



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Übungen zu Wirtschaftsinformatik 2

LE 01 – Grundlagen von VBA und MS Access

Prof. Dr. Thomas Off

<http://www.ThomasOff.de/lehre>



Lernziel und Inhalte

Lernziel

- Wiederholung der Inhalte von Wirtschaftsinformatik 1
- Reaktivierung der vorhandenen Programmierkenntnisse
- Vorbereitung auf den Eingangstest für die Übung zu Wirtschaftsinformatik 2

Inhalt

- VBA für MS Access als Programmierumgebung innerhalb von MS Access praktisch kennenlernen
- Zentrale Konzepte der prozeduralen Programmierung in VBA kennenlernen und anwenden
- Ereignisverarbeitung und Oberflächengestaltung in VBA kennenlernen
- Aufbauen auf Kenntnissen anderer (prozeduraler) Programmiersprachen



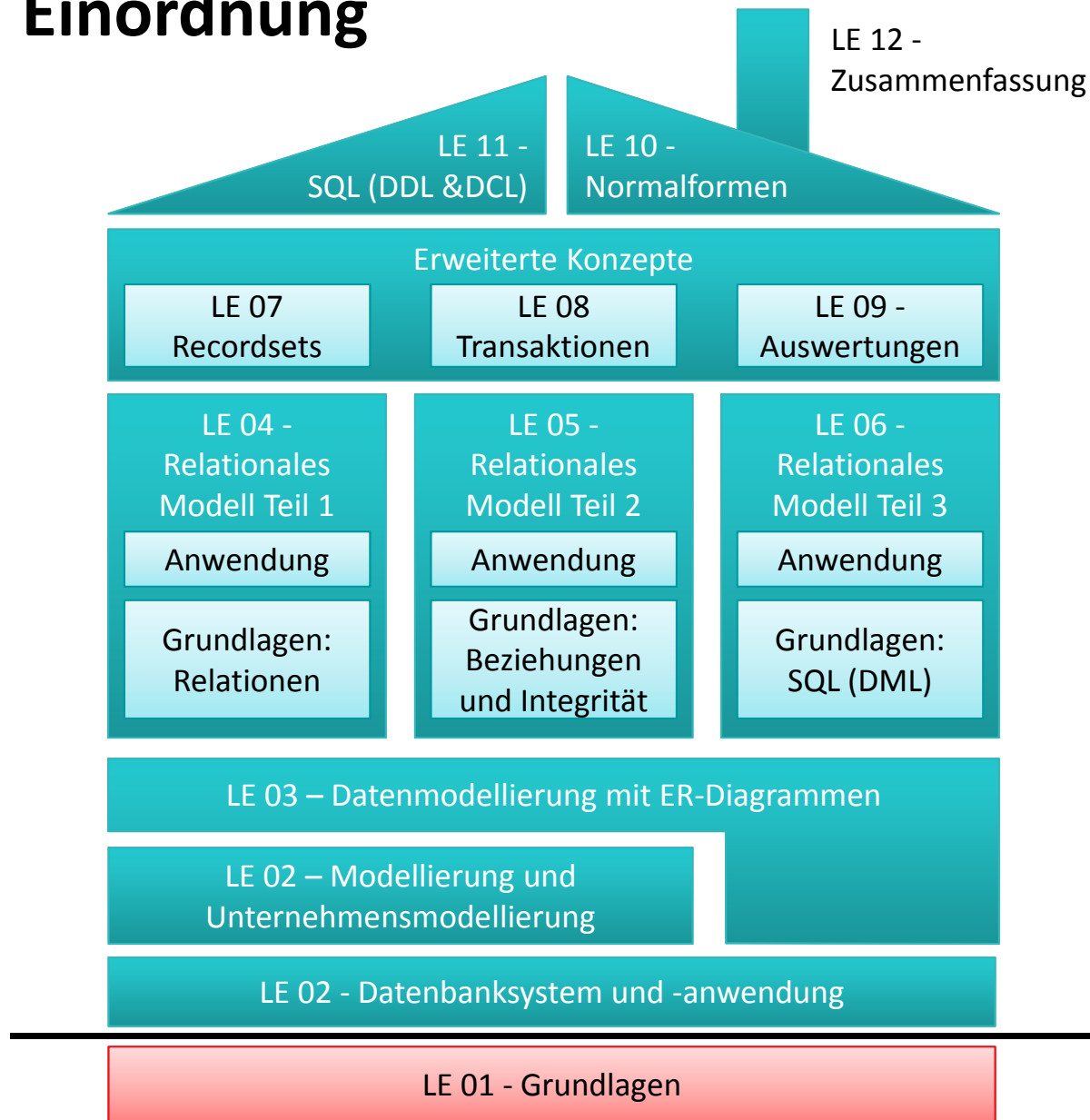
Hinweis

Das Seminar und die Übung zur LE 01 dienen der Wiederholung. Sie sind im WS 13/14 im Wesentlichen nur bei individuellem Bedarf zu absolvieren.

Ausnahmen sind

- die folgenden Übungen Ü1.10 bis Ü1.12, die absolviert werden müssen,
- sowie Ü1.13 bis Ü1.15, die absolviert werden sollten.

Einordnung





Inhalt

Lernziel, Inhalt und Einordnung

Einstieg in MS Access mit VBA

- Übung Hallo Welt-Programm

Übungen Grundlagen von VBA und MS Access

- Variable und Konstante mit Datentypen, Wert, Ausdruck, Zuweisung
- Verzweigungen
- Schleifen
- Module, Prozeduren und Funktionen
- Formulare, Ereignisse

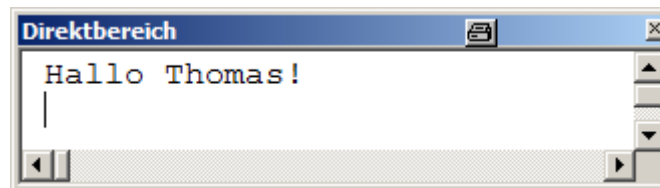
Ausblick

Übung: Hallo Welt!



Übung Ü1.0

- Schreiben Sie ein Programm in VBA für MS-Access, das eine Begrüßung und Ihren Namen im Direktfenster ausgibt.
- Beispiel:



Übung: Hallo Welt!



Übung Ü1.0 - Lösungsvorschlag

```
Option Compare Database
Option Explicit

Sub HalloName()

    Debug.Print "Hallo Thomas!"

End Sub
```

Übung: Variable



Übung Ü1.1:

- Deklarieren Sie zwei Variablen vom Typ String
- Initialisieren Sie eine Variable mit Ihrem Vornamen, die andere mit Ihrem Nachnamen
- Geben Sie erst "Hallo Welt!" und dann "Hallo " gefolgt von den Variablenwerten für Vorname und Nachname aus
- Ihr Ergebnis könnte etwa so aussehen:

```
Direktbereich
Hallo Welt!
Hallo Max Mustermann!
```


Übung: Variable



Ü1.1 - Lösungsvorschlag

```
Sub HalloNameVar()  
  
    Dim strName As String  
    Dim strVorname As String  
  
    Let strName = "Mustermann"  
    Let strVorname = "Max"  
  
    Debug.Print "Hallo Welt!"  
    Debug.Print "Hallo " & strVorname & " " & strName & "!"  
  
End Sub
```

Übung: Variable



Ü1.2

- Deklarieren Sie zwei Variablen vom Typ Integer
- Initialisieren Sie die Variablen mit unterschiedlichen Werten
- Implementieren Sie einen Wertetausch der beiden Variablen
 - Hat Variable A den Wert W1 hat und Variable B den Wert W2,
 - soll nach dem Wertetausch A den Wert W2 und B den Wert W1 haben
- Geben Sie die Variablen vor und nach dem Wertetausch im Direktbereich aus
- Ihr Ergebnis könnte so aussehen:

```
Direktbereich
Variable A:
5
Variable B:
2
Variable A:
2
Variable B:
5
```

Übung: Variable



Ü1.2 - Lösungsvorschlag

```
Sub Wertetausch()  
  
    ' Zwei Variablen für Werte  
    Dim intValue1 As Integer  
    Dim intValue2 As Integer  
  
    ' Eine Hilfsvariable  
    Dim intValue3 As Integer  
  
    Let intValue1 = 5  
    Let intValue2 = 2  
  
    ' Ausgabe des Zustands vor Tausch  
    Debug.Print "Variable A:"  
    Debug.Print intValue1  
  
    Debug.Print "Variable B:"  
    Debug.Print intValue2  
  
    ' rechts gehts weiter -->
```

```
' <-- Fortsetzung von links  
  
' Variablenwerte tauschen  
' So geht es nicht!  
' Let intValue1 = intValue2  
' Let intValue2 = intValue1  
' Es geht nur mit Hilfsvariable  
Let intValue3 = intValue1  
Let intValue1 = intValue2  
Let intValue2 = intValue3  
  
' Ausgabe des Zustands nach Tausch  
Debug.Print "Variable A:"  
Debug.Print intValue1  
  
Debug.Print "Variable B:"  
Debug.Print intValue2  
  
End Sub
```

Übung: Verzweigungen



Ü1.3

- Deklarieren Sie eine Variable vom Typ Byte für den Wochentag
 - der Wert 1 soll Montag entsprechen, der Wert 2 Dienstag, ...
- Initialisieren Sie die Variable mit einer beliebigen Zahl
- Implementieren Sie eine Verzweigung die den Name des Wochentags ausgibt
- Wenn die Variable einen Wert > 7 hat, soll "Ungültiger Wochentag" ausgegeben werden

Übung: Verzweigungen



Ü1.3 – Lösungsvorschlag A

```
Sub WochentagNameA()  
  Dim bytWochentag As Byte  
  Dim strWochentagName As String  
  Let bytWochentag = 1  
  
  Select Case bytWochentag  
    Case 1  
      Let strWochentagName = "Montag"  
    Case 2  
      Let strWochentagName = "Dientag"  
    Case 3  
      Let strWochentagName = "Mittwoch"  
    Case 4  
      Let strWochentagName = "Donnerstag"  
    Case 5  
      Let strWochentagName = "Freitag"  
    Case 6  
      Let strWochentagName = "Samstag"  
    Case 7  
      Let strWochentagName = "Sonntag"  
    Case Else  
      Let strWochentagName = "Ungültiger Wochentag"  
  End Select  
  
  Debug.Print "Tag " & bytWochentag & " entspricht " & strWochentagName  
End Sub
```

Übung: Verzweigungen



Ü1.3 – Lösungsvorschlag B

```
Sub WochentagNameB()  
    Dim bytWochentag As Byte  
    Dim strWochentagName As String  
  
    Let bytWochentag = 5  
  
    If bytWochentag = 1 Then  
        Let strWochentagName = "Montag"  
    ElseIf bytWochentag = 2 Then  
        Let strWochentagName = "Dientag"  
    ElseIf bytWochentag = 3 Then  
        Let strWochentagName = "Mittwoch"  
    ElseIf bytWochentag = 4 Then  
        Let strWochentagName = "Donnerstag"  
    ElseIf bytWochentag = 5 Then  
        Let strWochentagName = "Freitag"  
    ElseIf bytWochentag = 6 Then  
        Let strWochentagName = "Samstag"  
    ElseIf bytWochentag = 7 Then  
        Let strWochentagName = "Sonntag"  
    Else  
        Let strWochentagName = "Ungültiger Wochentag"  
    End If  
  
    Debug.Print "Tag " & bytWochentag & " entspricht " & strWochentagName  
End Sub
```

Übung: Schleifen



Ü1.4

- Schreiben Sie eine Zählerschleife, alle geraden Zahlen im Bereich von 0 bis 20 addiert
- Geben Sie die Summe im Direktbereich aus

Ü1.5

- Implementieren Sie die Aufgabenstellung aus Ü1.4 mit einer vorprüfenden/kopfgesteuerten Schleife anstelle der Zählerschleife

Ü1.6

- Passen Sie Ihre Lösung aus Ü1.5 so an, dass die Schleife nachprüfend ist und rückwärts von 20 bis 0 läuft

Übung: Schleifen



Ü1.4 – Lösungsvorschlag

```
Sub ZählerSchleifenAddition()  
  
Dim bytZaehler As Byte  
Dim intSumme As Integer  
  
Let intSumme = 0  
  
For bytZaehler = 0 To 20 Step 2  
    Let intSumme = intSumme + bytZaehler  
    'Debug.Print "Zähler: " & bytZaehler  
    'Debug.Print "Zwischensumme: " & intSumme  
Next  
  
Debug.Print intSumme  
  
End Sub
```


Übung: Schleifen



Ü1.5 – Lösungsvorschlag

```
Sub VorprfdSchleifenAddition()  
  
Dim bytZaehler As Byte  
Dim intSumme As Integer  
  
Let bytZaehler = 0  
Let intSumme = 0  
  
Do While bytZaehler <= 20  
    Let intSumme = intSumme + bytZaehler  
    'Debug.Print "Zähler: " & bytZaehler  
    'Debug.Print "Zwischensumme: " & intSumme  
    Let bytZaehler = bytZaehler + 2  
Loop  
  
Debug.Print intSumme  
End Sub
```

Übung: Schleifen



Ü1.6 – Lösungsvorschlag

```
Sub NachprfdSchleifenAddition()  
  
Dim bytZaehler As Byte  
Dim intSumme As Integer  
  
Let intSumme = 0  
Let bytZaehler = 20  
  
Do  
    Let intSumme = intSumme + bytZaehler  
    'Debug.Print "Zähler: " & bytZaehler  
    'Debug.Print "Zwischensumme: " & intSumme  
    Let bytZaehler = bytZaehler - 2  
Loop While bytZaehler >= 2  
  
Debug.Print intSumme  
End Sub
```

Übung: Prozedur



Ü1.7: Schreiben Sie

- in einem Modul Bestellungen
- eine Prozedur, die einen Währungsbetrag als Netto übergeben bekommt,
- die Steuer (19 Prozent) und den Brutto-Betrag errechnet und
- Netto, Steuer und Brutto-Betrag im Direktbereich ausgibt
- Rufen Sie die Prozedur aus einer anderen Prozedur mit verschiedenen Beispielwerten auf

Übung: Prozedur



Ü1.7 – Lösungsvorschlag (Teil 1)

```
Option Compare Database
Option Explicit

Const bytSteuersatz = 19

Sub ausgebenBetrag(pcurNetto As Currency)
    Dim curBrutto As Currency
    Dim curSteuer As Currency
    Let curSteuer = (pcurNetto * bytSteuersatz) / 100
    Let curBrutto = pcurNetto + curSteuer

    Debug.Print "Netto : " & pcurNetto
    Debug.Print "MwSt  : " & curSteuer
    Debug.Print "-----"
    Debug.Print "Brutto: " & curBrutto
End Sub
' Fortsetzung auf der nächsten Folie
```

Übung: Prozedur



Ü1.7 – Lösungsvorschlag (Teil 2)

```
' Fortsetzung der vorherigen Folie

Sub Beispiele()

    Debug.Print vbNewLine & "Beispiel 1"
    ausgebenBetrag 100
    Debug.Print vbNewLine & "Beispiel 2"
    ausgebenBetrag 50
    Debug.Print vbNewLine & "Beispiel 3"
    ausgebenBetrag 25
    Debug.Print vbNewLine & "Beispiel 4"
    ausgebenBetrag 10

End Sub
```

Übung: Funktion



Ü1.8: Schreiben Sie

- in einem Modul Bestellungen
- eine Funktion, die einen Währungsbetrag als Netto übergeben bekommt,
- die Steuer (19 Prozent) errechnet und als Ergebnis zurückgibt
- verwenden Sie in der aufrufenden Prozedur dieses Ergebnis, um Netto, Steuer und Brutto-Betrag im Direktbereich auszugeben

Übung: Funktion



Ü1.8 – Lösungsvorschlag (Teil 1)

```
Option Compare Database
Option Explicit

Const bytSteuersatz = 19

'Variante A
Function berechneSteuerA(pcurNetto As Currency) _
    As Currency

    Dim curSteuer As Currency
    Let curSteuer = (pcurNetto * bytSteuersatz) / 100
    Let berechneSteuerA = curSteuer

End Function

' Fortsetzung auf der nächsten Folie
```

Übung: Funktion



Ü1.8 – Lösungsvorschlag (Teil 3)

```
'Variante B
Function berechneSteuerB(pcurNetto As Currency) _
    As Currency

    Let berechneSteuerB = (pcurNetto * bytSteuersatz) / 100

End Function

' Fortsetzung auf der nächsten Folie
```


Übung: Funktion



Ü1.8 – Lösungsvorschlag (Teil 3)

```
Sub Beispiele18()  
  Dim curNetto As Currency, curBrutto As Currency  
  Dim curSteuer As Currency  
  
  Debug.Print vbNewLine & "Beispiel 1"  
  Let curNetto = 100  
  Let curSteuer = berechneSteuerA(curNetto)  
  Let curBrutto = curNetto + curSteuer  
  Debug.Print "Netto : " & curNetto  
  Debug.Print "MwSt  : " & curSteuer  
  Debug.Print "-----"  
  Debug.Print "Brutto: " & curBrutto  
  
  Debug.Print vbNewLine & "Beispiel 2"  
  Let curNetto = 50  
  Let curSteuer = berechneSteuerB(curNetto)  
  Let curBrutto = curNetto + curSteuer  
  Debug.Print "Netto : " & curNetto  
  Debug.Print "MwSt  : " & curSteuer  
  Debug.Print "-----"  
  Debug.Print "Brutto: " & curBrutto  
End Sub
```

Übung: Oberflächenelement Eigenschaften



Ü1.9

- Erstellen Sie das folgende Formular eines kleinen Taschenrechners aus geeigneten Oberflächenelementen
- Setzen Sie das Feld "Ergebnis" des Taschenrechner-Formulars aus der vorherigen Ü1.9 auf inaktiv (gesperrt), so dass ein dort eingegebener Wert nicht verändert werden kann
- Entfernen Sie im Formular den Datensatzmarkierer und die Navigationsschaltflächen
- Beispiel:

The screenshot shows a window titled "frmRechner" with a blue header bar. Below the header, there are three input fields labeled "Zahl1", "Zahl2", and "Ergebnis". The "Zahl1" and "Zahl2" fields are empty and have a light blue border. Between "Zahl1" and "Zahl2" is a plus sign "+". Between "Zahl2" and "Ergebnis" is an equals sign "=" inside a blue button. The "Ergebnis" field is empty and has a light blue border. The entire form is set against a white background.

Übung: Oberflächenelement Eigenschaften



Ü1.9 – Lösungsvorschlag

- sehen Sie sich auch die Musterlösung in der bereitgestellten Access-Datenbank an

The screenshot shows the 'frmRechner' form in Design View. The 'Detailbereich' contains three text boxes labeled 'Zahl1', 'Zahl2', and 'Ergebnis', and three buttons labeled 'Ungebunden', '+', and 'Ungebunden', followed by an '=' button and another 'Ungebunden' button. The Properties window is open, showing the 'Format' tab. The 'Auswahltyp' is 'Formular'. The 'Datensatzmarkierer' property is set to 'Nein' and is circled in red.

Format	Daten	Ereignis	Andere	Alle
Rahmenart				Veränderbar
Datensatzmarkierer				Nein
Navigationsschaltflächen				Nein
Navigationsschriftart				
Trennlinien				Nein
Bildlaufleisten				In beide Richtungen

The screenshot shows the 'frmRechner' form in Design View. The 'Detailbereich' contains three text boxes labeled 'Zahl1', 'Zahl2', and 'Ergebnis', and three buttons labeled 'Ungebunden', '+', and 'Ungebunden', followed by an '=' button and another 'Ungebunden' button. The Properties window is open, showing the 'Ereignis' tab. The 'Auswahltyp' is 'Textfeld'. The 'Aktiviert' property is set to 'Ja' and is circled in red.

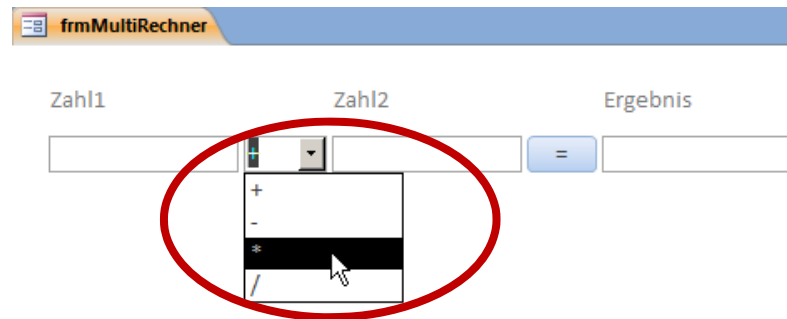
Format	Daten	Ereignis	Andere	Alle
Gültigkeitsmeldung				
Filter anwenden				Datenbankstandard
Aktiviert				Ja
Gesperrt				Ja
Smarttags				

Übung: Formulare und Ereignisse



Ü1.10

- Ändern Sie die Oberfläche des Taschenrechners so, dass anstelle des "+"-Operators eine Aufklappliste mehrere Operationen (z.B. Addition, Subtraktion, Multiplikation) anbietet



- Erstellen Sie eine Ereignisprozedur,
 - die beim Klick auf die Schaltfläche "=" zunächst prüft, welchen Wert die Aufklappliste hat
 - das Ergebnis der Operation berechnet und in das Ergebnisfeld schreibt

Übung: Formulare und Oberflächenelemente



Ü1.10 – Lösungsvorschlag (Teil 1)

- Lösung in Form von Folien nicht sinnvoll darstellbar
- Sehen Sie sich die Musterlösung in der bereitgestellten Access-Datenbank an

Übung: Formulare und Oberflächenelemente



Ü1.10 – Lösungsvorschlag (Teil 2)

– Ereignisprozedur

```
Option Compare Database
Option Explicit

Private Sub btnErgebnis_Click()

    Dim dblZahl1 As Double
    Dim dblZahl2 As Double
    Dim dblErgebnis As Double

    Dim strOperator As String

    Let dblZahl1 = Val(Me.txtZahl1.Value)
    Let dblZahl2 = Val(Me.txtZahl2.Value)
    Let dblErgebnis = 0

    Let strOperator = Me.cmbOperator.Value

    ' Fortsetzung links -->
```

```
' <-- Fortsetzung von rechts

Select Case strOperator
    Case "+"
        dblErgebnis = dblZahl1 + dblZahl2
    Case "-"
        dblErgebnis = dblZahl1 - dblZahl2
    Case "*"
        dblErgebnis = dblZahl1 * dblZahl2
    Case "/"
        dblErgebnis = dblZahl1 / dblZahl2
End Select

Let txtErgebnis.Value = dblErgebnis

End Sub
```

Module, Prozeduren/Funktionen: Übung Ü1.10



Ü1.10: Modul Kunden

- Legen Sie ein Modul mdlKunden an
- Deklarieren Sie im Modul eine private Variable lngKundenNr vom Typ Long, die innerhalb des Moduls gültig ist.
- eine öffentliche Prozedur "setzeAktuellerKundeld", die als Parameter eine ID vom Typ Long übergeben bekommt und den Wert der Variable lngKundenNr zuweist
- eine öffentliche Funktion "gibAktuellerKundeld", die den Wert der Variablen lngKundenNr zurückliefert

Module, Prozeduren/Funktionen: Übung Ü1.11



Ü1.11: Modul Benutzer

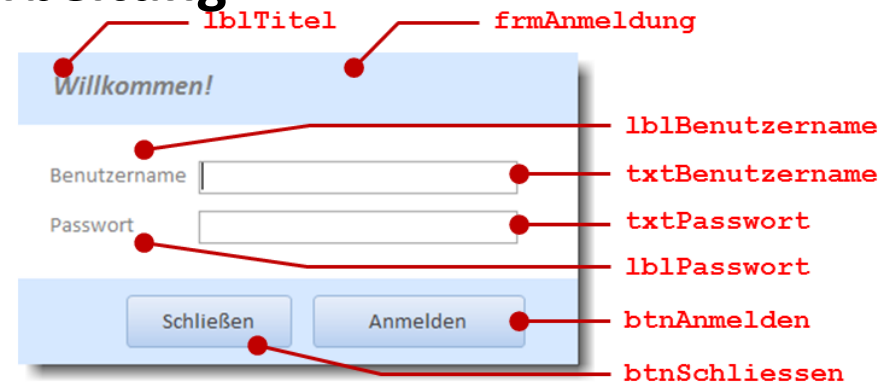
- Legen Sie ein Modul mdlBenutzer an
- Deklarieren Sie im Modul eine private Variable bollstBenutzerAdmin vom Typ Boolean (Sie wird später benötigt, um zwischen normalen Kunden und Administratoren zu unterscheiden.)
- Schreiben Sie eine öffentliche Funktion istBenutzerAdmin(), die den Wert dieser Variable zurückliefert
- Schreiben Sie eine öffentliche Funktion anmelden,
 - die einen Benutzernamen und ein Passwort als Parameter übergeben bekommt und
 - den Wert der bollstBenutzerAdmin mit False initialisiert
 - einen Wahrheitswert zurückliefert.
 - Wenn der eingegebene Benutzername „user“ und das Passwort „test“ ist, soll die Prozedur mdlKunden.setzeAktuellerKundeID mit dem Wert 1 aufgerufen werden, und die Funktion soll als Rückgabewert true zurückliefern.
 - Andernfalls soll die Prozedur mdlKunden.setzeAktuellerKundeID mit dem Wert -1 aufgerufen werden, und die Funktion false zurückliefern.

Oberflächen und Ereignisse: Übung Ü 1.12



Ü1.12: Oberfläche und Ereignisverarbeitung

- Erstellen Sie ein Formular zur Anmeldung eines Benutzers. Geben Sie den Formularelemente aussagekräftige Namen.
- Schreiben Sie eine Ereignisprozedur, die aufgerufen wird, sobald auf die Schaltfläche „Anmelden“ geklickt wird. Rufen Sie aus dieser Ereignisprozedur die Funktion anmelden (aus Übung Ü1.11) auf, übergeben Sie dabei den Text aus den Feldern „Benutzername“ und „Passwort“ und speichern Sie den Rückgabewert in einer Variablen vom Typ Boolean.
- Zeigen Sie in einem Meldungsfenster
 - eine Willkommensmeldung, wenn der Rückgabewert der Funktion anmelden true ist und schließen Sie das Formular.
 - eine Fehlermeldung, wenn der Rückgabewert false und leeren Sie das Feld "Passwort", indem Sie ihm Null zuweisen
- Hinweis: Gehen Sie zunächst davon aus, dass der Benutzer immer Text in die Felder eingibt.



Oberflächen und Ereignisse: Übung Ü1.13



Ü1.13 (Zusatz)

- Implementieren Sie eine Ereignisprozedur, die ausgeführt wird, wenn der Benutzer im Anmeldedialog aus Übung Ü1.12 auf Schließen klickt
 - Zeigen Sie eine Meldung an, ob der Benutzer wirklich schließen möchte oder nicht
 - Wenn ja, dann schließen Sie das Fenster. Wenn nicht, lassen Sie es offen.

Ü1.14 (Zusatz)

- Ändern Sie die Implementierung aus Übung Ü1.12, so dass der Benutzer eine Fehlermeldung bekommt, wenn nicht beide Felder "Benutzername" und "Passwort" gefüllt sind.

Ü1.15 (Zusatz)

- Ändern Sie die Eigenschaften des Passwortfeldes aus Übung Ü1.12, so dass es das Passwort nicht mehr im Klartext zeigt
- Ändern Sie die Eigenschaften des Formulars, so dass es als Dialog (Popup: Ja, Rahmenart: Dialog, Bildlaufleisten: Nein) angezeigt wird
- Datensatzmarkierer und Navigationsschaltflächen ausblenden