Kapitel 19

Vererbung

Vererbung

- Neben der Datenabstraktion und der Datenkapselung ist die Vererbung ein weiteres Merkmal der OOP.
- Durch Vererbung werden die Methoden und die Eigenschaften einer Klasse (Basisklasse) an eine andere Klasse (abgeleitete Klasse) vererbt.
- Die abgeleitete Klasse verfügt neben ihren eigenen Methoden und Eigenschaften auch über die der Basisklasse.

Vererbung

- Wenn eine Klasse B von einer Klasse A abgeleitet werden soll, so muss dies wie folgt geschrieben werden:

```
classdef B < A
    ...
end
```

- Somit erbt Klasse B alle vererbbaren (*public*, *protected*) Eigenschaften und Methoden, die von Klasse A kommen.

Vererbung

- In unserem CKonto-Beispiel gibt es nun eine neue Klasse CGiroKonto, die von der Basisklasse CKonto abgeleitet wird, also von CKonto erbt.

```
classdef CGiroKonto < CKonto
    ...
end
```

Vererbung

- CGiroKonto hat also alle vererbbaren (*public*, *protected*) Eigenschaften und Methoden der Basisklasse CKonto.
- Zusätzlich hat sie außerdem noch die eigenen Eigenschaften:
  - Dispokreditrahmen (m_dispo)
  - Sollzinssatz (m_sollzins)
- und die Methoden:
  - Konstruktor
  - SetDispo und GetDispo
  - SetSollzins und GetSollzins
  - Eingabe
  - Ausgabe

Zugriffsrecht protected

- Bisher haben wir die Zugriffsrechte *private* und *public* kennen gelernt.
- Das Zugriffsrecht *protected* wird bei der Vererbung wichtig.
- Auf *protected* Member einer Klasse kann nur von abgeleiteten Klassen zugegriffen werden.

Für properties:

```
properties (Access = protected)
    m_name;
end
```

Für methods:

```
methods (Access = protected)
    function name = ...
end
```

Zugriffsrechte auf Basisklassen

- Eine abgeleitete Klasse hat nur Zugriff auf die *public* und die *protected* Member der Basisklasse.
- Sie hat keinen Zugriff auf die *private* Member.

- WICHTIG:
  Der Konstruktor muss immer unter *public* stehen!

Mehrfache Vererbung

- Eine Klasse kann auch von mehreren Basisklassen erben. Dies wird durch ein & gekennzeichnet.

```
classdef C < A & B
    ...
end
```

- Somit erbt hier Klasse C alle vererbbaren (public, protected) Eigenschaften und Methoden, die von Klasse A und von Klasse B kommen.

Mehrfache Vererbung

- Im folgenden Beispiel wird die Klasse *CPlusKonto* von den Klassen *CSparKonto* und *CGiroKonto* abgeleitet.

```
classdef CPlusKonto < CSparKonto & CGiroKonto
    ...
end
```

Aufruf einer Methode der Basisklasse

- CKonto und CGiroKonto haben beide die Methode *Eingabe()*.
- Möchte man in CGiroKonto die *Eingabe()* von CKonto aufrufen, so erzeugt dies einen Fehler:

```
function obj = Eingabe(obj)
    obj = obj.Eingabe();
end
```

- Hier würde das Programm immer wieder die *Eingabe()* von CGiroKonto aufrufen.

Aufruf einer Methode der Basisklasse

- Da man aber die *Eingabe()* von CKonto aufrufen will, muss man dies so schreiben:

```
function obj = Eingabe(obj)
    obj = obj.Eingabe@CKonto();
end
```

Aufruf einer Methode der Basisklasse

- Die allgemeine Form lautet:

```
<Methodenname>@<Klassenname>(object , <Parameter>)
```

bzw.:

```
object.<Methodenname>@<Klassenname>(<Parameter>)
```

Aufruf einer Methode der Basisklasse

- So ähnlich verhält es sich im Konstruktor.
- Soll der Konstruktor von CGiroKonto den Konstruktor von CKonto aufrufen, muss man schreiben:

```
function obj = CGiroKonto(name, nummer, stand, dispo, sollzins)
    obj@CKonto(name, nummer, stand);
    ...
end
```

Beispiel der Klasse CGiroKonto

- Ein komplettes Beispiel der Klassen CGiroKonto und CKonto nebst Hauptprogramm finde Sie unter Beispiel 20 (auf der Webseite zu finden).