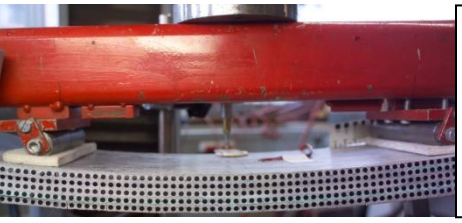
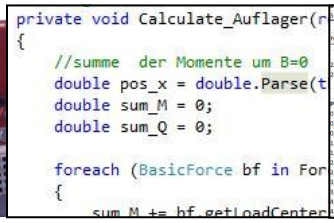
Skriptum Bauinformatik

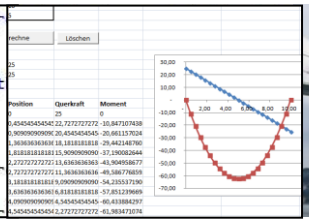
SS 2013 (Vorlesung IV)

Stand: 23.04.2013

Dr. Johannes Lange







Inhalt

Objektorientierte Programmierung – „Großes“ Beispiel

Schon im Skript III wurden die grundlegenden Schritte der Objektorientierung in VBA beschrieben. An dieser Stelle soll dies nicht wiederholt, sondern anhand eines einfachen Beispiels erklärt werden. Ziel ist es, die Anwendung der Objektorientierung in VBA zu zeigen. Nutzen Sie hierfür das Beispiel in Excel (**VorlesungUebungIVb.xlsm**) aus der alle Code-Schnipsel kopiert sind.

Zunächst wird eine Klasse erstellt und ein Objekt daraus instanziiert (erzeugt). Diese Klasse wird durch Attribute und Methoden ergänzt und die Kapselung der Attribute mit dem Aufruf *property* gezeigt. Die Funktion „with“ zur Reduzierung der Schreibarbeit und die Verwendung einer zweite Klasse schließen dieses große Beispiel ab.

Klasse erstellen

In VBA werden Klassen als Klassenmodule erstellt. Diese können im z.B. Menü über *Einfügen/Klassenmodul* erzeugt werden. Es entsteht ein neues Klassenmodul mit dem automatischen Namen „*Klasse1*“, der über die Eigenschaften verändert werden kann (Abbildung 1).

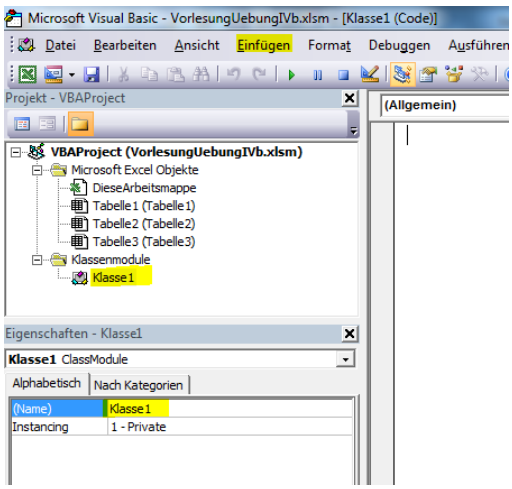


Abbildung 1: Klasse erzeugen und umbenennen

Die Klasse ist angelegt, das Codefenster jedoch noch leer.

Zunächst wird in der Klasse ein öffentliches (*Public*) Attribut (Variable) und eine öffentliche (*Public*) Methode (Funktion) erstellt.

```
'Klasse1 Attribute
Public Value As String


'Klasse1 Methoden
Public Function TestFunc()
    MsgBox ("Starte TestFunc()")
    TestFunc = "ok"
End Function
```

- In einem Klassenmodul ist kein „*Sub*“ notwendig, diesen „Rahmen“ bietet das Klassenmodul automatisch.
- Als Public deklarierte Attribute (Variablen) und Methoden (Funktionen) können aus anderen (Klassen-)Modulen verändert bzw. aufgerufen werden.
- „*Public*“ oder „*Private*“ ersetzt „*Dim*“.
- Eine Funktion muss immer einen Rückgabeparameter haben, der genauso heißt wie die Funktion (hier: *TestFunc*).
- Die Klasse kann nicht eigenständig ausgeführt werden → die Erzeugung eines Objekts ist notwendig. (Klasse ist der Bauplan des Objekts)

Erzeugen eines Objekts der Klasse

Über das Menü (Einfügen/Modul) wird ein „normales“ Modul erzeugt, in dem ein Objekt der Klasse erzeugt werden soll.

Zunächst wird eine Variable vom Typ Klasse1 deklariert. Intellisense zeigt die erstellte Klasse in der Typen-Auswahlliste schon an (Abbildung 2).

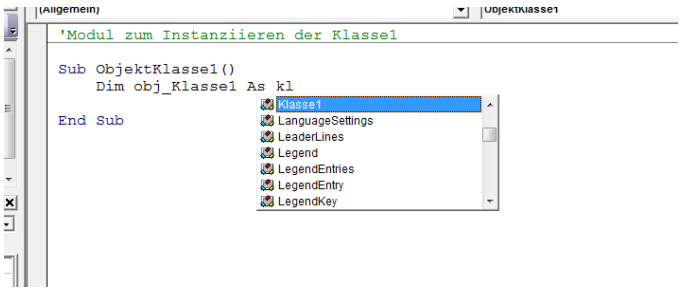


Abbildung 2: Intellisense:neue Klasse in der Auswahlliste

Der vollständige Code der Objekterzeugung besteht neben der Deklaration aus der Instanziierung des Objekts mit der Funktion „New“ und der Zuweisung mit der Funktion „Set“ und dem Zuweisungsoperator „=“.

```
Dim obj_Klasse1 As Klasse1      'nur Deklaration
Set obj_Klasse1 = New Klasse1   'erzeugen eines Objekts und Zuweisung
```

Ein Objekt ist instanziiert. „obj_Klasse1“ enthält alle Attribute und Methoden der Klasse und kann

von Modul aus auf alle Public-Elemente zugreifen. Zur Verwendung dieser Attribute und Methoden wird ein Punkt nach der Variablen gesetzt. Intellisense zeigt die möglichen Elemente (Abbildung 3).

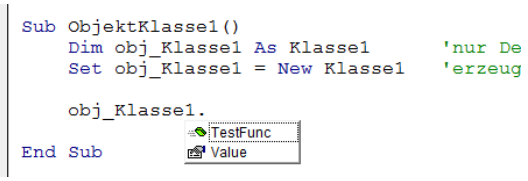


Abbildung 3: Public- Elemente werden mit Intellisense angezeigt

Es besteht jetzt Lese-/Schreibzugriff auf die Variable oder die Funktion. Folgender Code Zeigt das Schreiben und Lesen der Variable „Value“. Beachten Sie den Punkt als Verbindung zwischen Objekt und Variablenname. Entsprechend lässt sich die Methode aufrufen.

```
Sub ObjektKlasse1()
    'Deklaration und Objekterzeugung
    Dim obj_Klasse1 As Klasse1      'nur Deklaration
    Set obj_Klasse1 = New Klasse1   'erzeugen eines Objekts und Zuweisung

    'Schreiben und Lesen in/aus der Variable Value
    obj_Klasse1.Value = "Hallo Welt"
    MsgBox (obj_Klasse1.Value)

    'Starten der Methode und Ausgabe des Rückgabewerts
    result = obj_Klasse1.TestFunc()
    MsgBox ("Rückgabewert: " & result)
End Sub
```

Die Attribute und Methoden der Klasse lassen sich nur über das Objekt aufrufen, versuchen Sie dies mit der Klasse werden Sie keinen Erfolg haben:

```
'Klassendirektaufruf ist nicht möglich
Klasse1.Value = 5
```

Zugriffsfunktionen

Üblicherweise werden Attribute nicht direkt als Aufruf des Objekts verändert. Sie sollen gekapselt werden und nur über indirekten Zugriff (definierte Schnittstelle) bearbeitet werden können. Dies wird an dem neuen Attribut „Hoehe“ gezeigt. Neben der Deklaration Private werden zwei Methoden geschrieben, die dieses Attribut lesen und schreiben können:

```
Private Hoehe As Integer    'Private: nur aus dem Klassenmodul zugreifbar

'Zugriff mit Methoden
Public Function Hoehe_Ausgabe() 'Lesen
    Hoehe_Ausgabe = Hoehe
End Function
Public Function Hoehe_Setzen(loc_hoehe As Integer) 'Schreiben
    Hoehe = loc_hoehe
End Function
```

Die Funktion *Hoehe_Ausgabe()* setzt als Rückgabeparameter den Wert von *Hoehe* ein, *Hoehe_Setzen* bekommt einen Parameter mitgegeben (*loc_hoehe*) und weist diesem eine *Hoehe* zu.

```
'Hoehe bearbeiten
obj_Klasse1.
d Sub
```

Hoehe_Ausgabe
Hoehe_Setzen
TestFunc
Value

Im Modul zeigt Intellisense, dass die *public* Methoden *Hoehe_Ausgabe* und *Hoehe_Setzen* zugreifbar sind, jedoch nicht die *Private* –Variablen *Hoehe* selbst. Die Methoden können wie im folgenden Listing geschrieben und ausgelesen werden.

```
'Hoehe bearbeiten Schreiben und Lesen
obj_Klasse1.Hoehe_Setzen (10)
MsgBox ("Hoehe ausgeben: " & obj_Klasse1.Hoehe_Ausgabe())
```

Eine elegantere Möglichkeit für diesen Zugriff ist die Verwendung einer Property mit ihren Lese- und Schreibroutinen. Sie ist im Menü über „Einfügen/Prozedur einfügen“ erstellbar (Abbildung 4).

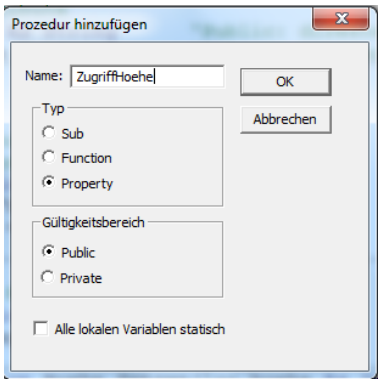


Abbildung 4: Prozedur hinzufügen - Erstellung einer Property

Dies erzeugt in der Klasse eine Code-Schablone für den Zugriff auf eine Variable, die entsprechend erweitert werden muss (siehe Listing2).

Listing 1: Code-Schablone

```
'Zugriff mit Getter, Setter
Public Property Get ZugriffHoehe() As Variant

End Property

Public Property Let ZugriffHoehe(ByVal vNewValue As Variant)

End Property
```

Listing 2: Ergänzung, Erweiterung der Codeschablone

```
'Zugriff mit Getter, Setter
Public Property Get ZugriffHoehe() As Variant
    ZugriffHoehe = Hoehe
End Property

Public Property Let ZugriffHoehe(ByVal vNewValue As Variant)
    Hoehe = vNewValue
End Property
```

Der Zugriff aus dem Modul erfolgt vergleichbar zu ersten Version, nur dass keine Aufrufe über Methoden mit Klammern erfolgen. Ob der Zugriff direkt auf eine *Public* Variable mit dem Namen *ZugriffHoehe* oder auf eine *Property* erfolgt, kann von hier aus nicht erkannt werden.

```
'Zugriff mit Getter und Setter
obj_Klasse1.ZugriffHoehe = 50
MsgBox ("Get/Let: " & obj_Klasse1.ZugriffHoehe)
```

With- Beschleunigter Zugriff

Wenn Sie die Schreibarbeit scheuen, immer den Namen des Objekts im Modul davor zuschreiben, verwenden Sie die Funktion *With*. Alle nachfolgenden Aufrufe werden automatisch ergänzt und auch Intellisense wird angepasst:

```
'Spar-Funktion mit With
With obj_Klasse1
    .ZugriffHoehe = 50
    MsgBox ("With probieren: " & .ZugriffHoehe)
End With
```

Initialisierung und Terminierung eines Objekts

Schon beim Erzeugen des Objekts sowie beim Löschen kann es notwendig sein Variablen zu initialisieren oder zu speichern. Hierzu können zwei Methoden der Klasse aufgerufen werden. Sie lassen sich auch in Dropdownfeldern oberhalb des Code-Fensters aufrufen.

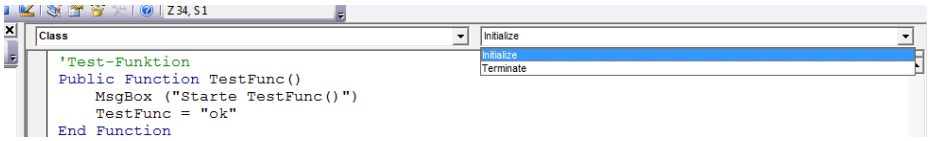


Abbildung 5: Dropdown-felder für Initialize und Terminate

```
'Beim Erzeugen (Instanziieren des Objekts)
Private Sub Class_Initialize()
    MsgBox ("Erzeuge Objekt der Klasse <Klasse1>")
    Hoehe = 100 'Möglicher Initialwert
End Sub

'Beim Löschen des Objekts
Private Sub Class_Terminate()
    MsgBox ("Lösche ein Objekt der Klasse <Klasse1>")
End Sub
```

Erzeugen eines zweiten Objekts

Das Instanziieren eines zweiten Objekts geht exakt wie beim ersten, nur dass die Namen nicht gleich seinen dürfen. Es können beliebig viele (soweit der Speicher des Rechners das hergibt) Objekte erzeugt werden.

```
'zweites Objekt erzeugen
Dim Obj_Klasse2 As Klasse1
Set Obj_Klasse2 = New Klasse1

MsgBox (Obj_Klasse2.ZugriffHoehe & " " & obj_Klasse1.ZugriffHoehe)
```

Die Abfrage *MsgBox* zeigt, dass die zwei Objekte etwas Unterschiedliches beim *ZugriffHoehe* zurückgeben.

Arbeiten mit einer zweiten Klasse

Mehrere Klassen arbeiten in einem Programm miteinander, im Folgenden wird gezeigt, wie eine zweite Klasse und ein entsprechendes Objekt verwendet werden kann. Zunächst wird eine zweite Klasse angelegt, in der nur eine Variable und eine Initialisierung festgelegt werden.

```
'Klasse 2 zum experimentieren

Public hoehe von Klasse2

Private Sub Class_Initialize()
    MsgBox ("Initialize Klasse 2")
    hoehe_von_Klasse2 = 500
End Sub
```

In *Klasse1* wird eine Variable mit dem Typ *Klasse2* erzeugt und eine Zugriffsfunktion erstellt, in der ein Objekt der *Klasse2* erzeugt und die Variable abgefragt wird:

```
Private TestKlasse2 As Klasse2

Public Function TestKlasse2_Func()
    Set TestKlasse2 = New Klasse2
    TestKlasse2_Func = TestKlasse2.hoehe_von_Klasse2
End Function
```

Das Modul ruft jetzt nur noch die Funktion aus der Klasse1 ab und kennt die Klasse2 gar nicht. Die Klasse 2 ist also gekapselt.

```
'Zweite Klasse indirekt abfragen
MsgBox ("Zweite Klasse abfragen: " & obj_Klasse1.TestKlasse2_Func())
```

Viel Spaß beim experimentieren! Die Denkweise der Objektorientierung ist normalerweise nicht einfach verständlich, doch wenn Sie einige Beispiele durchspielen, werden Sie sich dies Stück für Stück erschließen. Visual Basic wird hierzu noch einige weitere Ergänzungen enthalten und sofort mit der Definition einer Klasse anfangen.

Allgemeines

Termine

	28.03.2013	(Ostern Abreisetag)	
1	04.04.2013	VBA I	
2	11.04.2013	VBA II	
3	18.04.2013	VBA III	
4	25.04.2013	VBA IV / Obj. Orient.	Test VBA
5	02.05.2013	VB I	Start Hausarbeit
	09.05.2013	-------	Feiertag
6	~~16.05.2013~~	VBII	→ *Terminverschiebung Di. 14.5. 17.20-18.50*
7	23.05.2013	Allg. Softwareentwicklung	Test VB
	30.05.2013	-------	Feiertag
8	~~06.06.2013~~	? DB (Datenbanken)	→ *Terminverschiebung →??*
9	13.06.2013	? BIM (Building Inf. Model)	
10	20.06.2013	? GIS*	
11	27.06.2013	? GIS*	Ende Hausarbeit
12	04.07.2013	? Wiederholung	
	????	? Klausur	

* GIS-Vorlesung wird durch Herrn Dipl.-Ing. Wilke gegeben.

Räume und Uhrzeit:

Vorlesung / Übung : Donnerstag: 15.40 – 17.10 // 17.20 – 18.50
Raum 215 / Rechnerpool

Klausuren

2 * Quicki-Test (15 min) (20%)
1 * Hausarbeit (10h) (30%)
1 * Abschlussklausur (50%)

Kontakt

Fragen, Informationen, Kritik und *Anmerkungen* am besten per Email oder während der Vorlesungen/Übungen:

Email: lange.johannes@gmx.de

Dieses Skript findet sich in digitaler Form auch unter www.scraper.de.