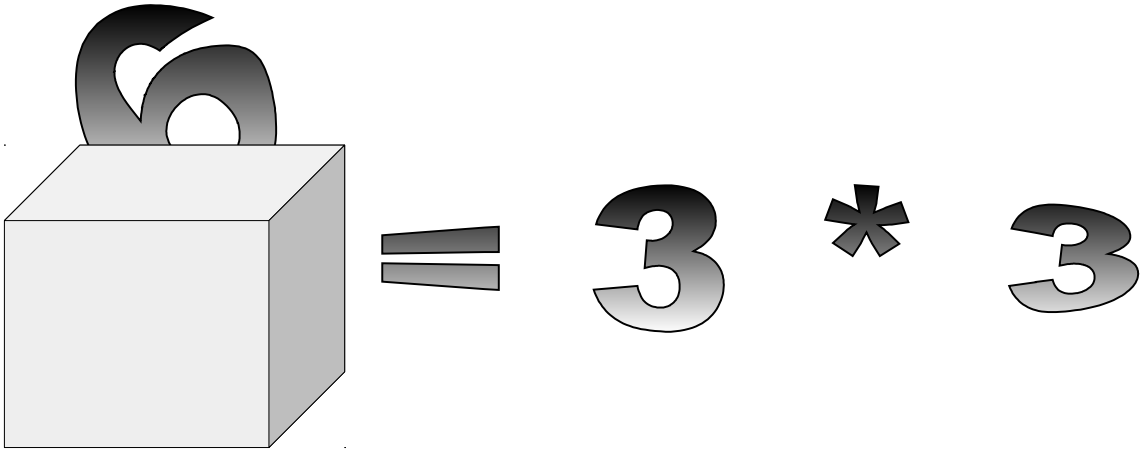
V(isual) B(asic for) A(pplication)
Datenzugriff in Microsoft Access



Module

- Dateien zur Zusammenfassung von Code zu einem Thema.
- Funktionale Einheit eines VBA-Projekts.
- Wiederverwendung von Code.

Standardmodule

- Verwendung in der prozeduralen Programmierung.
- Dateien mit der Endung *.bas.
- Zusammenfassung von Subroutinen zu einem Thema.
- Deklaration von globalen Variablen, die im gesamten Projekt genutzt werden.

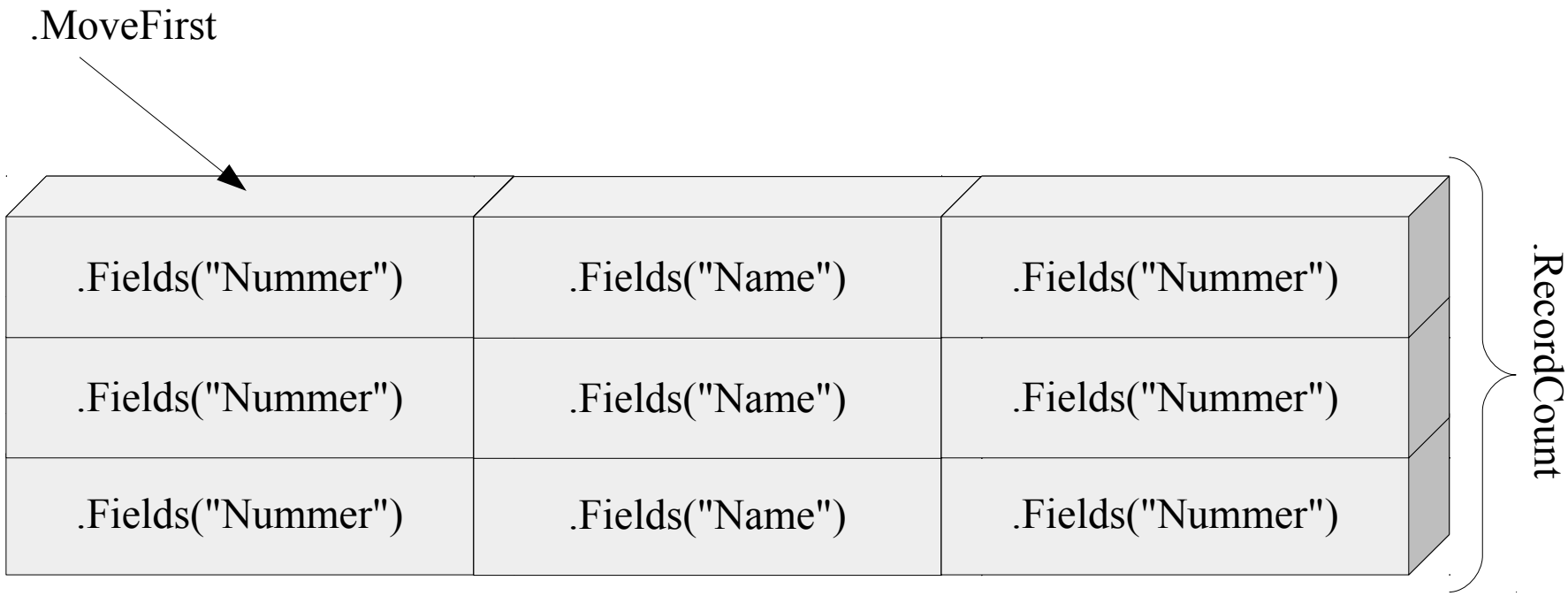
Klassenmodule

- Allgemeine Beschreibung von Objekten.
- Attribute und Subroutinen einer Objekt-Kategorie.
- Dateien mit der Endung „.cls“.
- Vordefinierte Module zur Beschreibung von Microsoft Access Objekten wie zum Beispiel der Datenzugriff auf Tabellen.
- Benutzerdefinierte Module zur Beschreibung von Objekten wie zum Beispiel Kunden etc.

Klasse

- Abstraktion von Dingen aus der realen Welt.
- Allgemeine Beschreibung von einem Microsoft Access Objekt. Welche Eigenschaften hat ein Objekt? Wie verhält sich das Objekt?
- Bauplan für ein bestimmtes Ding zum Beispiel für ein Formular, ein Bericht etc.
- Formale Beschreibung einer Gruppe.
- Vorlage für die Erzeugung eines Objektes.

Klasse „Datensätze (Zeilen) einer Tabelle“



Attribute

- Beschreibung eines Gegenstandes, einer Person, etc.
- Allgemeingültige Beschreibung für ein Microsoft Access Objekt.
- „Variablen“ eines Objekts.
- Jedes Objekt einer Klasse hat die gleichen Attribute. Zum Beispiel hat ein Recordset die Eigenschaft RecordCount (Anzahl der Datensätze).
- Die Ausprägung der Attribute kann sich bei den verschiedenen Objekten unterscheiden. Zum Beispiel hat Tabelle A 5 Datensätze, aber Tabelle B 2 Datensätze.

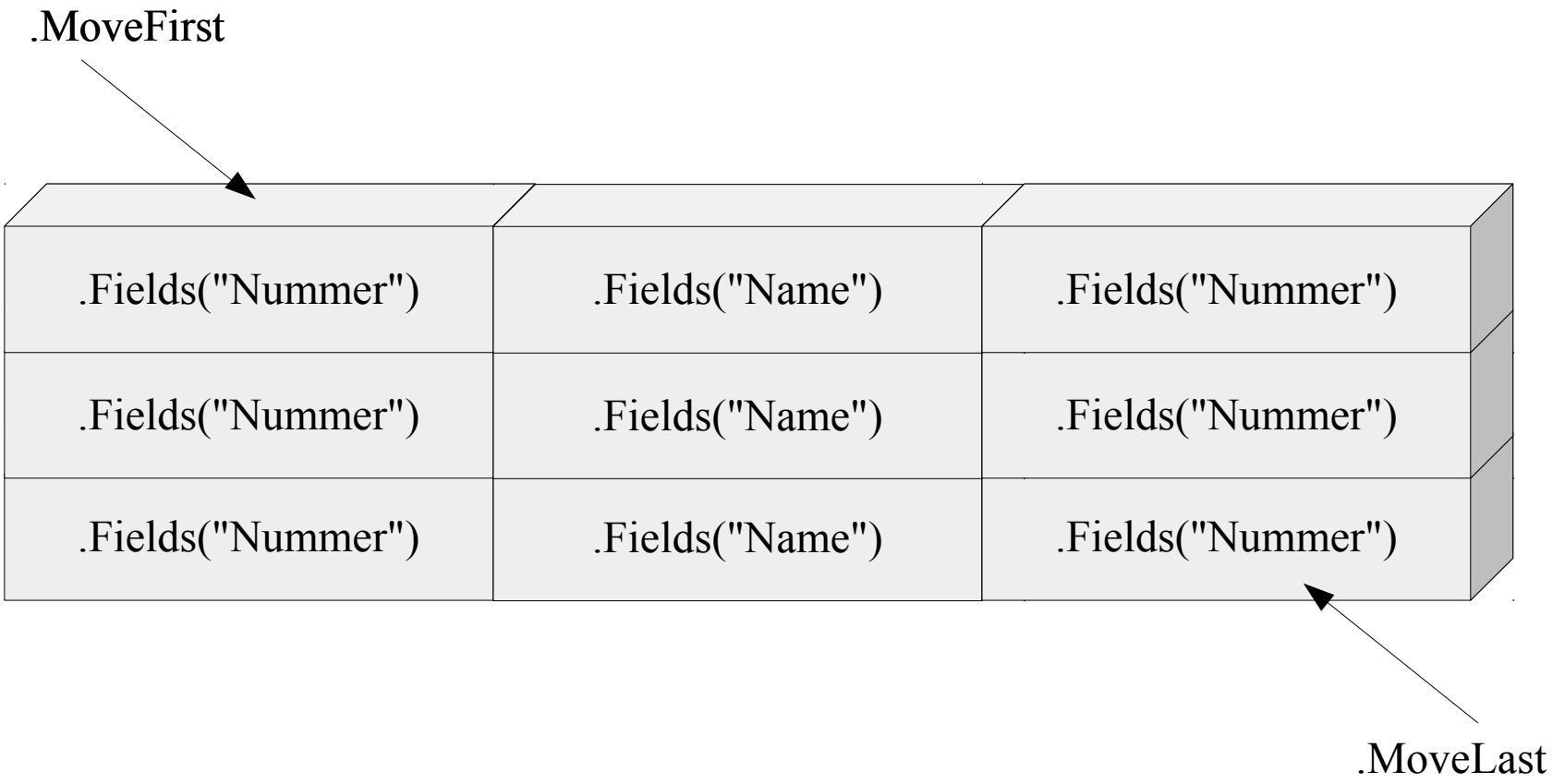
Beispiel


.Fields("Nummer")
.Fields("Name")
.Fields("Nummer")
.Fields("Nummer")
.Fields("Name")
.Fields("Nummer")
.Fields("Nummer")
.Fields("Name")
.Fields("Nummer")
.RecordCount

Methoden

- Was kann ein Objekt machen?
- Beschreibung der Funktionalität eines Objekts.
- Veränderungen von Objekten von außen her.
- Prozeduren, die in Klassen gekapselt sind.
- Alle Objekte einer Klasse haben die gleichen Methoden.
- Zum Beispiel wird mit Hilfe von .MoveNext oder .MoveLast der Datensatzzeiger einer Tabelle verändert.

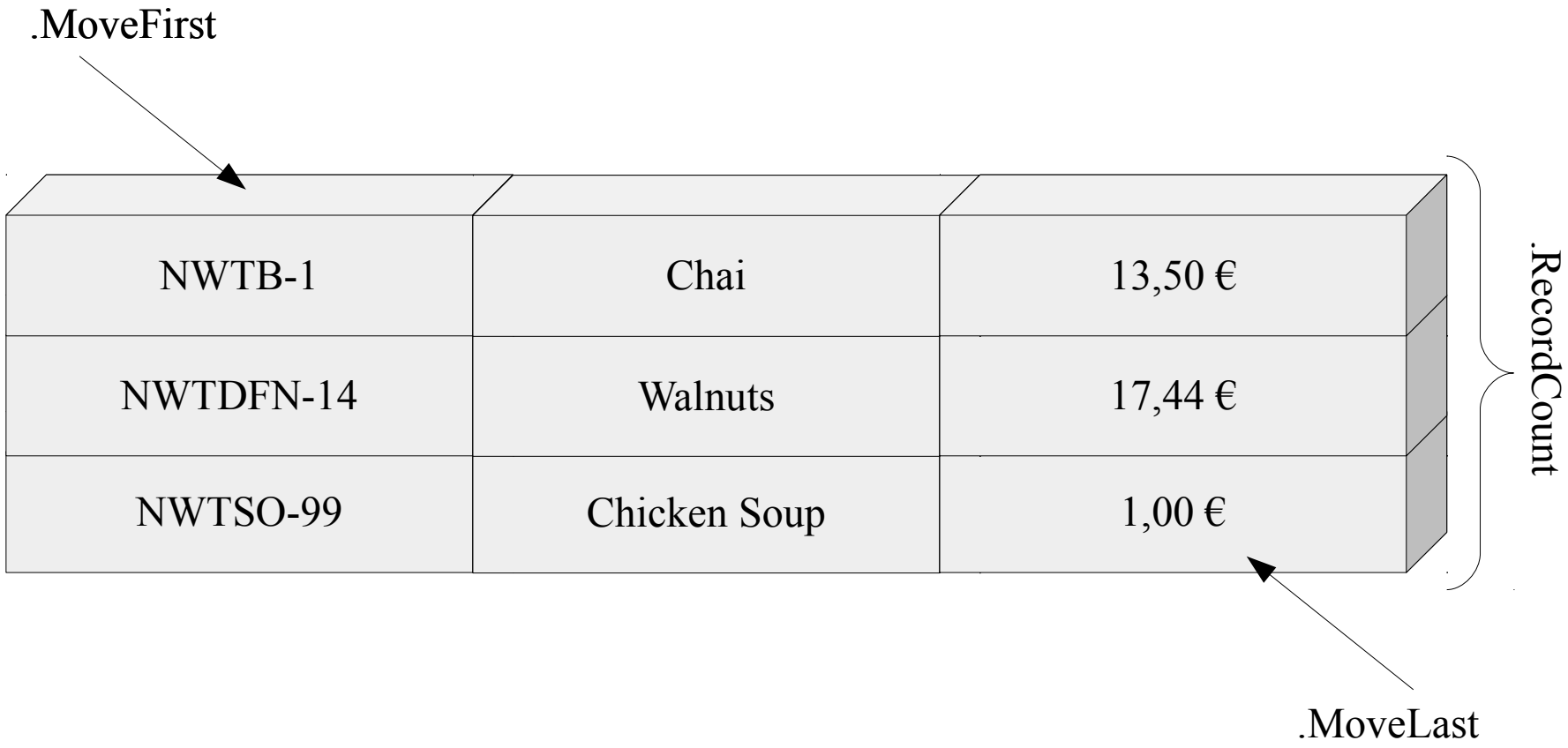
Klasse „Recordset“



Objekte

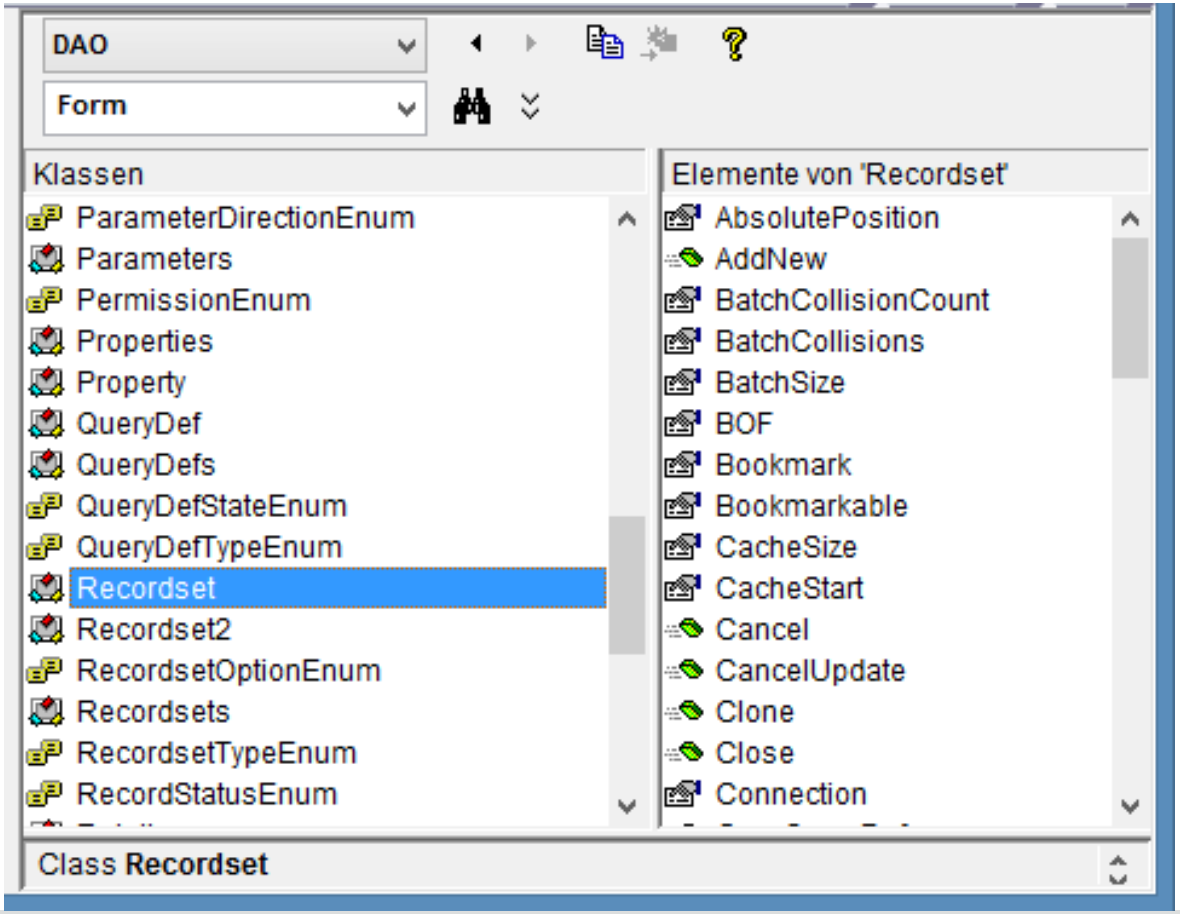
- Ein Ding (Exemplar, Instanz) aus der realen Welt.
- Objekte wie Formulare und Berichte in Microsoft Access.
- Eine abgeschlossen Einheit.
- Alle Elemente einer Kategorie von Dingen haben die gleichen Attribute, aber in unterschiedlichen Ausprägungen. Sie nutzen die gleichen Methoden zum Ändern ihrer Attributwerte.

Objekt „Datensätze (Zeilen) der Tabelle ...“



Informationen zu Objekten

- … im VBA-Editor: *Ansicht – Objektkatalog*.



Aufbau des Objektkatalogs

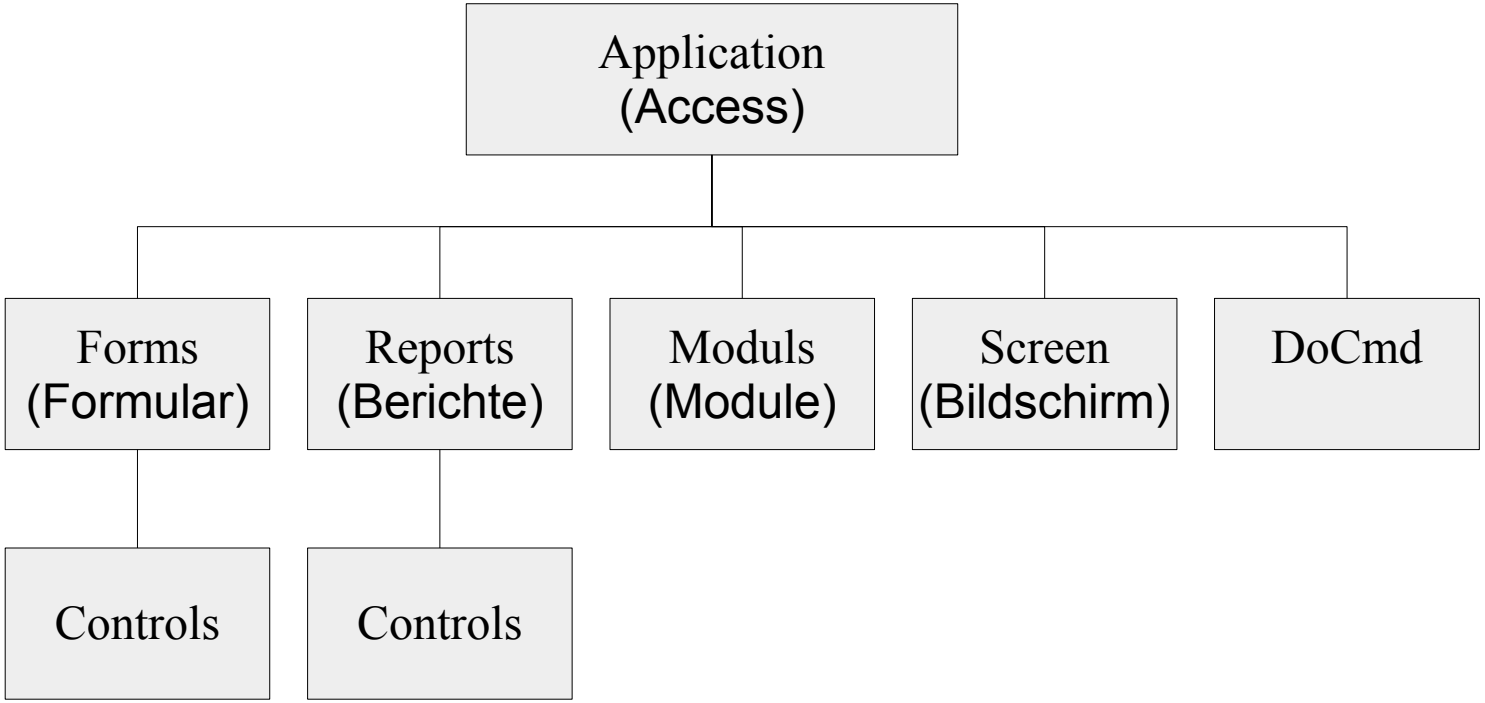
- Klassen werden in einer Bibliothek wie Bücher zusammengefasst. In diesem Beispiel ist die Bibliothek DAO ausgewählt.
- In dem linken Listenfeld werden alle Klassen in der gewählten Bibiothek angezeigt. In dem Beispiel ist die Klasse Recordset ausgewählt.
- In dem rechten Listenfeld werden alle Elemente einer Klasse angezeigt. In diesem Beispiel werden Methoden und Attribute aus der Klasse Recordset dargestellt. Die Methode .AddNew erzeugt einen neuen Datensatz, das Attribut .Bookmark symbolisiert ein Lesezeichen und so weiter

Elemente von Klassen im Objektkatalog

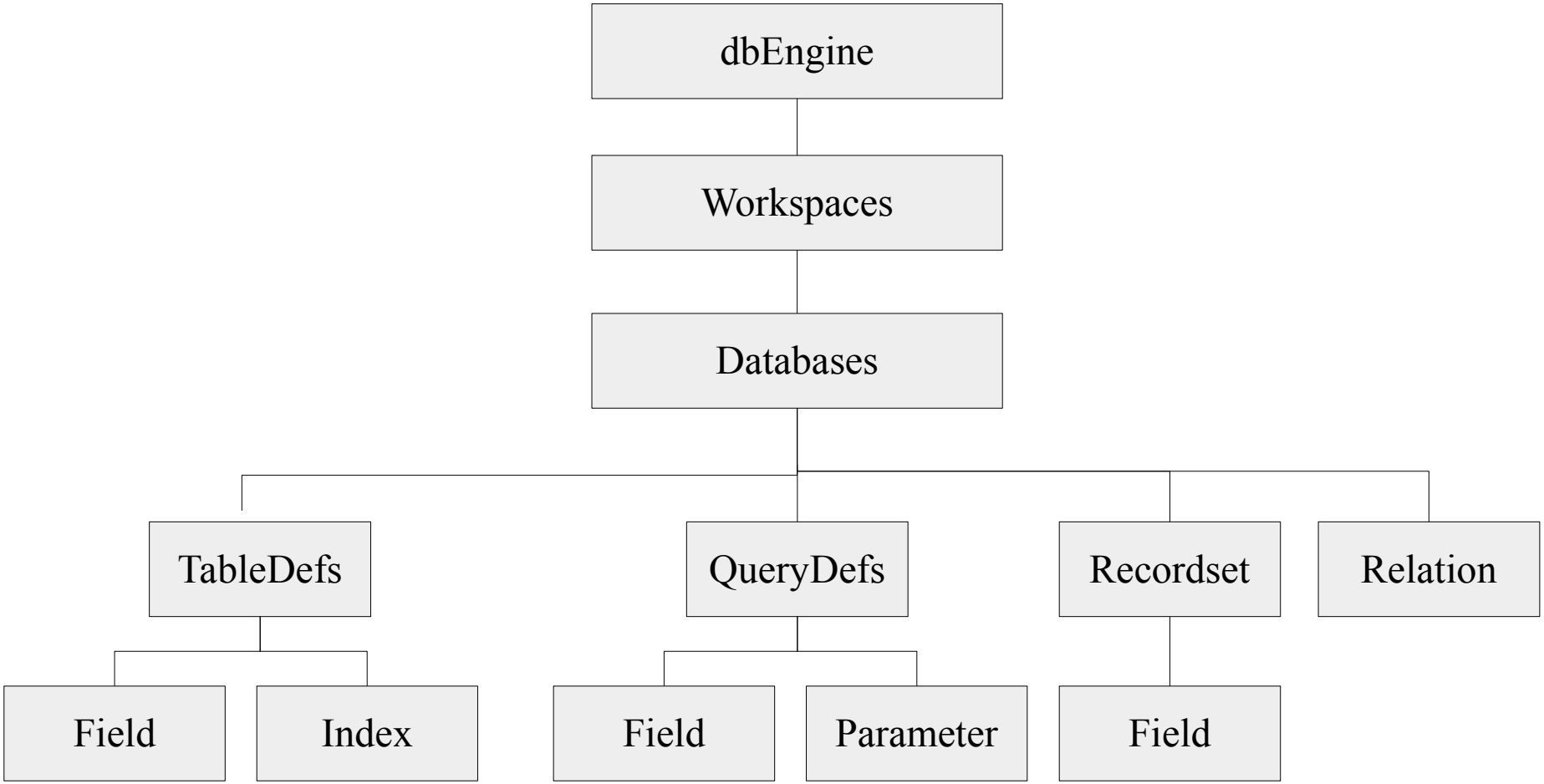
Objektmodell

- Hierarchische Anordnung von Objekten.
- Darstellung von Objekten und deren Abhängigkeit untereinander in einem Modell.
- Bibliotheken fassen Objekte in einem Modell zusammen.

Modell „Microsoft Access“

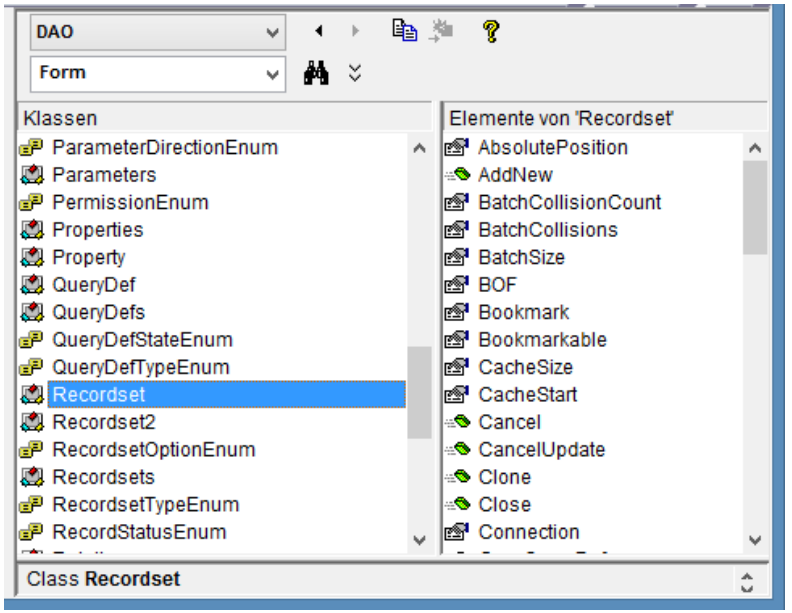


Modell „D(ata)A(ccess)O(bject)“ (Ausschnitt)

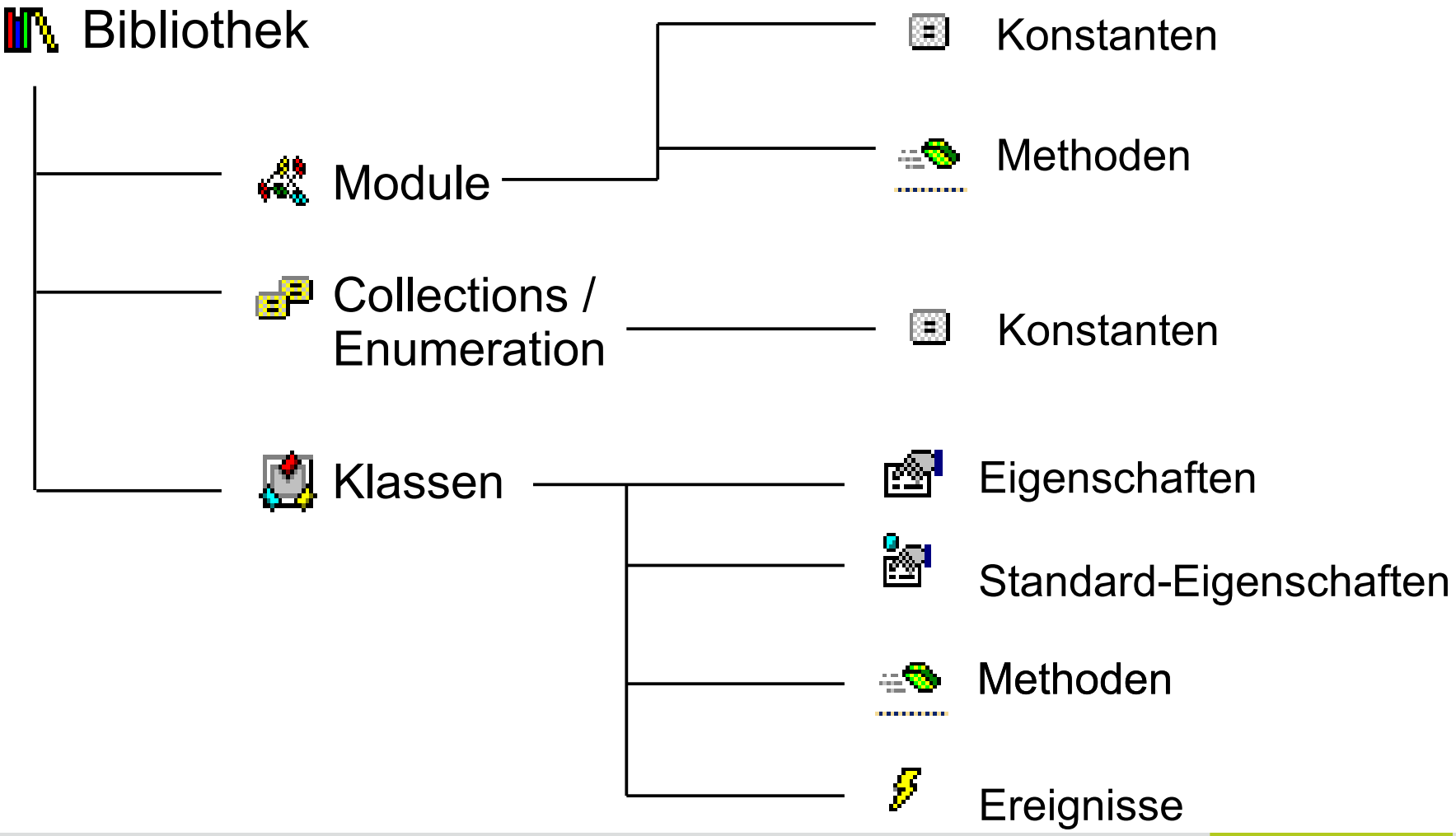


Informationen zu Objektmodellen

- … im VBA-Editor: *Ansicht – Objektkatalog*.
- Mit Hilfe des DropDown-Feldes am oberen linken Rand wird eine Bibliothek ausgewählt. In diesem Beispiel ist das Modell DAO ausgewählt.
- Entsprechend der gewählten Bibliothek werden die dazugehörigen Klassen in der linken Liste angezeigt.

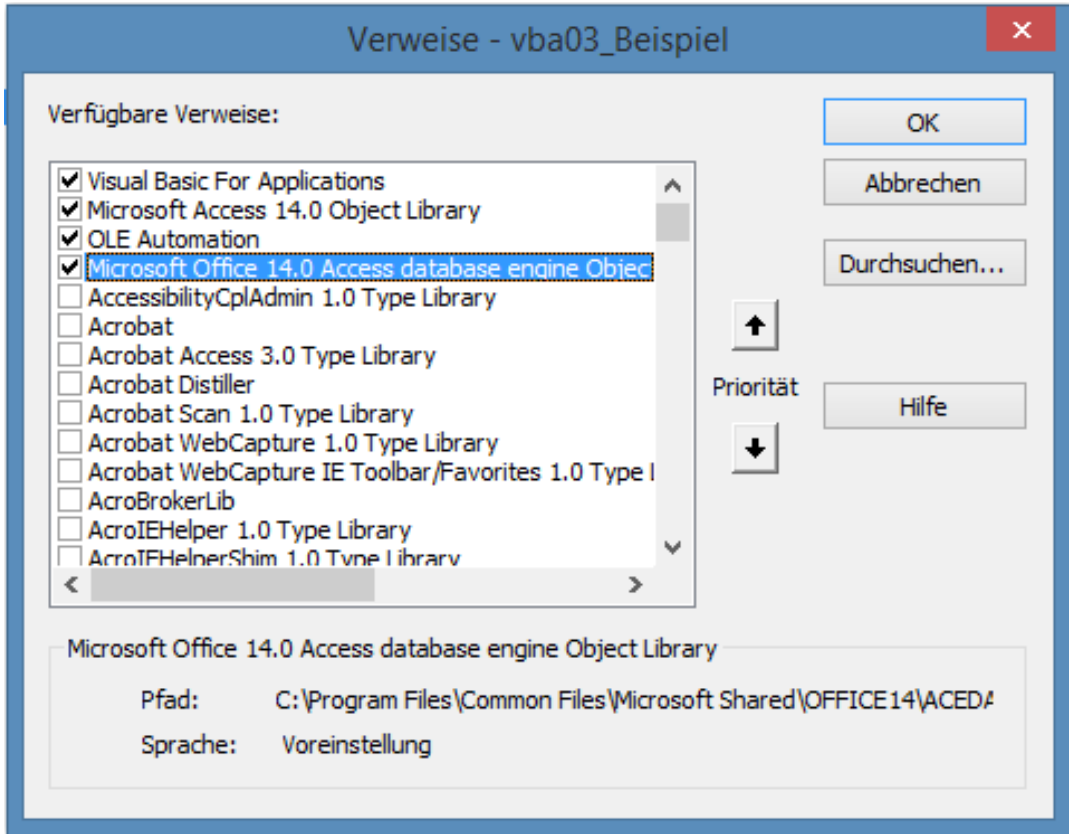


Elemente im Objektkatalog



Verweise auf Bibliotheken

- … im VBA-Editor: *Extras* – *Verweise*.



Standardverweise

- *Visual Basic for Application*. Die Bibliothek beschreibt alle Elemente der Programmiersprache VBA.
- *OLE Automation*. Einbetten oder Verlinken von Objekten wie zum Beispiel Bilder.
- *Microsoft Access 14.0 Object Library*. Beschreibung der Objekte von Microsoft Access. Die Version 14.0 bezieht sich auf Microsoft Access 2010.
- *Microsoft Office 14.0 Access database engine Object Library.* Beschreibung der Datenbank selbst und deren Datenzugriff darauf. Die Bibliothek benutzt die Jet-Datenbank Version 4.0 (.mdb) oder die Version 12.0 (accdb).

Einfügung von Verweisen

- *Extras – Verweise*.
- In der Liste wird zum Beispiel zu dem Eintrag *Microsoft DAO 3.6 Object Library* geblättert. Hinweis: Die Liste ist alphabetisch sortiert.
- Klick in das Kontrollkästchen links vom Namen des Verweises. Das Häkchen in einem Kästchen kennzeichnet den aktiven Verweis.
- *OK* schließt das Fenster.

D(ata)A(ccess)O(bject)

- Arbeiten mit der Datenbank.
- Datenzugriff auf Tabellen und Abfragen in Microsoft Access.
- Die Bibliothek *Microsoft DAO 3.6 Object Library* muss für Microsoft Access 2007 und älter eingebunden werden.
- Seit Microsoft Access 2010 Standardbibliothek unter dem Namen *Microsoft Office 14.0 Access database engine Object Library.*

Objekt-Variable

```
Sub datenzugriff()

    Dim dbs As DAO.Database
    Dim rs As DAO.Recordset

End Sub
```

- Die Variable speichert einen Verweis auf ein Objekt vom Typ „Klasse“.
- Instanzen von einer Klasse aus einer bestimmten Bibliothek.
- Platzhalter für eine Referenz auf ein konkretes Objekt.

… deklarieren

Dim	dbs	As	DAO	.	Database
Dim	[name]	As	[bibliothek]	.	[klasse]

- Der Variablenname ist frei wählbar. Der gewählte Name sollte aber das zu speichernde Objekt widerspiegeln.
- Dem Schlüsselwort As folgt die Klasse. Die Klasse spiegelt den gewünschten Objekttyp wieder.

… von der Klasse

Dim	dbs	As			Database
Dim	dbs	As	DAO	.	Database
Dim	[name]	As	[bibliothek]	.	[klasse]

- As folgt immer ein Klassenname. In diesem Beispiel Database.
- Der Klassenname gibt über die Objektkategorie Auskunft.

… in der Bibliothek

Dim	dbs	As	DAO	.	Database
Dim	[name]	As	[bibliothek]	.	[klasse]

- Der Klassenname kann mit einem Bibliotheksnamen verbunden werden.
- Der Punkt-Operator verbindet den Bibliotheksnamen, in dem die gewünschte Klasse definiert ist, mit dem Klassennamen.
- Der Bibliotheksname kann weggelassen werden, wenn die Klasse eindeutig einer Bibliothek zugeordnet werden kann.

Initialisierung

```
Dim dbs As DAO.Database

Set dbs = CurrentDb
```

- Mit Hilfe des Gleichheitszeichens wird der Variablen eine Referenz auf ein bestimmtes Objekt zugewiesen.
- Die Objekt-Variable verweist auf ein konkretes Objekt.

Erläuterung

Set	dbs	=	CurrentDB
Set	[name]	=	[objekt]

- Die Zuweisung beginnt mit dem Schlüsselwort Set (setze).
- Dem Schlüsselwort folgt der Name.
- Mit Hilfe des Gleichheitszeichens wird der Variablen ein Verweis auf ein Objekt übergeben. In diesem Beispiel wird ein Verweis auf die aktuelle Datenbank (CurrentDB) übergeben.

Objekt-Variable leeren

```
Dim dbs As DAO.Database

Set dbs = Application.CurrentDb
Set dbs = Nothing
```

- Das Schlüsselwort Nothing ist ein Platzhalter für „kein Verweis“.
- Die Objekt-Variable verweist auf kein Objekt.

Objekt-Variable vom Typ „Datenbank“

```
Dim dbs As DAO.Database
```

- Die Objekt-Variable kann auf Objekte von der Klasse DAO.Database verweisen.
- Die Objekt-Variable kann auf eine Microsoft Access-Datenbank oder eine ODBC-Datenbank verweisen.

Verweis auf die aktuell geöffnete Datenbank

```
Dim dbs As DAO.Database

Set dbs = Application.CurrentDb
```

- Die Methode CurrentDb der Klasse Application gibt ein Verweis auf die aktuelle Datenbank zurück.
- Der Verweis auf die aktuelle Datenbank wird in diesem Beispiel in der Objekt-Variable dbs gespeichert.

Klassen und Methoden

```
Dim dbs As DAO.Database

Set dbs = Application.CurrentDb
Set dbs = CurrentDb
```

- Die Methode CurrentDb der Klasse Application gibt ein Verweis auf die aktuelle Datenbank zurück.
- Klassen und die darin definierten Methoden werden mit dem Punktoperator verbunden.
- Wenn die Methode eindeutig einer Klasse zugeordnet werden kann, kann der Klassenname entfernt werden.

Speicherort der aktuellen Datenbank

```
Dim speicherort As String

speicherort = Application.CurrentProject.Path
```

- Das Wurzelelement Application (Microsoft Access) hat das Objekt CurrentProject (das aktuelle Projekt).
- Das Objekt CurrentProject hat das Attribut Path.
- Die Eigenschaft Path gibt den Pfad des aktuell geöffneten Projekts (der Microsoft Access - Datei) wieder.

Klassen und Attribute

```
Dim speicherort As String

speicherort = Application.CurrentProject.Path
```

- Klassen und die darin definierten Attribute werden mit dem Punktoperator verbunden.

Abbildung des Objektmodells

```
Dim speicherort As String

speicherort = Application.CurrentProject.Path
```

- Mit Hilfe des Punktoperators wird die Hierachie in einem Objektmodell abgebildet.
- In diesem Beispiel wird das Nachfolger-Element CurrentProject dem Vorgänger-Element Application zugeordnet.
- Die erste Klasse in der Auflistung spiegelt die Wurzel des Objektmodells wieder. In diesem Beispiel ist die Klasse Application die Wurzel des Objektmodells „Microsoft Access“.

Verweis auf eine externe Datenbank

```
Dim wks As DAO.Workspace
Dim dbs As DAO.Database
Dim speicherort As String

speicherort = Application.CurrentProject.Path
speicherort = speicherort & "\" & "vba04_Beispiel_Backend.accdb"

Set wks = DBEngine.Workspaces(0)
Set dbs = wks.OpenDatabase(speicherort)

dbs.Close
Set dbs = Nothing
Set wks = Nothing
```

Schritte

- Zuerst wird ein Arbeitsbereich für die zu öffnende Datenbank definiert.
- Im nächsten Schritt wird die Datenbank geöffnet.
- Falls die Datenbank nicht mehr gebraucht wird, wird die Datenbank geschlossen.

1. Schritt: Definition eines Arbeitsbereichs

```
Dim wks As DAO.Workspace
Set wks = DBEngine.Workspaces(0)
Set wks = Nothing
```

- Die Variable wks kann einen Verweis auf einen Arbeitsbereich / Benutzersitzung (DAO.Workspace) speichern.
- Der Ausdruck DBEngine.Workspaces(0) ist ein Verweis auf den Standard-Arbeitsbereich von Microsoft Access. Der Benutzername im Standard-Arbeitsbereich lautet „Admin“.

dbEngine

Set	wks	=	dbEngine

- Eigenschaft der Application (der Microsoft Access Anwendung).
- Wurzel-Element des Modells „DAO“.
- Platzhalter für die Datenbank.

Arbeitsbereich

Set	wks	=	dbEngine	.	Workspaces	(	0	)

- Die Hierachie der Objekte wird durch den Punkt symbolisiert.
- Das Objekt Workspaces ist ein Kind-Element von dbEngine.
- Der Buchstabe „s“ am Ende des Namens kennzeichnet eine Sammlung (Collection). Sammlungen sind null-basiert. Die 0 in den runden Klammern symbolisiert das erste Element in der Sammlung.

2. Schritt: Öffnen der Datenbank

```
Dim dbs As DAO.Database

Set dbs = wks.OpenDatabase(speicherort)
```

- Die Variable dbs kann einen Verweis auf eine Datenbank (DAO.Database) speichern.
- Die Methode .OpenDatabase ist in DAO.Workspace definiert und öffnet eine Datenbank in dem Standard-Arbeitsbereich.
- In den runden Klammern wird der Methode der Name Datenbank übergeben. Der Pfad zu der Datenbank muss absolut angegeben werden.

3. Schritt: Schließen der Datenbank

```
Dim dbs As DAO.Database

Set dbs = wks.OpenDatabase(pfad)
dbs.close
```

- Mit Hilfe der Methode .close wird eine, von VBA geöffnete Datenbank geschlossen.
- Falls die aktuelle Datenbank mit Hilfe von Application.CurrentDb geöffnet wird, wird die Methode nicht benötigt.

Objekt-Variablen und Methoden

Set	dbs	=	wks	.	OpenDatabase	(	pfad	)
			dbs	.	Close			

- Objekt-Variablen und Methoden werden mit einem Punkt verbunden.
- Die Methode rechts vom Punkt verändert die Instanz links vom Punkt.
- Die Objekt-Variable ist vom Typ einer Klasse … In dieser Klasse ist Methode (rechts vom Punkt) definiert.
- Der Methodenname ist in der Klasse eindeutig.
- Dem Methodenname folgt eine Parameterliste.

Parameterliste von Methoden

Set	dbs	=	wks	.	OpenDatabase	(	pfad	)
			dbs	.	Close			

- Dem Methodenname folgt eine Parameterliste.
- In der Parameterliste werden der Methode Parameter als Start-Werte für in der Methode definierte Variablen übergeben. Der Methode OpenDatebase wird der Name und Speicherort der zu öffnenden Datenbank übergeben.
- Die Parameter werden durch Kommata getrennt.
- Die Anzahl der Parameter sind abhängig von dem definierten Methodenkopf.

Leere Parameterliste von Methoden

Set	dbs	=	wks	.	OpenDatabase	(	pfad	)
			dbs	.	Close			

- Die Klammern können leer sein. Der Methode werden keine Parameter übergeben.
- Die Klammern der Parameterliste können weggelassen werden.

Objekt-Variable vom Typ „Recordset“

```
Dim rs As DAO.Recordset
```

- Die Objekt-Variable kann auf Objekte von der Klasse DAO.Recordset verweisen.
- Die Objekt-Variable kann auf Datensätze (Zeilen) in Tabellen und Abfragen verweisen.

Öffnen eines Recordsets

```
Set rs = dbs.OpenRecordset(tblName, dbOpenTable)

rs.Close
```

- Mit Hilfe der Methode .OpenRecordset() werden Datensätze zum Lesen und Schreiben geöffnet.
- Falls das zu öffnende Element nicht vorhanden ist, wird ein Laufzeitfehler angezeigt.

Erläuterung

Set	rs	=	dbs	.	OpenRecordset	(	tblName	,	dbOpenTable	)

- Direkt auf den Methodennamen folgt die Parameterliste.
- Die einzelnen Parameter werden durch Kommata getrennt.
- Erster Parameter: Wo sind die Datensätze gespeichert? In diesem Beispiel wird ein Tabellenname genutzt.
- Zweiter Parameter: Wie erfolgt der Zugriff? Mit Hilfe von Konstanten wird der Zugriff angegeben. In diesem Beispiel wird einen Tabelle geöffnet. Der Nutzer hat Lese- und Schreibrechte.

Möglichkeiten für den Zugriff

- dbOpenTable. Standardzugriff, wenn keine Angaben gemacht wurden. Öffnen einer lokalen Tabelle.
- dbOpenDynaset. Öffnen einer verknüpften Tabelle oder einer Abfrage. Die Abfrage kann in Microsoft Access definiert sein oder direkt mit Hilfe von SQL angegeben werden.
- dbOpenSnapshot. „Momentaufnahme“ von Datensätzen. Nur lesenden Zugriff.

Schließen eines Recordsets

```
rs.Close
```

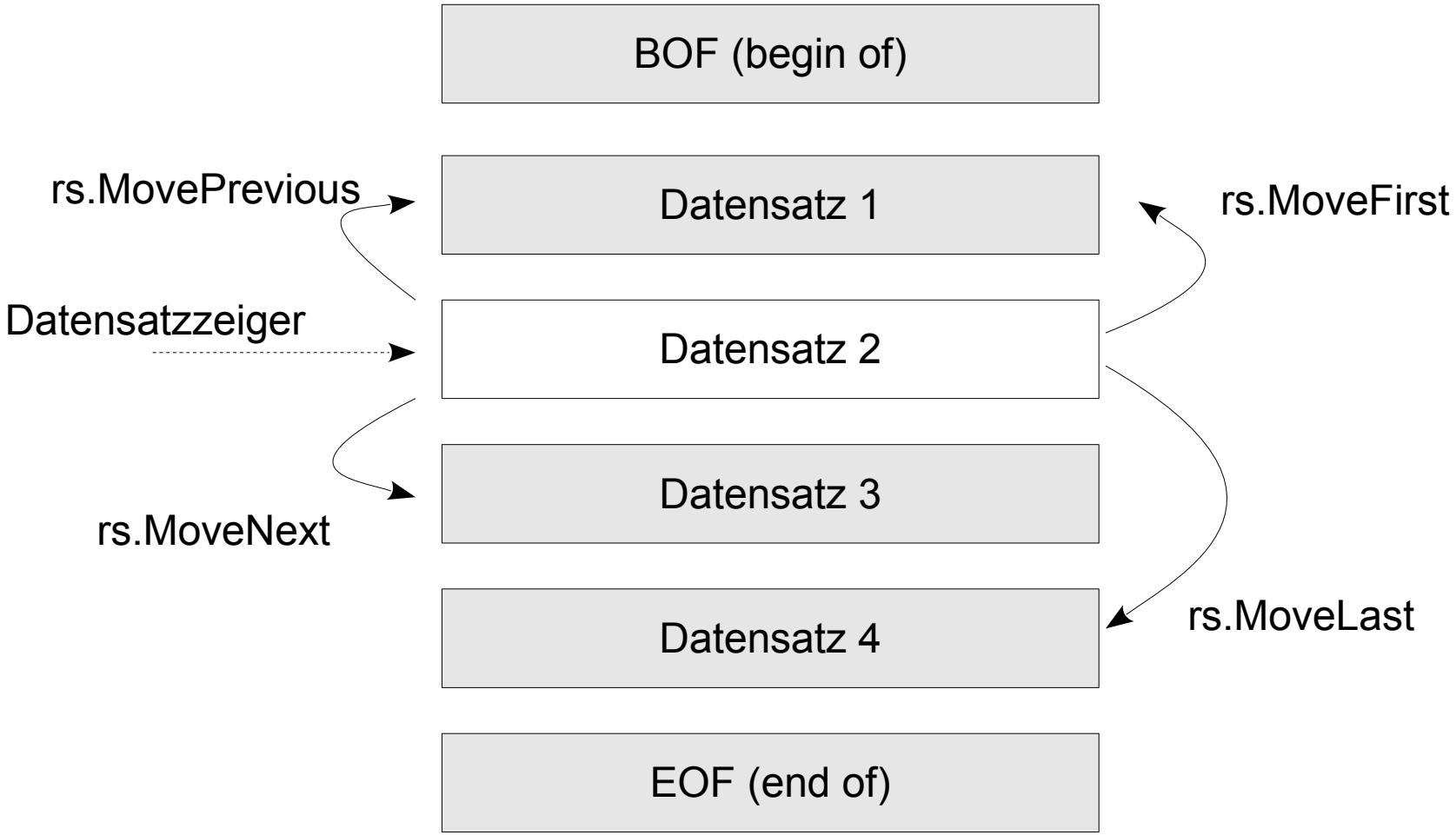
- Sobald die Datensätze nicht mehr benötigt werden, wird die Tabelle mit der Methode .close geschlossen.

Anzahl der Datensätze

```
Debug.Print "Anzahl der Datensätze: " & rs.RecordCount
```

- Hinweis: Die Datensätze wurden mit Hilfe der Konstanten dbOpenTable geöffnet.
- Das Attribut .RecordCount enthält die Anzahl der Datensätze der geöffneten Tabelle.

Navigation in den Datensätzen



Felder in einem Datensatz

```
produktnummer = rs.Fields("Produktcode")
produktnummer = rs.Fields(3)
```

- Jeder Datensatz hat eine bestimmte Anzahl von Datenfeldern.
- Die Datenfelder können mit Hilfe der Sammlung .Fields eines Recordset durchlaufen werden.
- Die Sammlung .Fields enthält alle Feldnamen (siehe Entwurfsansicht einer Tabelle)

… mit Hilfe des Index lesen

```
produktnummer = rs.Fields(3)
```

- .Fields ist eine Sammlung von Datenfeldern.
- Dem Namen der Sammlung folgen direkt die runden Klammern. In den runden Klammern wird ein Index angegeben.
- Ein Index einer Sammlung ist immer eine Ganzzahl.
- Auflistungen sind null-basiert. Das erste Element hat den Index 0, das zweite Element 1 und so weiter.
- Falls das Element in der Sammlung nicht vorkommt, wird der Fehler „Element in der Auflistung nicht gefunden“ angezeigt.

… mit Hilfe des Namens lesen

```
produktnummer = rs.Fields("Produktcode")
```

- Jedes Element in der Sammlung .Fields kann über den Namen angesprochen werden.
- Der Name des Datenfeldes wird in den runden Klammern der Sammlung als String übergeben.
- Der Name kennzeichnet eindeutig ein Datenfeld in dem geöffneten Datensatz.

Weitere Schreibweise

```
produktnummer = rs.Fields("Produktcode")
produktnummer = rs!Produktcode

stueckzahlMin = rs![Mindeststückzahl für Nachbestellung]
```

- Die Objekt-Variable wird vom Namen des Datenfeldes durch ein Ausrufezeichen getrennt.
- Falls der benutzerdefinierte Name Sonderzeichen wie zum Beispiel das Leerzeichen oder Umlaute enthält, muss der Name mit Hilfe der eckigen Klammern zusammengefasst werden.

Punkt-Operator

- Rechts vom Punkt-Operator befindet sich ein, in Microsoft Office definiertes Element.
- Objekt-Variablen und deren Eigenschaften und Methoden werden verbunden.
- Die Hierarchie in einem Objektmodell wird abgebildet.
- Bibliotheken und deren darin enthaltenen Klassen werden verknüpft.

Nutzung des Ausrufezeichens

- Verbindung von Objekten und deren Kind-Elementen.
- Rechts vom Ausrufezeichen werden benutzerdefinierte Namen von Datenfelder, Objekten etc. genutzt.

Wert eines Datenfeldes

```
produktnummer = rs.Fields("Produktcode").Value
produktnummer = rs.Fields(3).Value
produktnummer = rs!Produktcode.Value
```

- Die Eigenschaft .value gibt den Wert des Datenfeldes zurück.
- Es wird der, in einem Datenfeld gespeicherte Wert gelesen.

Hinzufügung eines Datensatzes

```
rs.AddNew

rs.Fields("Produktcode").Value = "NWTMN-1"
rs.Fields("Artikelname").Value = "Neuer Artikel"

rs.Update
```

- Datensätze werden immer am Ende einer Tabelle eingefügt.
- Datenfelder vom Typ „AutoWert“ werden automatisiert gesetzt.

1. Schritt: Datensatz anlegen

```
rs.AddNew
```

- Der Datensatz wird am Ende der Tabelle neu angelegt.
- Die Datenfelder werden mit den angegebenen Standardwerten belegt.

2. Schritt: Datenfelder füllen

```
rs.AddNew

rs.Fields("Produktcode").Value = "NWTMN-1"
rs.Fields("Artikelname").Value = "Neuer Artikel"
```

- Links vom Zuweisungsoperator steht das zu füllende Datenfeld. Das Datenfeld ist ein Synonym für eine Variable.
- Rechts vom Zuweisungsoperator steht der zu speichernde Wert. Der Wert kann als Literal im Code übergeben werden. Der Wert kann aber auch durch einen Ausdruck berechnet werden.

3. Schritt: Änderungen speichern

```
rs.AddNew

rs.Fields("Produktcode").Value = "NWTMN-1"
rs.Fields("Artikelname").Value = "Neuer Artikel"

rs.Update
```

- rs.Update speichert den neuen Datensatz.
- Der Datensatzzeiger zeigt auf den eingefügten Datensatz.

Änderungen an einem Datensatz

```
rs.Edit

rs.Fields("Beschreibung").Value = "Beschreibungstext"

rs.Update
```

- Mit Hilfe den entsprechenden Befehlen wird der Datensatzzeiger auf den zu ändernden Datensatz verschoben.

1. Schritt: Kopieren des Datensatzes

```
rs.Edit
```

- Der Datensatz wird zur Bearbeitung in einem Puffer gespeichert.

2. Schritt: Änderungen des Wertes

```
rs.Edit

rs.Fields("Beschreibung").Value = "Beschreibungstext"
```

- Das zu ändernde Datenfeld steht links vom Zuweisungsoperator.
- Rechts vom Zuweisungsoperator steht der zu speichernde Wert. Der Wert kann als Literal im Code übergeben werden. Der Wert kann aber auch durch einen Ausdruck berechnet werden.

3. Schritt: Änderungen speichern

```
rs.Edit

rs.Fields("Beschreibung").Value = "Beschreibungstext"

rs.Update
```

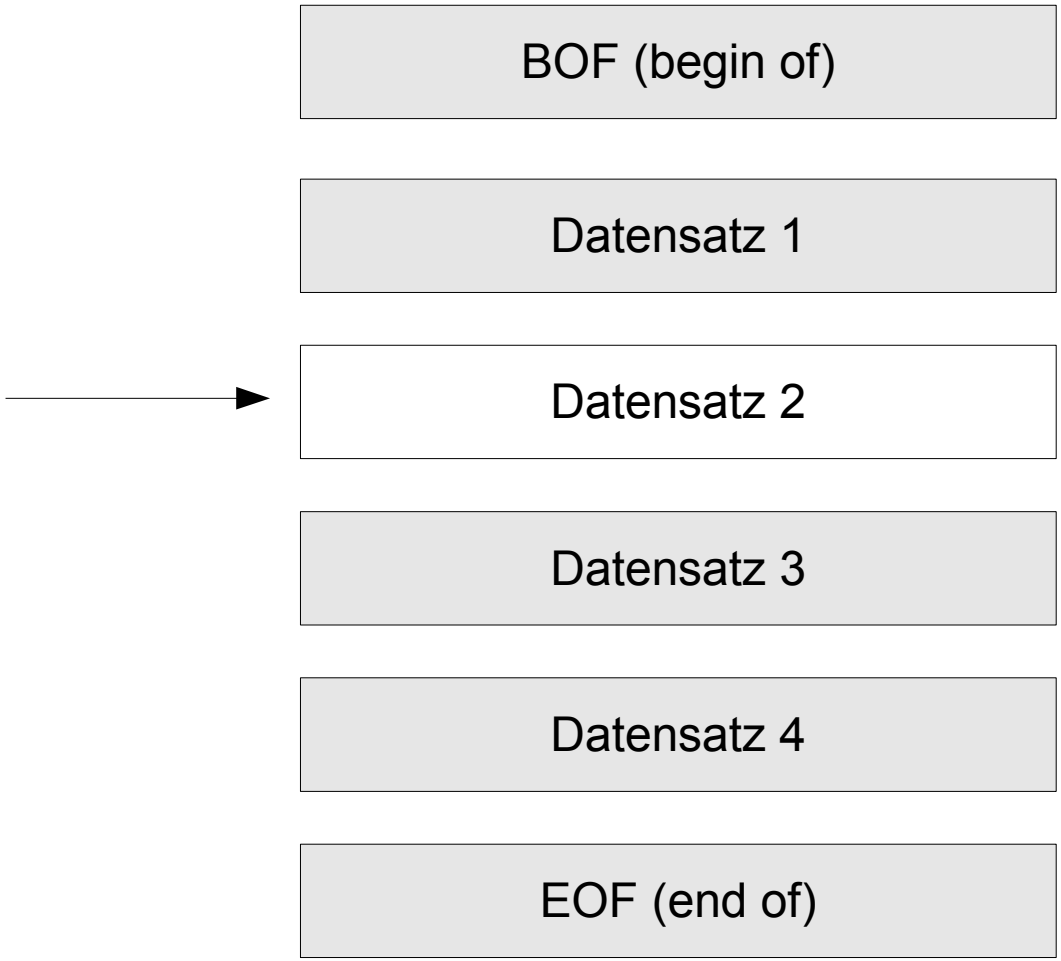
- rs.Update speichert den neuen Datensatz. Falls die Methode nicht aufgerufen wird, werden die Änderungen verworfen.
- Der Datensatzzeiger zeigt auf den geänderten Datensatz.

Löschung eines Datensatzes

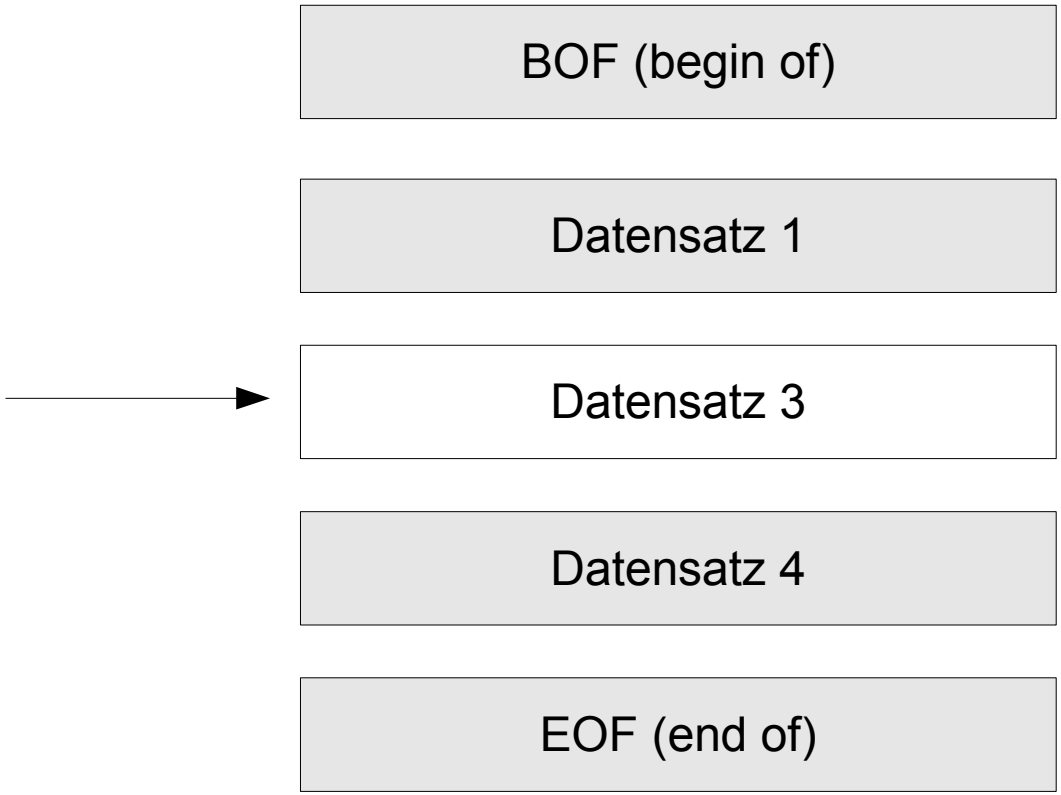
```
rs.MoveLast

rs.Delete
```

- Mit Hilfe den entsprechenden Befehlen wird der Datensatzzeiger auf den zu löschenden Datensatz verschoben.
- Die Methode .Delete löscht den aktuellen Datensatz ohne Nachfrage.
- Alle nachfolgenden Datensätze werden um eine Position nach oben verschoben.
- Der Datensatzzeiger zeigt aber auf die gleiche Position wie vor der Löschung, aber auf einen anderen Datensatz.

Datensatz-Zeiger vor der Löschung



Datensatz-Zeiger nach der Löschung



Objekt-Variable „Tabellendefinition“

```
Dim dbs As DAO.Database
Dim tblDef As DAO.TableDef
```

- Die Variable vom Datentyp DAO.TableDef speichert einen Verweis auf die Struktur einer Tabelle.
- Speicherung auf die Referenz einer Tabellendefinition.
- Die Struktur einer Tabelle wird in der Entwurfsansicht einer Microsoft Access – Tabelle abgebildet.

Tabellenstruktur lesen

```
Dim dbs As DAO.Database
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")
```

- Mit Hilfe von Application.CurrentDb.TableDefs("Artikel") wird der Objekt-Variablen tblDef ein Verweis auf die Struktur der Tabelle Artikel übergeben.

Erläuterung

- Die Objekt-Variable dbs ist ein Platzhalter für eine Datenbank.
- Jede Klasse vom Objekt Database besitzt eine Sammlung (Collection) von Tabellendefinition.
- Die Sammlung .TableDefs enthält zu jeder Tabelle in der gewählten Datenbank einen Eintrag.
- In den runden Klammern wird der Sammlung der Name der Tabelle als Index übergeben.
- Die Tabellendefinition des gewählten Elements wird als Verweis in der Objekt-Variablen gespeichert.

Neue Tabellenstruktur anlegen

```
Dim dbs As DAO.Database
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.CreateTableDef("KundenPlusBestellung")
```

- Mit Hilfe von .CreateTableDef("KundenPlusBestellung") wird eine neue leere Tabellenstruktur erzeugt.
- Der Methode wird der Name der zu erzeugenden Tabelle in den runden Klammern übergeben.
- Der VBA-Befehl entspricht *Erstellen – Tabellenentwurf* in der Gruppe *Tabellen*.

… und in die Datenbank einfügen

```
dbs.TableDefs.Append tblDef

Application.RefreshDatabaseWindow
```

- Mit Hilfe von .Append wird der Sammlung TableDefs ein Element hinzugefügt.
- Hinweis: Die Tabelle muss mindestens ein Feld enthalten.
- Die Methode .RefreshDatabaseWindow des Objekts Application aktualisiert das Datenbankfenster. Die neu erstellte Tabelle wird im Navigationsfenster angezeigt.

Alle Felder in einer Tabelle

```
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")

Debug.Print tblDef.Fields(0).Name
Debug.Print tblDef.Fields(1).Name
```

- Die Objektvariable tblDef hat ein untergeordnetes Objekt .Fields.
- Das untergeordnete Objekt ist eine Auflistung aller Felder in der angegebenen Tabelle.
- Die Definition der Tabellenspalten kann gelesen und verändert werden.

Anzahl der Felder

```
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")

Debug.Print tblDef.Fields.Count
```

- Das Attribut .Count gibt die Anzahl der Felder in der angegebenen Tabelle zurück.

Feld x in der Sammlung

```
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")

Debug.Print tblDef.Fields(0).Name
Debug.Print tblDef.Fields(1).Name
```

- Dem Namen der Sammlung .Fields folgt in runden Klammern ein Index.
- Mit Hilfe des Index wird eindeutig ein Element in der Sammlung identifiziert.
- Der Index ist immer vom Datentyp „Ganzzahl“.

Index eines Elements

```
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")

Debug.Print tblDef.Fields(0).Name
Debug.Print tblDef.Fields(1).Name
```

- Als Index wird immer eine Ganzzahl genutzt.
- Das erste Element hat den Index 0.
- Das letzte Element hat den Index tblDef.Fields.Count – 1.

Lesen der Feldnamen

```
Dim tblDef As DAO.TableDef

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("Artikel")

Debug.Print tblDef.Fields(0).Name
Debug.Print tblDef.Fields(1).Name
```

- Das Attribut .Name gibt den Namen des Elements in der Auflistung wieder.
- In diesem Beispiel wird der der erste und zweite Feldname einer Tabelle zurückgegeben.

Objekt-Variable vom Typ „Tabellenfeld“

```
Dim fld As DAO.Field
```

- Die Bibliothek .DAO enthält die Klasse .Field.
- Die Klasse ist Basis für Tabellenfelder einer Tabelle in Microsoft Access.

Neues Feld anlegen

```
Dim fld As DAO.Field

Set tblDef = dbs.CreateTableDef("KundenPlusBestellung")

Set fld = tblDef.CreateField("KundenName", dbText, 255)
```

- Mit Hilfe der Methode .CreateField() eines Objekts „Tabellendefinition“ wird ein neues Feld erzeugt.
- Der Methode wird als erster Parameter der Name des Feldes übergeben.

Parameter der Methode .CreateField()

```
Dim fld As DAO.Field

Set fld = tblDef.CreateField("KundenName", dbText, 255)
Set fld = tblDef.CreateField("IDBestellsumme", dbLong)
```

- Der Methode wird als erster Parameter der Name des Feldes übergeben.
- Der zweite Parameter legt den Datentyp des Feldes fest. In diesem Beispiel wird ein Feld vom Datentyp „Text“ und ein weiteres Feld vom Datentyp „Long“ angelegt. Eine Auflistung der Datentypen finden Sie unter https://msdn.microsoft.com/de-de/library/office/ff845405.aspx.

Erzeugung von Feldern vom Typ „Text“

```
Dim fld As DAO.Field

Set fld = tblDef.CreateField("KundenName", dbText, 255)
```

- Mit Hilfe der Konstanten dbText wird ein Feld vom Datentyp „Text“ angelegt.
- Der dritte Parameter legt die maximale Zeichenlänge fest.

Felder in die Sammlung „Fields“ einhängen

```
Dim fld As DAO.Field

Set fld = tblDef.CreateField("KundenName", dbText, 255)
tblDef.Fields.Append fld
```

- Die Methode .Append fügt in die Sammlung Fields ein passendes Objekt ein.
- Die Sammlung .Fields des Platzhalter tblDef wird verändert. Die Objekt-Variable tblDef verweist auf die zu ändernde Tabellendefinition.

Feld vom Datentyp „AutoInkrement“

```
Set fld = tblDef.CreateField("IDBestellsumme", dbLong)
fld.Attributes = dbAutoIncrField
tblDef.Fields.Append fld
```

- Das Feld ist vom Datentyp dbLong. Das Feld kann Ganzzahlen vom Datentyp Long speichern.
- Jedes Feld hat Attribute (Attributes). Mit Hilfe von Konstanten können die Attribute gesetzt werden.
- In diesem Beispiel wird bei Neuanlage des Datensatz das Feld automatisch auf einen eindeutigen Wert vom Typ Long gesetzt. Ein interner Zähler wird um eins hochgezählt-

Objekt-Variable vom Typ „Index“

```
Dim idx As DAO.Index
```

- Die Bibliothek .DAO enthält die Klasse .Index.
- Jedes Feld in einer Datenbank kann indexiert werden. Die Suche nach Werten wird durch einen Index beschleunigt.

Erzeugung eines Indizes

```
Dim dbs As DAO.Database
Dim tblDef As DAO.TableDef
Dim idx As DAO.Index

Set dbs = Application.CurrentDb
Set tblDef = dbs.TableDefs("KundenPlusBestellung")

Set idx = tblDef.CreateIndex("PrimaryKey")
```

- Mit Hilfe der Methode .CreateIndex wird ein neuer Index erzeugt.
- Der Methode wird in runden Klammern der eindeutige Name des Index übergeben.
- Der Objekt-Variablen wird ein Verweis auf den neu erzeugten Index übergeben.

Alle Indizies einer Tabelle

```
Dim tblDef As DAO.TableDef
Dim idx As DAO.Index

Set tblDef = dbs.TableDefs("KundenPlusBestellung")

Set idx = tblDef.CreateIndex("PrimaryKey")
tblDef.Indexes.Append idx
```

- Die Sammlung .Indexes eines Objekts „Tabellendefinition“ enthält alle Indizies.

Anhängen eines neuen Index

```
Dim tblDef As DAO.TableDef
Dim idx As DAO.Index

Set tblDef = dbs.TableDefs("KundenPlusBestellung")

Set idx = tblDef.CreateIndex("PrimaryKey")
tblDef.Indexes.Append idx
```

- Mit Hilfe der Methode .Append einer Sammlung wird der Auflistung .Indexes ein neuer Index hinzugefügt.
- In diesem Beispiel wird der Index PrimaryKey hinzugefügt.

Zusammenfassung von Anweisungen

```
With idx

    Set fld = .CreateField("IDBestellsumme")
    .Fields.Append fld
    .Unique = False
    .Primary = True

End With
```

Aufbau

```
With [objektvariable]
     [anweisung]
End With
```

Erläuterung

- Zusammenfassung von Anweisungen, die ein Objekt verändern.
- Dem Schlüsselwort With folgt das Objekt oder ein Platzhalter für das Objekt.
- Zwischen den Schlüsselwörtern With und End With folgen Anweisungen, die das Objekt verändern oder Methoden des Objekts nutzen.

Aufbau der Anweisungen

```
With idex
    Set fld = .CreateField("IDBestellsumme")
End With
```

```
Set fld = idex.CreateField("IDBestellsumme")
```

Erläuterung

- Dem Schlüsselwort With folgt ein Platzhalter für ein Objekt oder das Objekt selbst.
- Methoden und Attribute, die sich auf dieses Objekt beziehen, benötigen links vom Punktoperator keine Angabe des Objekts.
- Der Punktoperator zur Verknüpfung mit dem angegebenen Objekt wird aber benötigt.
- Die genutzten Methoden oder Attribute können rechts oder links vom Zuweisungsoperator stehen.

Eindeutiger Index?

```
Dim fld As DAO.Field
Dim idx As DAO.Index

Set idx = tblDef.CreateIndex("PrimaryKey")
idex.Unique = False
```

- Die Eigenschaft .Unique legt fest, ob der Index eindeutig ist oder nicht.

Setzen eines Primärschlüssels

```
Dim fld As DAO.Field
Dim idx As DAO.Index

Set idx = tblDef.CreateIndex("PrimaryKey")
idex.Primary = True
```

- Die Eigenschaft .Primary legt fest, ob der Index als Primärschlüssel genutzt wird oder nicht.
- Ein Primärschlüssel ist für jeden Datensatz eindeutig.

Erzeugung von Feldern für ein Index

```
Dim fld As DAO.Field
Dim idx As DAO.Index

Set idx = tblDef.CreateIndex("PrimaryKey")

Set fld = idex.CreateField("IDBestellsumme")
```

- Mit Hilfe der Methode .CreateField() der Klasse „Index“ wird ein Feld im Index erzeugt.
- Setzen der Feldeigenschaft *Indiziert* eines Feldes in der Entwurfsansicht eines Feldes.
- Der Methode wird der Feldname übergeben.
- Hinweis: Das Feld muss in der Tabelle existieren.

… und anhängen

```
Dim fld As DAO.Field
Dim idx As DAO.Index

Set idx = tblDef.CreateIndex("PrimaryKey")

Set fld = idex.CreateField("IDBestellsumme")
idex.Fields.Append fld
```

- Der Sammlung .Fields der Klasse „Index“ wird mit Hilfe der Methode .Append das neu erzeugte Feld angehängt.