
Sample Code Reference Sheet

Internal Documentation: Header

```
' [Title]
' [Description]
' Created by [name]
' Last updated on [date]
' Version [number]
' [Subject]
' [Project]
```

Combo-Box

```
cbb[name].Items.Add([item]) ' Adds item to combo-box
cbb[name].SelectedIndex = -1 ' Unselects all items in combo-box
```

List-Box

```
lst[name].Items.Add([item]) ' Adds item to list-box
lst[name].Items.Clear() ' Clears items in the list-box
lst[name].Items.Remove([item]) ' Removes an item
lst[name].Items.RemoveAt([index]) ' Removes the item on the index
lst[name].SelectedIndex = -1 ' Makes sure no items are highlighted
```

Check-Box

```
ckb[name].Checked = True ' Checks the check-box
```

Radio Button

```
rbtn[name].Checked = True ' Checks the radio button
```

XML

Imports

```
Imports System.IO
Imports System.Xml
```

Load

```
Dim xml_Doc As XmlDocument
Dim xml_NodeList As XmlNodeList
Dim xml_Node As XmlNode
' Declares XML variables

xml_Doc = New XmlDocument ' Creates the XML document

xml_Doc.Load("[file location]") ' Loads file to xml_Doc
xml_NodeList = xml_Doc.SelectNodes("/[root]/[parent]") ' Enters the nodes

For Each xml_Node In xml_NodeList
    [variable] = xml_Node.ChildNodes.Item(0).InnerText ' Child nodes
Next
```

Load (Alternate)

```
Dim xmlReader As XmlTextReader
xmlReader = New XmlTextReader("C:\Users\...\U301\ComputerParts\PartsList TNG.xml")
```

```

Dim strTempPartNo As String
Dim strTempDesc As String
Dim sglTempPrice As Single
Dim sglTempDscPrice As Single
Dim blTempIsDisc As Boolean
Dim intCurrentPart As Int16
xmlReader.Read() ' read the xml declaration <? xml versi
xmlReader.Read() ' read the root
xmlReader.Read() ' read the Parent
While (xmlReader.Name() <> "Parts_List")
xmlReader.Read() ' <Part>
strTempPartNo = xmlReader.ReadElementString("PartNo")
strTempDesc = xmlReader.ReadElementString("Description")
sglTempPrice = xmlReader.ReadElementString("Price")
sglTempDscPrice = xmlReader.ReadElementString("DiscPrice")
blTempIsDisc = xmlReader.ReadElementString("Discontinued")
xmlReader.Read() ' </Part>
ReDim Preserve arrParts(intCurrentPart)
arrParts(intCurrentPart) = New PartDetails(strTempPartNo, strTempDesc, sglTempPrice, sglTempD:
intCurrentPart += 1
End While
xmlReader.Read() ' Close the root (PartsList)
xmlReader.Close() 'close the reader

```

Save

```

Dim xml_Settings As XmlWriterSettings = New XmlWriterSettings()
xml_Settings.Indent = True
Dim xml_writer As XmlWriter = XmlWriter.Create("[file location]",
xml_settings)
' Initiates XML with required settings

xml_writer.WriteStartDocument() ' Opens the document
xml_writer.WriteStartElement("[name]") ' Creates root node

For i = 0 To arr[name].Length - 1
    xml_writer.WriteStartElement("[name]") ' Creates parent node

    With arr[name](i)
        xml_writer.WriteElementString("[name]", .[variable]) ' Creates child
        node
    End With

    xml_writer.WriteEndElement() ' Closes parent node

Next

xml_writer.WriteEndElement() ' Closes root node
xml_writer.WriteEndDocument() ' Closes the document
xml_writer.Close() ' Closes the writer

```

Error Provider

```

epd[name].SetError([object], [message]) ' Sets the error provider message

```

Record Structure

```

Public Structure rec[name]
    Public [name] As [data type]

```

```
End Structure
' Creates a public structure
```

Multiple Forms

```
Dim frm[name] As New [form name] ' Declares the form as a variable

frm[name].Show() ' Shows the form
frm[name].ShowDialog() ' Shows the form exclusively

Me.Close() ' Closes the form
```

Message-Box

Normal

```
MessageBox.Show([message], [title], MessageBoxButtons.[buttons],
MessageBoxIcon.[icon]) ' Shows a message-box
```

Multiple Answers

```
Dim msgboxResult As DialogResult ' Variable used to store the answer

[msgboxResult] = MessageBox.Show([message], [title],
MessageBoxButtons.[buttons], MessageBoxIcon.[icon])

If [msgboxResult] = DialogResult.Yes Then
End If
```

Random Numbers

```
Randomize() ' Prepares to randomise
[variable] = Rnd() ' Chooses a random number greater than or equal to 0
and less than 1
```

ASCII Table & Values

On [form]_KeyPress

```
lbl[name].Text = e.KeyChar ' Shows the key pressed on the label
lbl[name].Text = Asc(e.KeyChar) ' Shows the ASCII value of the key pressed

' The value "e" within KeyPress is a numeration which returns the key that
has been pressed
```

Class Structure

```
Public Class cls[name]
    Public [name] As [data type]
End Class
' Creates a public class
```

Sort (Selection & Quick)

Selection Sort

```
' The selection sort algorithm sorts an array by repeatedly finding the
minimum element(considering ascending order) from unsorted part and
putting it at the beginning. The algorithm maintains two subarrays in a
given array.
' 1) The subarray which is already sorted.
' 2) Remaining subarray which is unsorted.
Dim minDex, intLoop1, intLoop2 As Int32

For intLoop2 = 0 To arr[name].Count - 2
    minDex = intLoop1
```

```

For intLoop1 = intLoop2 + 1 To arr[name].Count - 2
    If arr[name](intLoop1) < arr[name](minDex) Then minDex =
intLoop1
Next

Swap(minDex, intLoop2)

```

Next

Quick Sort

```

QuickSort(arr[name], 0, arr[name].Count - 1)

' Quick sort algorithm (divide and conquer in-place) recursively
' 1. Choose any element in the array array[p..r]. Call this element the
pivot q
' 2. Rearrange the elements in array[p..r] so that all elements in
array[p..r] that are less than Or equal (<=) To the pivot q are To its
left And all elements that are greater (<) than the pivot q are To its
right
' 3. call quicksort on subarray array[p..q-1] and call quicksort on
subarray array[q+1..r] Note item q Is Not part of either of these
subarrays
' 4. these subarrays are then joined together recursively as each call to
quicksort ends
Dim Low, High As Int16
Dim Pivot As String

Low = intFirst
High = intLast
Pivot = intSubList((intFirst + intLast) \ 2)

Do
    While intSubList(Low) < Pivot
        Low = Low + 1
    End While

    While intSubList(High) > Pivot
        High = High - 1
    End While

    ' intCompCount += 1
    If Low <= High Then
        Swap(Low, High)
        Low = Low + 1
        High = High - 1
    End If
Loop While Low <= High

If intFirst < High Then QuickSort(intSubList, intFirst, High)
If Low < intLast Then QuickSort(intSubList, Low, intLast)

```

Swap

```

Sub Swap(intNdxA As Int32, intNdxB As Int32)
    Dim intTemp As Int32

    intTemp = arr[name](intNdxA)
    arr[name](intNdxA) = arrNumbers(intNdxB)
    arr[name](intNdxB) = intTemp
End Sub

```

Function Structure

```
Sub [name]([variable] As [data type])  
End Sub  
  
[name]([parameter])
```

CSV

Load

```
Using MyReader As New Microsoft.VisualBasic.FileIO.TextFieldParser("[file  
location"])  
    MyReader.TextFieldType = FileIO.FieldType.Delimited  
    MyReader.SetDelimiters(",")  
  
    While Not MyReader.EndOfData  
        [variable] = MyReader.ReadFields() ' Read a new row  
    End While  
End Using
```