19.8.2 Klasse Expression

Diese Klasse repräsentiert einen *dynamischen* Gambas-Ausdruck. Dynamisch bezeichnet in diesem Zusammenhang die Möglichkeit, eigene *Symbole* im Ausdruck anzugeben und diesen nicht definierten Symbolen erst zur Laufzeit des Gambas-Programms Werte zuzuweisen.

Die Anzahl der Eigenschaften der Klasse *Expression* ist recht überschaubar. Sie besitzt neben den drei u.a. Eigenschaften nur eine Methode *Compile*, die Sie jedoch meist nur indirekt einsetzen werden.

Eigenschaft Expression	Type	Beschreibung
.Environment	Collection	In der Collection *Expression.Environment* werden die Werte für die nicht definierten Symbole im Ausdruck gespeichert oder diese Werte werden aus der Collection ausgelesen.
.Text	String	In *Expression.Text* wird der Ausdruck als Text in einer Zeichenkette gespeichert. Dieser Ausdruck kann fast alle Operatoren und Unterprogramme von Gambas enthalten.
.Value	Variant	*Expression.Value* gibt den berechneten Wert des Ausdrucks zurück.

Tabelle 19.8.2.1: Eigenschaften der Klasse Expression

Im folgenden Projekt wird gezeigt, wie man die Klasse *Expression* zur Auswertung eines Ausdrucks einsetzt. Das folgende Konzept hat sich bewährt:

- Zuerst werden zwei Variablen angelegt, die zur Laufzeit ein neues Expression-Objekt generieren und eine Collection, die später der Expression-Eigenschaft *Expression.Environment* zugewiesen werden kann.
- Enthält der Ausdruck nicht definierte Symbole, werden den nicht definierten Symbolen Werte zugewiesen und diese Werte in einer Variablen vom Daten-Typ 'Collection' gespeichert.
- Dann wird der Ausdruck angegeben, von dem der Wert berechnet werden soll.
- Danach wird der Expression-Eigenschaft *Expression.Environment* die gefüllte Collection zugewiesen.
- Abschließend kann der berechnete und in der Eigenschaft *Expression.Value* gespeicherte Wert des Ausdrucks weiter verwendet werden.

Der Quelltext setzt das o.a. Konzept konsequent um:

```
' Gambas class file

Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
  vlbParameterA.Type = vlbParameterA.Number
  vlbParameterA.SetFocus
  vlbParameterB.Type = vlbParameterB.Number
  vlbArgumentX.Type = vlbArgumentX.Number
  vlbFunktionswert.Type = vlbFunktionswert.Number
  vlbFunktionswert.ReadOnly = True
' Initialisierung:
  txbExpression.Text = "a*sin(x)+cos(b*x)" ' Default-Expression
  vlbParameterA.Value = +0.125
  vlbParameterB.Value = -2.5
  vlbArgumentX.Value = 0.525
End ' Form_Open()

Private Function Compute(sExpression As String, fParameterA As Float, fParameterB As Float, \
                         fArgumentX As Float) As Float
  Dim fFunktionswertY As Float
  Dim cEnvironment As New Collection
  Dim myExpression As New Expression

' cEnvironment.Add(fParameterA, "a") ' ALTERNATIVE
' cEnvironment.Add(fParameterB, "b")
' cEnvironment.Add(fArgumentX, "x")
  cEnvironment["a"] = fParameterA ' Dem Symbol a wird der Wert vlbParameterA.Value zugewiesen
  cEnvironment["b"] = fParameterB ' Dem Symbol b wird der Wert vlbParameterB.Value zugewiesen
  cEnvironment["x"] = fArgumentX  ' Dem Symbol x wird der Wert vlbArgumentX.Value zugewiesen
```

```
  myExpression.Environment = cEnvironment
  myExpression.Text = txbExpression.Text
  fFunktionswertY = myExpression.Value

  Return fFunktionswertY

End ' Function(..)

Public Sub btnComputeY_Click()
  vlbFunktionswert.Value = Compute(txbExpression.Text, vlbParameterA.Value,  \
                                   vlbParameterB.Value, vlbArgumentX.Value)
End ' btnComputeY_Click()

Public Sub btnClose_Click()
  FMain.Close
End ' btnClose_Click()
```

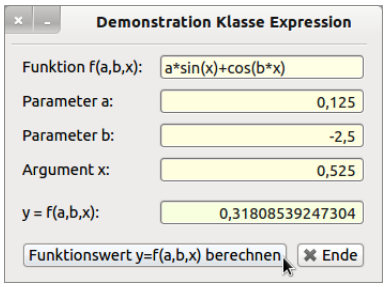

Abbildung 19.8.2.2: Berechnung des Funktionswertes (Funktionsschar)

Ist Ihnen aufgefallen, das es in diesem Projekt recht dynamisch zugeht? Sie können nämlich nicht nur den Ausdruck ändern, sondern auch die Werte der beiden Parameter der Funktionsschar f(a,b,x) und das Argument x. Sie implementieren eine einfache Fehlerbehandlung, wenn Sie die folgenden Zeilen im Quelltext:

```
...
fFunktionswertY = myExpression.Value
  Return fFunktionswertY
...
```

gegen diese Zeilen austauschen:

```
TRY fFunktionswertY = myExpression.Value
If ERROR Then
    Message.Error(Error.Where & gb.NewLine & "Fehlertext: " & Error.Text)
Else
     Return fFunktionswertY
Endif ' ERROR ?
```

Damit erhalten Sie zum Beispiel beim Ausdruck '*a*sin(x)+cos(b*x)+c*' die folgende Fehlermeldung:

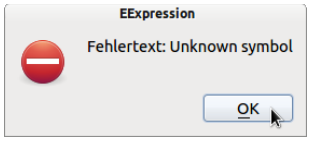

Abbildung 19.8.2.3: Fehlermeldung 1

und für den Ausdruck 'a/(sin(x-b))' mit dem Parameterwert b=2 und dem Argument x=2 diese Meldung:

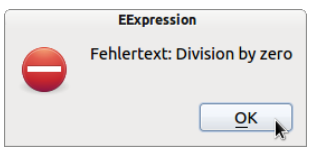

Abbildung 19.8.2.4: Fehlermeldung 2