Notizen AIK ProgTech 3:
Statische Member ("Klassenvariablen") und Statische Methoden ("Klassenmethoden")

Klaus Kusche, 2011/2012

Statische Member

- Statische Member sind Member, in deren Deklaration im **`class`** vorne **`static`** steht.
- Statische Member funktionieren wie andere **statische Variablen**:
  - Sie existieren, *solange das Programm läuft* (vom Programmstart bis zum Programmende), auch wenn es gar keine Objekte der Klasse gibt.
  - Sie existieren *nur genau einmal pro Klasse*, *nicht* einmal pro Objekt (d.h. alle Objekte einer Klasse greifen auf *denselben Wert* zu).
- Statische Member müssen nicht nur im **`class`** deklariert werden, sondern zusätzlich außerhalb des **`class`** definiert werden, und zwar mit *klassenname***`::`** vor dem Namen (siehe Beispiel), so wie Methoden außerhalb des **`class`**:
  - Die *Deklaration im* **`class`** sagt nur, dass die Klasse eine solche Variable hat.
  - Erst die *Definition außerhalb des* **`class`** legt wirklich *Speicher* dafür an (weil eine Klassenvariable ja *nicht* wie normale Membervariablen automatisch erst beim Anlegen eines Objektes angelegt wird).

  Fehlt die separate Definition, kommt eine Fehlermeldung vom Linker.

  Beispiel:

```
class myClass {
  ...
  static int myObjCtr;
};

int myClass::myObjCtr = -1;
```

- Klassenvariablen werden so wie statische und globale Variablen beim Programmstart *automatisch auf 0* gesetzt. Will man sie auf einen anderen Wert initialisieren, gehört die *Initialisierung in die Definition*, ***nicht*** in die Deklaration (denn initialisieren kann man nur an der Stelle, wo der Speicher angelegt wird)!

  Falls das statische Member selbst wieder ein Objekt ist, wird (noch vor **`main`** !) sein *Konstruktor* ausgeführt. Wie bei globalen Objekten gilt:
  Die Reihenfolge, in der Konstruktoren beim Programmstart ausgeführt werden, ist zufällig, das kann bei voneinander abhängigen statischen / globalen Objekten zu ganz kniffeligen Fehlern führen!
  (d.h. man darf sich im Konstruktor nicht darauf verlassen, dass andere statische / globale Variablen schon in einem wohldefinierten Zustand sind!).
- Bei der Aufspaltung in **`.cpp`**- und **`.h`**-Files gehört die Definition in den **`.cpp`**-File (so wie der Code von Methoden, die nicht im **`class`** stehen), nicht in den **`.h`**-File!

- Der *Zugriff* auf statische Member erfolgt genauso wie auf normale Member.

  Ist ein statisches Member **`public`**, kann auch in Code außerhalb der Klasse direkt mit *klassenname*`::`*staticMember* darauf zugegriffen werden:

  ```
  myClass::myCounter = -1;
  ```

- Typische Anwendungen:
  - *Zähler* für die Anzahl der erzeugten Objekte und andere Statistiken, sowie zur Erzeugung eindeutiger Nummern für jedes neue Objekt.
  - *Verzeichnis* aller Objekte einer Klasse.

Statische Methoden

- Statische Methoden einer Klasse sind Methoden,
  die nicht für ein bestimmtes Objekt, sondern *für die ganze Klasse aufgerufen werden* (sogar dann, wenn *gar keine* Objekte der Klasse existieren).
- Sie werden deklariert, indem man im **`class`** vor den Prototypen der Methode **`static`** schreibt.
- Statische Methoden können auch wie ganz normale Methoden aufgerufen werden (mit *obj*`.`*meth*`(`…`)` oder *ptr*`->`*meth*`(`…`)`), aber das kommt *praktisch nicht* vor.

  Normalerweise werden sie ohne Objekt *nur mit dem Klassennamen* aufgerufen:
  klassenname`::`*meth*`(`…`)`

  Bei einem Aufruf *innerhalb des Codes der Klasse* kann das *klassenname*`::` entfallen.
- Da statische Methoden ohne Objekt aufgerufen werden können,
  gibt es zwei *Einschränkungen*:
  - In einer statischen Methode gibt es *kein* **`this`**
    (keinen Pointer auf das eigene Objekt).
  - In einer statischen Methode darf *nur auf die statischen Member* der Klasse zugegriffen werden, nicht auf die normalen Member (denn normale Member gehören immer zu einem Objekt, und wir haben ja *kein Objekt*!).
- Typische Anwendung: *Zugriff auf statische Member*, z.B.
  - Initialisierung oder Auslesen der Statistiken und Objektzähler
    (d.h. *Get- und Set-Methioden für statische Member*)
  - Suchen eines Objektes im Objekt-Verzeichnis einer Klasse, …