
- Nach dem Doppelpunkt werden die zu initialisierenden Eigenschaften aufgelistet, wobei die Initialwerte jeder Eigenschaft in runden Klammern angegeben werden.

Destruktoren

- Destruktoren werden am Ende des Gültigkeitsbereichs des Objekts automatisch aufgerufen (z.B. Blockende).
- Bei globalen Objekten werden Destruktoren am Ende von *main* aufgerufen.
- Der Name eines Destruktors beginnt mit dem Tilde-Zeichen ~ gefolgt von dem Namen der Klasse.
- Ein Destruktor hat keine Parameter und liefert keinen Wert zurück. Er darf nicht im *private* Bereich einer Klasse liegen.

```
CKonto::~CKonto()
{
}
```

Aufruf von externen Funktionen in Methoden

- Gibt es innerhalb der Klasse keine Methode mit dem gleichen Namen, erfolgt der Aufruf der Funktion wie gewohnt.
- Gibt es innerhalb der Klasse eine Methode mit dem gleichen Namen wie die externe Funktion, wird die Klassen-Methode aufgerufen.
- Um aber beispielsweise eine Funktion der Standardbibliothek aufzurufen, schreibt man *std::* vor den Funktionsnamen.

```
std::pow(x, y);
```

Aufruf von externen Funktionen in Methoden

- Handelt es sich bei der aufzurufenden Funktion um eine Funktion, die in keinem Namensbereich liegt z.B. *Test()*, wird einfach auf die Angaben des Namensbereichs verzichtet:

```
z = ::Test();
```