Visual Basic 2012

Grundlagen der Programmierung

VBNET2012

Ricardo Hernández García

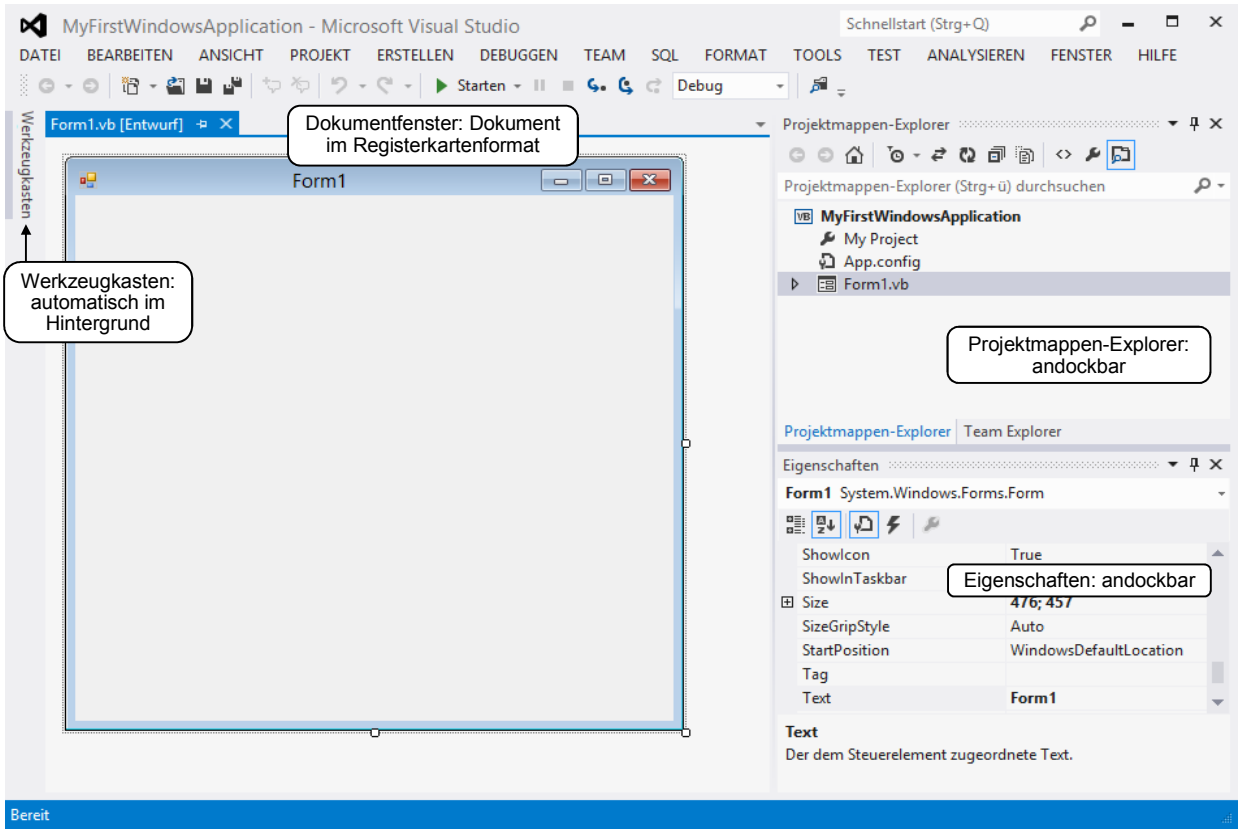
1. Ausgabe, März 2013

HERDT

3.4 Fenster in Visual Studio 2012 bedienen

Mit Fenstern arbeiten

Neben dem Arbeitsbereich enthält die Oberfläche mehrere zusätzliche Fenster, sogenannte **Toolfenster**, die Sie bei der Entwicklung Ihrer Anwendungen unterstützen.



Standardpositionen der Toolfenster

Um die Oberfläche Ihren Bedürfnissen anzupassen, können Sie Toolfenster …

- ✔ frei (unverankert) positionieren,
- ✔ am Rand eines anderen Fensters andocken (anheften),
- ✔ als Fenster im Registerkartenformat anordnen,
- ✔ automatisch in den Hintergrund legen,
- ✔ ein- bzw. ausblenden.

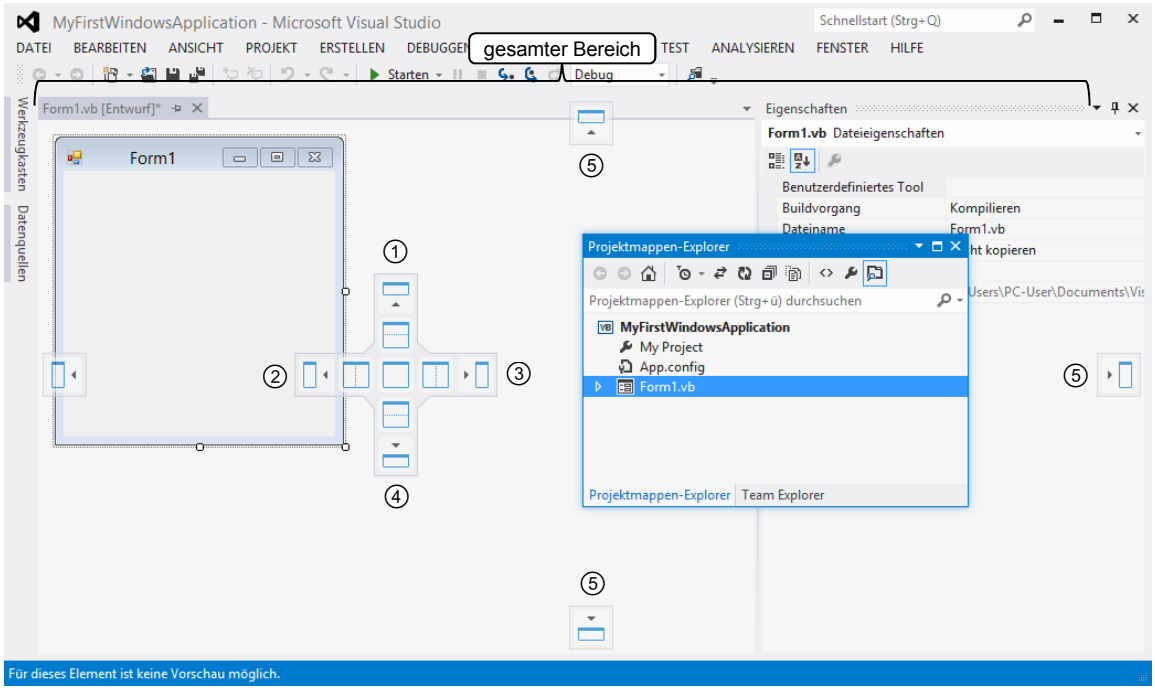
Position von andockbaren Fenstern ändern

▶ Klicken Sie in die Titelzeile des Fensters und ziehen Sie es in den gewünschten Bereich.
Es werden sogenannte Diamant-Führungssymbole eingeblendet.

▶ Ziehen Sie das Fenster auf das entsprechende Einfügesymbol, um es am oberen ①, linken ②, rechten ③ oder unteren ④ Rand des Arbeitsbereichs (links) oder des rechten Bereichs anzudocken.

oder Positionieren Sie das Fenster frei.

Um den jeweils **gesamten** linken, rechten, oberen oder unteren Bereich für ein Fenster zu verwenden, wählen Sie das entsprechende Einfügesymbol ⑤.



Fenster positionieren

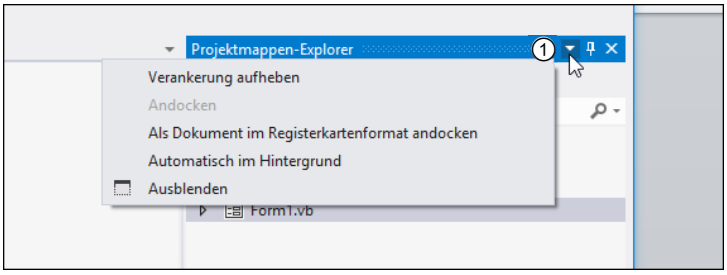
Sie möchten ...	
Fenster an der zuletzt verwendeten Andockposition platzieren	▶ Halten Sie die Taste Strg gedrückt und klicken Sie doppelt auf die Titelleiste eines frei positionierten Fensters.
Fenster im entsprechenden Bereich als Register einfügen	▶ Ziehen Sie das Fenster auf den gewünschten Registerbereich.

Fenster-Position wählen

▶ Klicken Sie in der Titelleiste eines Fensters auf die Pfeilschaltfläche (*Position des Fensters*) ①, um das Menü mit den grundlegenden Positionierungsarten zu öffnen.

oder Öffnen Sie das Kontextmenü der Titelleiste des Fensters.

▶ Wählen Sie den gewünschten Eintrag.



Position des Fensters wählen

Zu einem Projekt gehörende Fenster, sogenannte **Dokumentfenster**, wie z. B. der Programmcode-Editor und der Windows Forms-Designer, lassen sich nur frei anordnen und besitzen dieses Menü nicht.

Fenster unverankert positionieren

▶ Aktivieren Sie über die Pfeilschaltfläche in der Titelleiste den Eintrag *Verankerung aufheben*.

▶ Ziehen Sie das Fenster an die gewünschte Position.

Sie können ein Fenster auch außerhalb des Hauptfensters (Visual-Studio-Fensters) platzieren.

Unverankerte Fenster wieder andocken

- Wählen Sie über das Kontextmenü der Titelleiste des Fensters oder über die Pfeilschaltfläche den Eintrag *Andocken*.

Fenster als Dokumente im Registerkartenformat

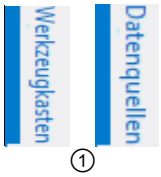
Neben den Dokumentfenstern können Sie auch Toolfenster, wie z. B. das Eigenschaftenfenster, im Arbeitsbereich als Dokument im Registerkartenformat anzeigen.

- Wählen Sie über die Titelleiste die Positionierungsart *Als Dokument im Registerkartenformat andocken*.

oder Ziehen Sie das Fenster auf den entsprechenden Registerbereich.

Fenster automatisch ausblenden

Fenster lassen sich so anordnen, dass sie nur als Register ① erscheinen. Das eigentliche Fenster ist ausgeblendet. Standardmäßig werden die Fenster *Werkzeugkasten* und *Datenquellen* auf diese Weise dargestellt.



- Klicken Sie auf das Symbol in der Titelleiste eines Fensters, um das Fenster als Register anzuzeigen.

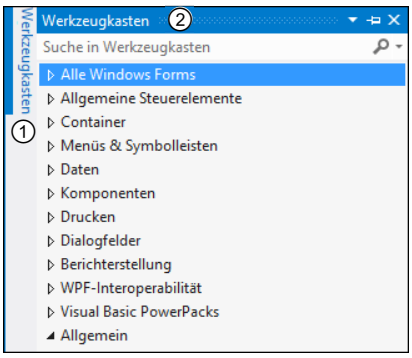
Über den Menüpunkt *FENSTER - Alle automatisch ausblenden* können Sie diese Einstellung für alle Toolfenster vornehmen.

Fenster vorübergehend einblenden

Fenster, die automatisch ausgeblendet werden, können Sie jederzeit einblenden.

- Klicken Sie mit der Maus auf das Register ①.

Das Fenster ② wird eingeblendet und kann verwendet werden. Sobald Sie in einen anderen Bereich der Entwicklungsumgebung klicken, wird das Fenster wieder ausgeblendet.



Fenster einblenden

Fenster dauerhaft einblenden

- Klicken Sie auf das Symbol in der Titelleiste des entsprechenden Fensters, um es dauerhaft einzublenden.

Fenster schließen bzw. öffnen

- Über das Schließfeld schließen Sie ein Fenster.
- Um ein Fenster wieder anzuzeigen, wählen Sie im Menü *ANSICHT* den entsprechenden Menüpunkt.

Alle Fenstereinstellungen wiederherstellen

- Rufen Sie den Menüpunkt *FENSTER - Fensterlayout zurücksetzen* auf, um die Fenstereinstellungen des ausgewählten Profils wiederherzustellen.

Zwei Grundformen von Schleifen-Strukturen sind in Visual Basic enthalten:

- **Bedingte Wiederholung:**

  Die Wiederholung des in der Schleife eingeschlossenen Programmteils ist an eine Bedingung geknüpft. Hierbei wird wiederum unterschieden, ob die Bedingung vor oder nach der Ausführung des eingeschlossenen Programmteils erfolgt:

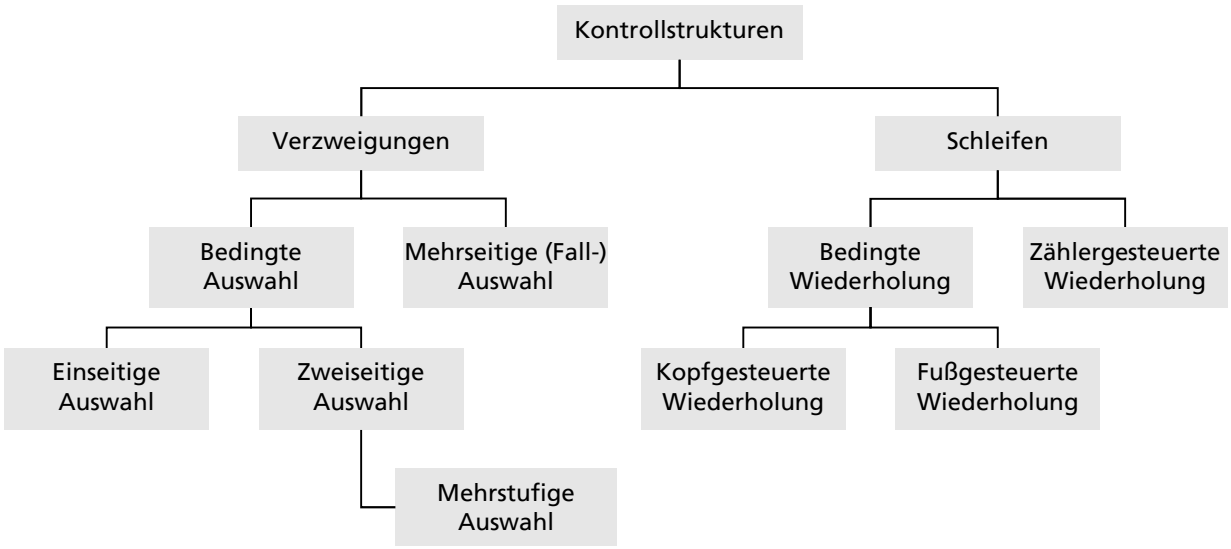
  Bei der **kopfgesteuerten Wiederholung** erfolgt die Überprüfung der Bedingung **vor** der Ausführung des eingeschlossenen Programmteils. Solange eine Bedingung erfüllt ist bzw. bis eine Bedingung erfüllt ist, wird der in der Struktur eingeschlossene Programmteil ausgeführt. Anschließend wird das Programm hinter der Schleifenstruktur fortgesetzt.

  Bei der **fußgesteuerten Wiederholung** wird zunächst der eingeschlossene Programmteil ausgeführt. **Anschließend** wird die Bedingung geprüft. Ist die Bedingung erfüllt bzw. noch nicht erfüllt, wird der Programmteil erneut ausgeführt. Anderenfalls wird das Programm hinter der Schleifenstruktur fortgesetzt.

- **Zählergesteuerte Wiederholung:**

  Ein Zähler bestimmt die Anzahl der Wiederholungen. Die Überprüfung, ob der Zähler den gewünschten Endwert erreicht hat, erfolgt **vor** der Ausführung des eingeschlossenen Programmteils.

Übersicht über die Kontrollstrukturen in Visual Basic

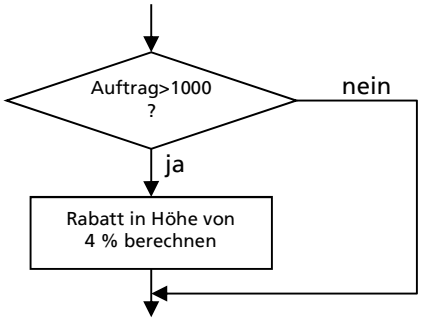


8.2 Einseitige Auswahl

Bei vielen Problemstellungen ist die Verarbeitung von Anweisungen von Bedingungen abhängig. Nur wenn die Bedingung erfüllt ist, wird die betreffende Anweisung (bzw. der Anweisungsblock) ausgeführt. Anderenfalls wird die Anweisung übersprungen. Für die Steuerung eines solchen Programmablaufs stellt Visual Basic die **einseitige** Auswahl (`If`-Anweisung) zur Verfügung.

Beispiel für die Verwendung einer einseitigen Auswahl

- Wenn (`If`) ein Kunde einen Auftrag über 1000,- EUR erteilt, dann (`Then`) erhält er 4 % Rabatt. Bei Aufträgen bis 1000,- EUR wird kein Rabatt gewährt, d. h., die Berechnung des Rabatts wird nicht ausgeführt.
- Die Überprüfung der Bedingung kann nur die beiden Ergebnisse "ja" oder "nein" ergeben, in Visual Basic entsprechend `True` (wahr) oder `False` (falsch).



Syntax der einseitigen Auswahl (`If`-Anweisung)

```
If Condition Then
  Statement ①
End If ②
```

- ✔ Die einseitige Auswahl beginnt mit dem Schlüsselwort `If`.
- ✔ Dahinter wird eine Bedingung (Condition) formuliert. Die Bedingung ist ein Ausdruck (Expression) und liefert als Ergebnis einen Wert vom Typ `Boolean` zurück.
- ✔ Nach der Bedingung folgt das Schlüsselwort `Then`.
- ✔ Anschließend folgt die Anweisung ①, die ausgeführt wird, wenn die Auswertung der Bedingung den Wert `True` ergibt. Liefert die Bedingung den Wert `False`, wird diese Anweisung übersprungen. Die Anweisung ① kann aus einer einzelnen Anweisung oder einem Anweisungsblock bestehen.
- ✔ Das Ende der einseitigen Auswahl wird durch `End If` ② gekennzeichnet.

Sehr kurze `If`-Anweisungen können in eine einzige Zeile geschrieben werden. In diesem Fall entfällt `End If`.

```
If Condition Then Statement
```

Beispiel: *Discount.sln*

Für einen Rechnungsbetrag (`InvoiceAmount`) soll abhängig von seiner Höhe ein Rabatt berechnet werden. Anhand des Wertes der Variablen `InvoiceAmount` wird entschieden, ob ein Rabatt gewährt wird. Dieser wird gegebenenfalls berechnet und vom Rechnungsbetrag subtrahiert. Der neue Rechnungsbetrag wird (abzüglich des Rabatts) ausgegeben. Währungsangaben werden hier nicht berücksichtigt.

```
   Module MdlDiscount
     Sub Main()
       Dim InvoiceAmount As Double = 0.0  ' Rechnungsbetrag
       Dim Txt As String
       Console.Write ("Bitte geben Sie den Rechnungsbetrag ein: ")
①      Txt = Console.ReadLine()
②      InvoiceAmount = CDbl(Txt)
③      If InvoiceAmount > 1000 Then
④        invoiceAmount -= InvoiceAmount * 0.04
         Console.WriteLine("Es wird ein Rabatt gewaehrt.")
       End If
⑤      Console.WriteLine("Gesamtbetrag: " & InvoiceAmount)
       Console.ReadKey()
     End Sub
   End Module
```

① Die eingegebene Zeichenkette wird eingelesen.

② Der String wird in eine `Double`-Zahl konvertiert und einer Variablen zugewiesen.

③ Der Wert dieser Variablen wird dann mit dem Wert 1000 verglichen.

④ Ist der Rechnungsbetrag `InvoiceAmount` größer als 1000, liefert die Bedingung den Wert `True`; der Rabatt wird berechnet und ein Text als Information ausgegeben.

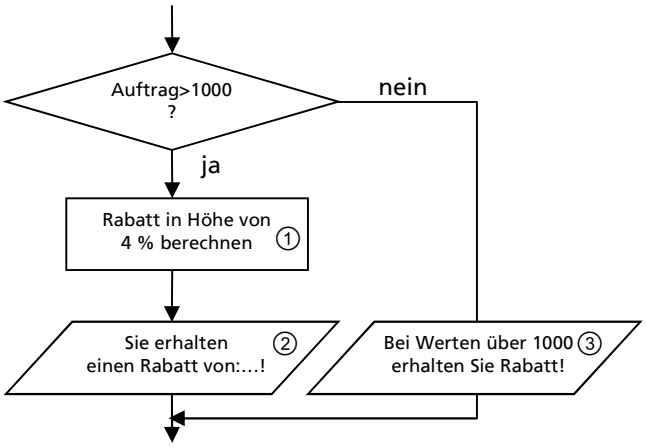
⑤ Der Rechnungsbetrag wird ausgegeben. Wenn kein Rabatt gewährt wird, wird der Anweisungsblock ④ übersprungen und der Rechnungsbetrag unverändert ausgegeben ⑤.

8.3 Zweiseitige Auswahl

Die **zweiseitige Auswahl** ist dadurch gekennzeichnet, dass einer von **zwei** Anweisungsblöcken in Abhängigkeit von einer Bedingung ausgeführt wird. Bestimmte Anweisungen werden durchgeführt, falls die Bedingung erfüllt ist. Falls die Bedingung nicht erfüllt ist, werden andere Anweisungen ausgeführt.

Beispiel für den Einsatz einer zweiseitigen Auswahl

- Wenn (`If`) ein Kunde einen Auftrag über 1000,- EUR erteilt, dann (`Then`) erhält er 4 % Rabatt ①. Eine Meldung ② informiert darüber.
- Sonst (`Else`), d. h. bei Aufträgen bis 1000,- EUR, wird kein Rabatt gewährt und eine entsprechende Meldung ③ ausgegeben.



Syntax der zweiseitigen Auswahl

- Nach dem Schlüsselwort `If` stehen die Bedingung ① und das Schlüsselwort `Then`.
- Anschließend folgt die Anweisung ②, die ausgeführt werden soll, wenn die Bedingung erfüllt ist.
- Nach `Else` schließt sich die Anweisung ③ an, die ausgeführt wird, wenn die Bedingung in der `If`-Anweisung **nicht** zutrifft (Ausdruck liefert `False`). Die Anweisung ② nach `Then` bleibt unberücksichtigt.
- `End IF` kennzeichnet das Ende der zweiseitigen Auswahl.

```
If Condition Then ①
  Statement1      ②
Else
  Statement2      ③
End If
```

Beispiel: *Discount2.sln*

Für einen Rechnungsbetrag (`InvoiceAmount`) soll abhängig von seiner Höhe ein Rabatt (`DiscountAmount`) berechnet werden. Wird ein Rabatt gewährt, wird dieser berechnet und der Rabattbetrag ausgegeben. Sofern kein Rabatt gewährt wird, erfolgt die Ausgabe einer entsprechenden Information.

```
...
①   Dim InvoiceAmount As Double = 0.0   ' Rechnungsbetrag
    Const DiscountRate As Double = 0.04 ' Rabatt 4%, als Konstante
    Dim DiscountAmount As Double = 0.0  ' Rabattbetrag
    Dim Txt As String
    Console.Write ("Bitte geben Sie den Rechnungsbetrag ein: ")
    Txt = Console.ReadLine()
    InvoiceAmount = CDbl(Txt)
②   If InvoiceAmount > 1000 Then
③     DiscountAmount -= InvoiceAmount * DiscountRate
      Console.WriteLine("Sie erhalten einen Rabatt von: " & DiscountAmount)
    Else
④     Console.WriteLine("Bei Werten ueber 1000 erhalten Sie Rabatt!")
    End If
...
```

① Die Variablen und eine Konstante werden definiert und initialisiert.

② Es wird ein Rabatt bei einem Rechnungsbetrag über 1000 gewährt.

③ Ist die Bedingung erfüllt, wird der Rabatt berechnet und ausgegeben.

④ Ist der eingegebene Wert kleiner oder gleich 1000, wird kein Rabatt gewährt. Sie erhalten jedoch eine Meldung, wann Sie Rabatt erhalten.

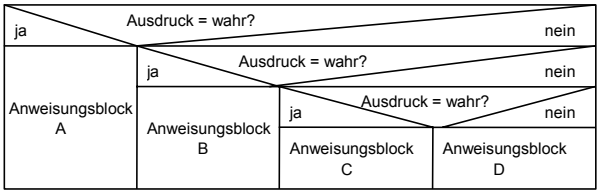
Mit dem `If`-Operator auswerten

In Visual Basic können Sie schnell zwei Werte abhängig von einer Bedingung mit dem `If`-Operator auswerten:

- ✔ `If (Condition, ValueTrue, ValueFalse)` - liefert die Bedingung `True`, wird `ValueTrue` berechnet und zurückgeliefert, ansonsten `ValueFalse`.
- ✔ `If (ValueA, ValueB)` - ist `ValueA` gleich `Nothing`, wird `Nothing` zurückgeliefert, ansonsten wird `ValueB` berechnet und zurückgeliefert.

8.4 Mehrstufige Auswahl

Mehrstufige Auswahl bedeutet, dass einer von mehreren Anweisungsblöcken in Abhängigkeit von Bedingungen ausgeführt wird. **Wenn** (`If`) die erste Bedingung erfüllt ist, **dann** (`Then`) wird Anweisungsblock A ausgeführt, **sonst** wird die nächste Bedingung ausgewertet.



Gegenüber der einseitigen und der zweiseitigen Auswahl ist die mehrstufige Auswahl hier nicht als Programmablaufplan, sondern als **Struktogramm** dargestellt. Ein Struktogramm (auch Nassi-Shneidermann-Diagramm genannt) ist eine sehr kompakte Darstellungsform für Programmabläufe.

Syntax der mehrstufigen Auswahl

- ✔ Der `If`- und der `Else`-Zweig unterscheiden sich nicht von der zweistufigen Auswahl.
- ✔ Zusätzlich können Sie mithilfe von `ElseIf`-Blöcken beliebig viele Bedingungen formulieren.

```
If Condition1 Then
  Statement
ElseIf Condition2 Then
  Statement
ElseIf Condition3 Then
  Statement
Else
  Statement
End If
```

Beispiel: *Discount3.sln*

Im Unterschied zu den beiden letzten Projekten soll bereits bei einem niedrigeren Rechnungsbetrag ein Rabatt in Höhe von 2 % gewährt werden.

```
    ...
        Dim InvoiceAmount As Double = 0.0   ' Rechnungsbetrag
        Dim Txt As String
        Console.Write("Bitte geben Sie den Rechnungsbetrag ein: ")
        ...
①       If InvoiceAmount > 1000 Then
②         InvoiceAmount -= InvoiceAmount * 0.04
          Console.WriteLine("Sie erhalten 4 % Rabatt")
③       Elseif InvoiceAmount > 500 Then
④         InvoiceAmount -= InvoiceAmount * 0.02
          Console.WriteLine("Sie erhalten 2 % Rabatt")
        Else
⑤         Console.WriteLine("Bei Werten ueber 500 erhalten Sie Rabatt!")
        End If
⑥       Console.WriteLine("Gesamtbetrag: " & InvoiceAmount)
    ...
```

① Zunächst wird geprüft, ob der Rechnungsbetrag 1000 übersteigt. Wenn dies der Fall ist, werden 4 % Rabatt gewährt ②.