7.4.9.4 Projekt – Sortierung eines abgeleiteten Arrays

Das Sortieren eines abgeleiteten Arrays wurde bereits im → Kapitel 7.4.9.3 beschrieben. Der gravierende Nachteil der im Kapitel vorgestellten Sortierung besteht darin, dass Sie das Feld, nach dem sortiert werden soll, fest im Programm vorgeben müssen. In diesem Kapitel wird eine Lösung für das Sortieren eines abgeleiteten Arrays vorgestellt, in dem das Feld *dynamisch* festgelegt werden kann, nach dem das Objekt-Array sortiert wird. Alle Ausführungen im → Kapitel 7.4.9.3 gelten auch für dieses Projekt, besonders die Details zur _compare()-Methode. Die Klasse CDS wird im Projekt um einige Variablen und eine Eigenschaft ergänzt.

7.4.9.4.1 Klassen-Definition

Das zu sortierende Array *aCDSArray* ist, wie im → Kapitel 7.4.9.3, ein eindimensionales, abgeleitetes Array, dessen Elemente Objekte einer selbst geschriebenen Klasse *CDS* sind. Der Array-Typ ist deshalb CDS[] respektive Object[].

- Alle Elemente des Arrays sind vom Typ CDS und bestehen im Projekt jeweils aus den Werten von sieben verschiedenen Variablen mit unterschiedlichem Daten-Typ, die in der Klasse CDS deklariert sind.
- Die _compare()-Methode wird gegenüber der Variante aus → Kapitel 7.4.9.3 entsprechend angepasst.
- Der Klasse CDS wird die Eigenschaft SortField vom Datentyp Integer hinzugefügt, auf die man lesend und schreibend zugreifen kann, um an geeigneter Stelle das Feld angeben zu können, nach dem sortiert werden soll.

Datei CDS.class:

```
[1]  ' Gambas class file
[2]
[3]  Public JGS As Integer
[4]  Public GebDatum As Date
[5]  Public Vorname As String
[6]  Public Nachname As String
[7]  Public Latein As Boolean
[8]  Public Kurs1 As String
[9]  Public Kurs2 As String
[10]
[11] Public Function _compare(DS As CDS) As Integer
[12]
[13]   Select Case $SortField
[14]   Case 1
[15]     Return Sgn(JGS - DS.JGS) ' Zahl
[16]   Case 2
[17]      Return - Sgn(DateDiff(GebDatum, DS.GebDatum, gb.Second)) ' Datum
[18]   Case 3
[19]     Return Comp(Vorname, DS.Vorname) ' Zeichenkette (String)
[20]   Case 4
[21]     Return Comp(Nachname, DS.Nachname)
[22]   Case 5
[23]     Return IIf(Latein, IIf(DS.Latein, 0, 1), IIf(DS.Latein, -1, 0)) ' Wahrheitswert
[24]   Case 6
[25]     Return Comp(Kurs1, DS.Kurs1)
[26]   Case 7
[27]     Return Comp(Kurs2, DS.Kurs2)
[28]    Default
[29]       Error.Raise(("Keine Sortier-Methode gefunden!"))
[30]   End Select
[31]
[32] End
[33]
[34] Private $SortField As Integer
[35] Property SortField As Integer
[36]
[37] Private Function SortField_Read() As Integer
[38]   Return $SortField
[39] End
[40]
[41] Private Sub SortField_Write(NewValue As Integer)
[42]   If NewValue < 1 Or NewValue > 7 Then
[43]      Error.Raise(("Eingabefehler! Eingabe nur aus [1..7]" wählen.))
[44]   Else
[45]     $SortField = NewValue
[46]   Endif
[47] End
```

Die Struktur der Elemente im abgeleiteten Array *aCDSArray* wird im Projekt besonders deutlich, wenn man ein neues Element des Arrays vom Typ CDS anlegt, mit Daten füllt und in das Array einfügt:

```
hCDS = New CDS
  hCDS.JGS = 11
  hCDS.GebDatum = Date(Year(Now()) - 17, 12, 13)
  hCDS.Vorname = "Gerd"
  hCDS.Nachname = "Geier"
  hCDS.Latein = True
  hCDS.Kurs1 = "Sozialkunde"
  hCDS.Kurs2 = "Chemie"
aCDSArray.Add(hCDS)
```

Für ein Array zum Beispiel mit 6 Elementen vom Typ CDS ergibt sich die folgende (formatierte) Anzeige des Arrays in der Konsole der IDE, die aus 6 Zeilen und 7 (fiktiven) Spalten oder Feldern besteht:

```
11      22.08.1996      Doris   Fuchs   True    Biologie        Deutsch
12      18.08.1997      Anna    Katze   False   Mathematik      Englisch
11      17.06.1996      Maria   Zebra   False   Informatik      Deutsch
12      02.05.1997      Anna    Uhu     False   Astronomie      Kunst
12      27.04.1996      Fred    Dachs   False   Astronomie      Deutsch
11      13.12.1997      Gerd    Geier   True    Sozialkunde     Chemie
```

Für das Sortieren des abgeleiteten Arrays werden die folgenden Forderungen erhoben:

- Das Array soll nach einer frei wählbaren Spalte (Feld) – zum Beispiel nach den Nachnamen – sortiert werden können.
- Wie bei einer Sortierung üblich, soll *optional* über eine der beiden Konstanten gb.Ascent (0) (Default) für eine aufsteigende oder gb.Descent (16) für eine absteigende Sortierung eine Sortierreihenfolge angegeben werden können.

7.4.9.4.2 Ansatz zur Umsetzung der Forderungen

Statt der möglichen Array-Klasse CDS[] – die automatisch vom Interpreter erzeugt würde und so über alle Eigenschaften und Methoden eines Arrays verfügt – wird im Projekt die Klasse Object[] eingesetzt. Das ist möglich, weil alle Elemente des zu sortierenden Arrays *Objekte* der Klasse CDS sind. Um den Nachteil zu beseitigen, das Feld – nach dem sortiert werden soll – nur statisch angeben zu können, wird die Klasse Object[] um eine Methode erweitert, um *mehrdimensional* zu sortieren. Benoît Minisini weist darauf hin, dass man die Klasse Object[] nicht direkt überschreiben kann, weil '[' und ']' in Projekt-Dateinamen nicht erlaubt sind. Er schlägt deshalb vor, dass in einer neuen Klasse über *Inherits Object[]* alle Methoden und Eigenschaften der Klasse Object[] geerbt werden und Erweiterungen dort implementiert werden. An allen Stellen im Projekt ersetzt die neue Klasse die Klasse Object[].

Der Quelltext der neuen Klasse CDSArray in der Datei CDSArray.class ist leicht zu überblicken:

```
[1]  ' Gambas class file
[2]
[3]  Inherits Object[]
[4]
[5]  Public Sub SortNew(Field As Integer, Optional Mode As Integer)
[6]    Dim iIndex As Integer
[7]
[8]    If Super.Count = 0 Then Return
[9]
[10]   For iIndex = 0 To Super.Count - 1
[11]     Super[iIndex].SortField = Field
[12]   Next
[13]
[14]   Super.Sort(Mode)
[15]
[16] End
```

Kommentare:

- Um keinen Quelltext zu duplizieren, der in Gambas bereits vorhanden ist, baut die neue Klasse CDSArray auf der Klasse Object[] auf – die ein Array von Objekten repräsentiert.
- Die Klasse 'CDSArray' erbt in der Zeile 3 alle Methoden und Eigenschaften der Klasse Object[ ].
- Der Klasse CDSArray wird in den Zeilen 5 bis 16 die Methode *SortNew(Field As Integer, Optional Mode As Integer)* hinzugefügt.

- Die Methode SortNew(..) ist einfach. Sie prüft zuerst, ob Elemente im zu sortierenden Array vorhanden sind, die sortiert werden können. Für diesen Fall wird die SortField-Eigenschaft in allen Elementen gesetzt, denn diese Eigenschaft ist Teil der Erweiterung der Klasse CDS, die später die Sortierung steuert. Der Einsatz des speziellen Objekts *Super* ermöglicht den Zugriff auf die Daten von Object[], von dem die CDSArray-Klasse abgeleitet ist. So wird auf jedes enthaltene CDS-Objekt zugegriffen.
- Ist die SortField-Eigenschaft aller Elemente vom Typ CDS gesetzt, wird über die gewöhnliche Object[].Sort([Mode As Integer])-Methode sortiert, nachdem über den *optionalen* Parameter 'Mode' die Sortierreihenfolge festgelegt wurde.
- Die Methode Object[].Sort([Mode As Integer]) führt intern den QuickSort-Algorithmus aus. Das ist ein vergleichsbasierter Sortieralgorithmus. Um zwei Objekte aus dem Array zu vergleichen, wird die _compare()-Methode in der Klasse CDS aufgerufen. Dazu sagt die Dokumentation: Objects are compared by calling the special public method _compare oder Objekte werden durch den Aufruf der speziellen öffentlichen Methode _compare() verglichen.

Was wurde bisher erreicht?

Es steht Ihnen einerseits die Klasse CDS als Daten-Container mit _compare()-Methode und der Eigenschaft *SortField* zur Verfügung und andererseits die neue Klasse *CDSArray*, welche die Klasse Object[] überschreibt und in der SortNew()-Methode eine Methode bereitstellt, in der Sie über die Eigenschaft *SortField* das Feld angeben können, nach dem sortiert werden soll und optional die Sortierreihenfolge.

Im Hauptprogramm werden die beiden Klassen CDS und CDSArray eingesetzt. Es wird ein Array *aCDSArray* der Klasse *CDSArray* generiert, auf unterschiedliche Art mit Daten gefüllt und nach einem frei wählbaren Feld sortiert sowie der Inhalt des originalen und des sortierten Arrays ausgegeben:

```
' Gambas class file

Public aCDSArray As CDSArray
Public iAsDescent As Integer = 0

Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
  rbtnAscent.Value = True
  rbtnAscent.SetFocus
End ' Form_Open()

Public Sub btnClassArray_Click()
  Dim iCount As Integer
  Dim hCDS As CDS
  Dim aNames As String[] = ["Adler", "Bär", "Dachs", "Fuchs", "Meise", "Uhu", "Zebra"]
  Dim aSurNames As String[] = ["Anna", "Bruno", "Doris", "Fred", "Maria", "Klaus", "Udo"]
  Dim aCourses1 As String[] = ["Mathematik", "Geschichte", "Astronomie", "Biologie", "Informatik"]
  Dim aCourses2 As String[] = ["Deutsch", "Physik", "Englisch", "Kunst", "Musik"]

  aCDSArray = New CDSArray

  Randomize

  For iCount = 0 To 4
    hCDS = New CDS
      hCDS.JGS = CInt(Rnd(11, 13)) ' 11..12
    ' Zufallsdatum: Jahr: passend zur JGS, Monat: 1..12, Tag: 1..28 (das passt immer...)
      hCDS.GebDatum = Date(Int(Rnd(Year(Now()) - 18, Year(Now()) - 16)), Int(Rnd(1, 13)), Int(Rnd(1, 29)))
      hCDS.Vorname = aSurNames[Int(Rnd(0, aSurNames.Count))]
      hCDS.Nachname = aNames[Int(Rnd(0, aNames.Count))]
      hCDS.Latein = CBool(Round(Rnd(0, 1)))
      hCDS.Kurs1 = aCourses1[Int(Rnd(0, aCourses1.Count))]
      hCDS.Kurs2 = aCourses2[Int(Rnd(0, aCourses2.Count))]

      aCDSArray.Add(hCDS)

  Next ' iCount

  hCDS = New CDS        ' Ein neues, aber leeres Array-Element vom Objekt-Typ CDS generieren ...
  aCDSArray.Add(hCDS)   ' und in das Array einfügen
' Letztes Array-Element mit den folgenden Daten überschreiben:
  aCDSArray[aCDSArray.Max].JGS = 11
  aCDSArray[aCDSArray.Max].GebDatum = Date(Year(Now()) - 17, 12, 13)
  aCDSArray[aCDSArray.Max].Vorname = "Gerd"
  aCDSArray[aCDSArray.Max].Nachname = "Geier"
  aCDSArray[aCDSArray.Max].Latein = True
```

```
  aCDSArray[aCDSArray.Max].Kurs1 = "Sozialkunde"
  aCDSArray[aCDSArray.Max].Kurs2 = "Chemie"

' Anzeige aller Datensätze
  ShowCDSArrayElements(aCDSArray)
' Sortierung Array:
' 1. Parameter: Feldnummer des Feldes, nach dem sortiert werden soll (SpinBox)
' 2. Parameter: Festlegung der Sortier-Reihenfolge (optional) (RadioButton)

  aCDSArray.SortNew(spinBox.Value, iAsDescent) ' → Feld 4 (Nachname), absteigend (Z..a)

  ShowCDSArrayElements(aCDSArray)

End ' btnClassArray_Click()

Private Sub ShowCDSArrayElements(aArray As CDSArray)
  Dim hCDS As CDS
  Dim i As Integer

' Anzeige aller Datensätze (Konsole der IDE)
  For Each hCDS In aArray
    Print hCDS.JGS,
    Print Format(hCDS.GebDatum, "dd.mm.yyyy"),
    Print hCDS.Vorname,
    Print hCDS.Nachname,
    Print hCDS.Latein,
    Print hCDS.Kurs1,
    Print hCDS.Kurs2
  Next ' hCDS

' Variante 2
' For i = 0 To aArray.Max
'   Print aArray[i].JGS,
'   Print Format(aArray[i].GebDatum, "dd.mm.yyyy"),
'   Print aArray[i].Vorname,
'   Print aArray[i].Nachname,
'   Print aArray[i].Latein,
'   Print aArray[i].Kurs1,
'   Print aArray[i].Kurs2
' Next
' Print

End ' ShowCDSArrayElements(aArray As ObjectSort)

Public Sub rbtnAscent_Click()
  iAsDescent = gb.Ascent
End ' rbtnAscent_Click()

Public Sub rbtnDescent_Click()
  iAsDescent = gb.Descent
End ' rbtnDescent_Click()
```

Ausgabe des originalen und des sortierten Arrays – sortiert nach den Nachnamen (absteigend) – in der Konsole der IDE:

```
11      22.08.1996      Doris   Fuchs   True    Biologie        Deutsch
12      18.08.1997      Anna    Katze   False   Mathematik      Englisch
11      17.06.1996      Maria   Zebra   False   Informatik      Deutsch
12      02.05.1997      Anna    Uhu     False   Astronomie      Kunst
12      27.04.1996      Fred    Dachs   False   Astronomie      Deutsch
11      13.12.1997      Gerd    Geier   True    Sozialkunde     Chemie
```

```
11      17.06.1996      Maria   Zebra   False   Informatik      Deutsch
12      02.05.1997      Anna    Uhu     False   Astronomie      Kunst
12      18.08.1997      Anna    Katze   False   Mathematik      Englisch
11      13.12.1997      Gerd    Geier   True    Sozialkunde     Chemie
11      22.08.1996      Doris   Fuchs   True    Biologie        Deutsch
12      27.04.1996      Fred    Dachs   False   Astronomie      Deutsch
```