Creating an event driven computer program using Visual Basic.NET

Level 2

Notes

for

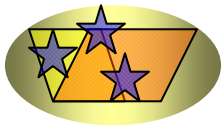
City & Guilds

7540 Unit 009

Written for Microsoft Visual Basic.NET®

Tench Computing Ltd
Pines
Glendale Road
Burgess Hill
West Sussex
RH15 0EJ

Web address: www.tenchcomputing.co.uk
Email address: jtench@globalnet.co.uk



About the author: *Jackie Tench MSc, ACIB, Cert Ed*

Jackie started her working career in branch banking with the Midland Bank (now HSBC) and was transferred to their Computing Department after achieving 100% in their ability test for programmers. She then worked for more than a decade in this department and was one of the first women to achieve a junior management grade at the age of 21. She attended a significant number of IBM programming training courses during her time there.

Jackie was the first woman to pass the ACIB (Associate Chartered Institute of Bankers) examinations in the Midland Bank (HSBC) and the youngest person at 21 years of age.

Jackie then left to raise a family but still found time to teach part-time at a college in Sheffield and to obtain a MSc in Computing and a Cert Ed in teaching.

When her children were old enough Jackie returned to work full-time and was a Senior Lecturer in Software Engineering and Computer Studies at a college in Brighton for nearly 10 years teaching all levels up to and including HND.

Therefore, Jackie has considerable business knowledge and qualifications plus wide experience in practical computing and training – covering areas such as structured design, analysis, coding, testing and implementing software applications plus training students to fulfil an important role in the computer industry.

Jackie has worked as a consultant for several blue chip companies and examination boards using her software engineering and educational training skills and is now one of the foremost experts in computing with an extensive knowledge of programming languages and applications.

Contents Page

Chapter 9

Objectives

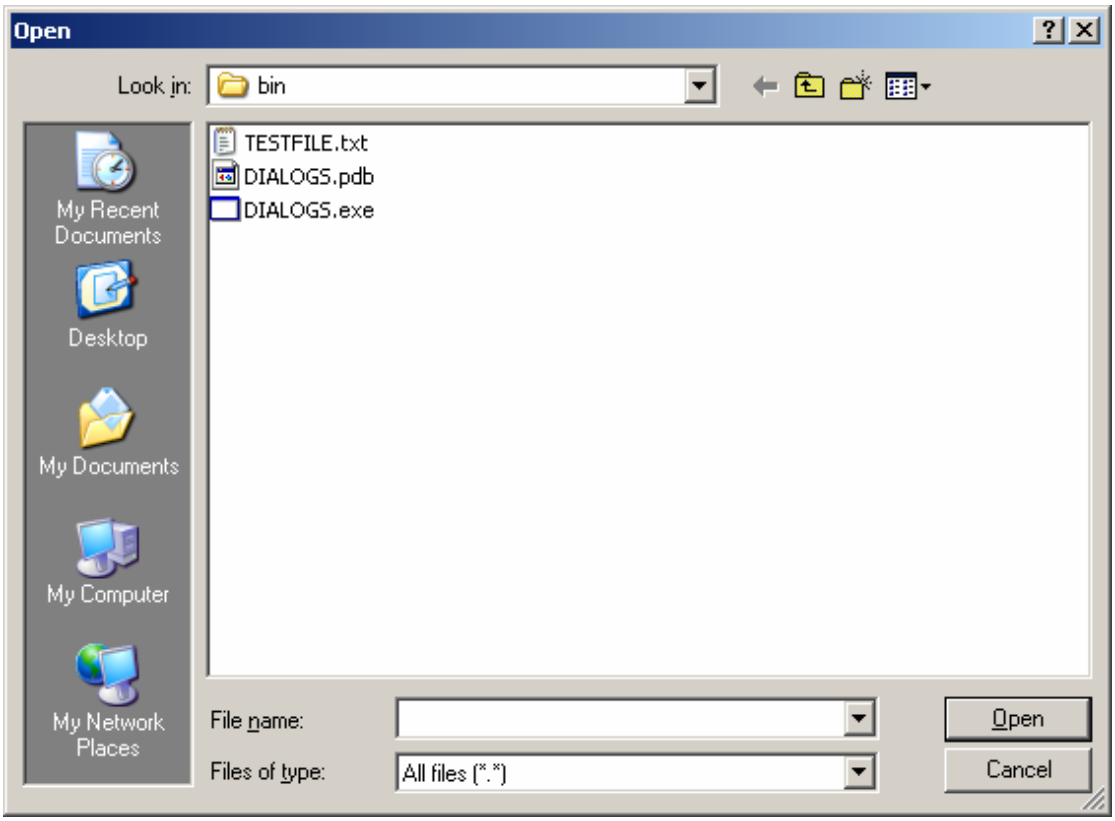
- Use the OpenFileDialog and SaveFileDialog controls
- Create, read from and write to sequential text files

Tasks such as opening files and saving files must be carried out by most programs. These tasks usually involve interaction with the user. The dialog controls allow you to display the standard dialog boxes used by Windows. This saves the programmer the rather tedious task of creating the dialog boxes. It also ensures that these dialog boxes are the same across all Windows programs making it easier for users to use programs.

OpenFileDialog control

The **OpenFileDialog** control is not visible at run time. A default name **OpenFileDialog1** is used for the **Name** property of the control but this name can be changed.

When the OpenFileDialog control is used in a program the following **Open** dialog appears on the screen so that the user can select which file to open.



ShowDialog() method

The ShowDialog() method is used to display a dialog box. If an OpenFileDialog1 is present on a form the following code will show the open dialog.

```
OpenFileDialog1.ShowDialog()
```

FileName property

The **FileName** property is used with the openFileDialog control to obtain the filename selected by the user as shown in the code extract below:

```
Dim sFileName As String

If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
' OpenFileDialog1.FileName contains the filename of the file the
' user selected
      sFileName = openFileDialog1.FileName
```

Filter property

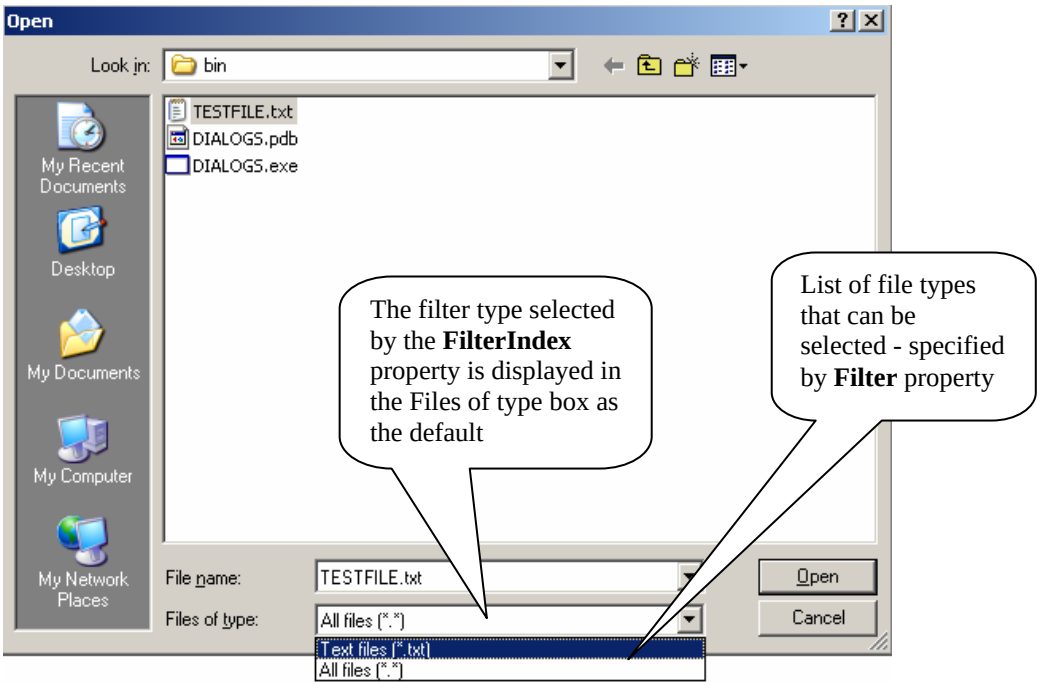
The Filter property is used with the openFileDialog to restrict the files displayed to only certain types of file. For example, to display only *.txt files in the dialog window the Filter property is set as below:

```
OpenFileDialog1.Filter = "Text files(*.txt)|*.txt"
```

The pipe | symbol is used to separate the options in the list. **Text files(*.txt)** is the text that is displayed to the user in the **Files of type** box. ***.txt** is used by the system to display the type of files in the window i.e. all files with an extension of **txt**.

It is possible to have more than one filter and the user can select one of the filters using a drop down list. For example, the code below creates two filters.

```
OpenFileDialog1.Filter = "Text files(*.txt)|*.txt|All Files(*.*)|*.*"
```



FilterIndex property

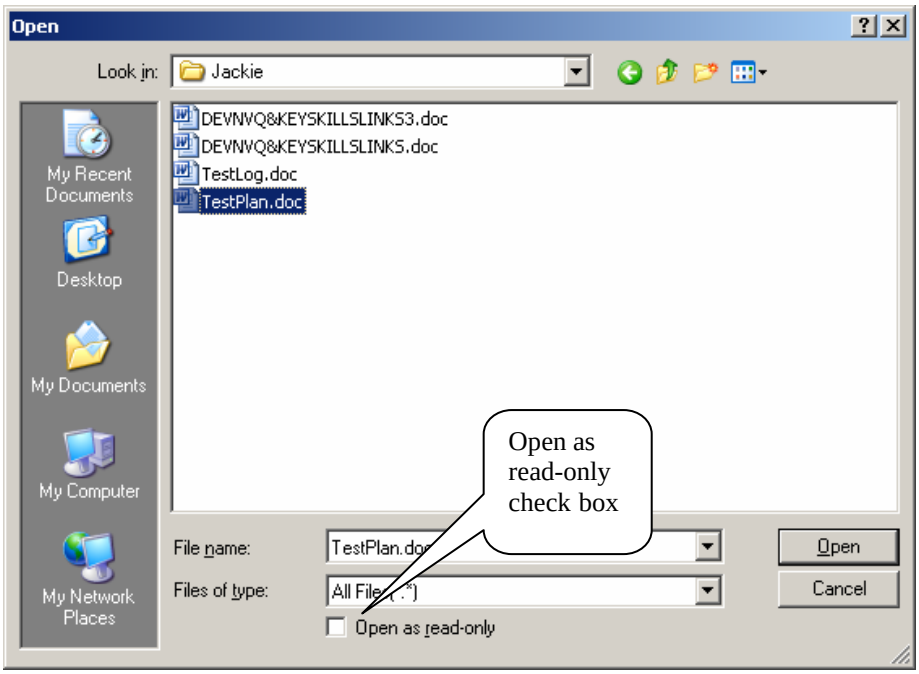
The index of the filter to be displayed to the user can be set using the FilterIndex property. The first filter has a value of 1.

```
OpenFileDialog1.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*"
   'The FilterIndex is used to select which text from the filter list
   'is to be displayed in the Files of Type box on the Open dialog
    OpenFileDialog1.FilterIndex = 2
```

For the code shown above setting the **FilterIndex** to 2 will display the text **All files (*.*)** in the **Files of type** box when the dialog is displayed.

ShowReadOnly property

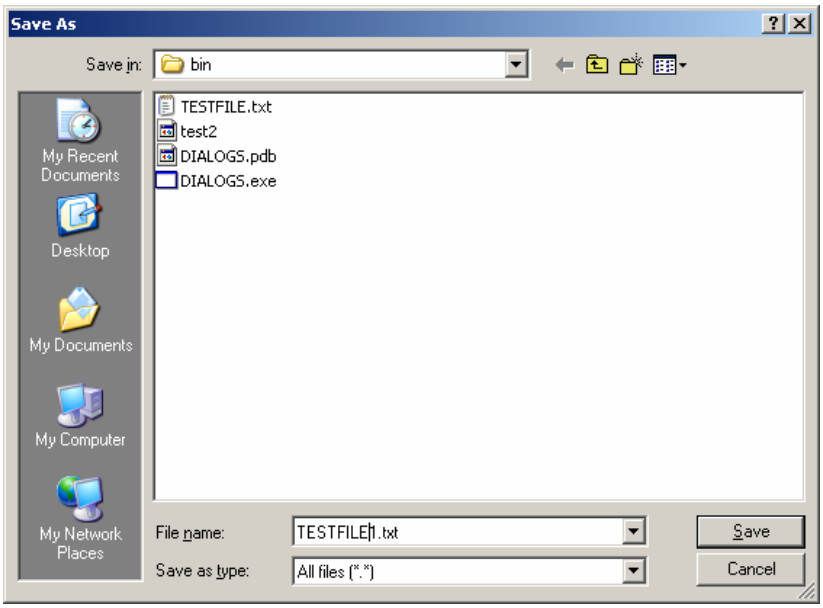
The ShowReadOnly property defaults to **False**. When this property is set to **True** the **Open as read-only** check box appears on the OpenFileDialog. If the user selects this check box the file will be opened as read only.



SaveFileDialog control

The SaveFileDialog control is not visible at run time. A default name **SaveFileDialog1** is used for the **Name** property of the control but this name can be changed.

When the SaveFileDialog control is used in a program the following **Save As** dialog appears on the screen so that the user can enter the filename for the file.



The FileName, Filter and FilterIndex properties work in the same way for this control as for the OpenFileDialog.

The ShowDialog() method works in the same way for this control as for the OpenFileDialog.

OverwritePrompt property

The OverwritePrompt property is set to **True** as the default. When this property is set to True a prompt will be displayed to warn the user if they have selected a filename which already exists when they save a file.

CheckFileExists property

The CheckFileExists property is set to **False** as the default. When this property is set to True a check is made that the specified filename exists before returning from the dialog.

Dialogs program

- Create a new project and save the project as **DIALOGS** and save the form as **FormDIALOGS.vb**.

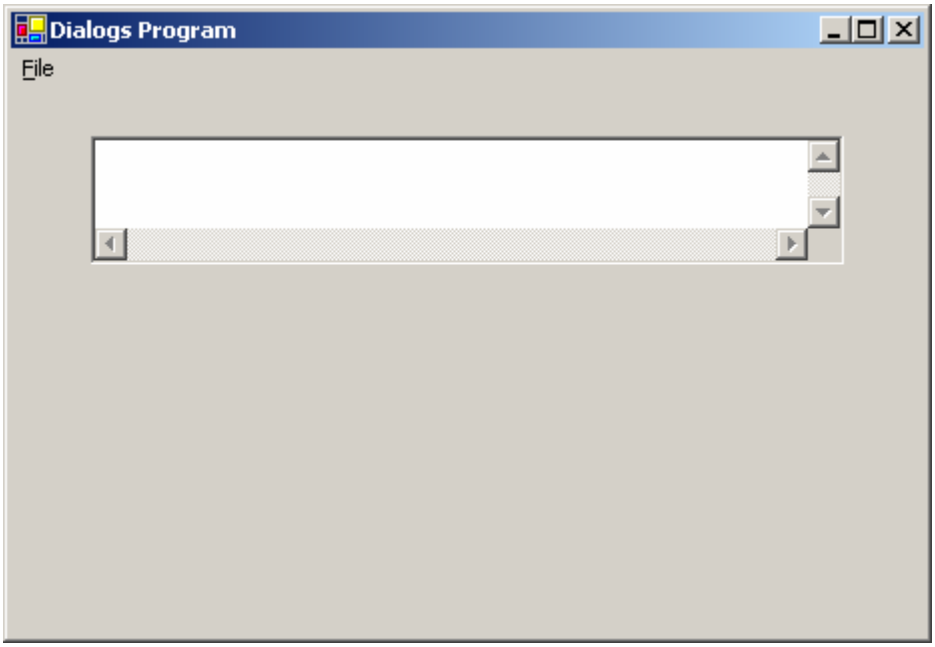
Build the form from the following table.

Object	**Property**	**Setting**
Form	Name	frmDialogs
	Text	Dialogs Program
	Size	464,320
	StartPosition	CenterScreen
TextBox	Name	txtUser
	Multiline	True
	Text	(empty)
	WordWrap	False
	Scrollbars	Both
	Location	40,24
	Size	376,64
OpenFileDialog	Name	OpenFileDialog1
SaveFileDialog	Name	SaveFileDialog1
	CheckFileExists	False

- Add a **MainMenu** control to the form and name it **MainMenu1**.
- Create the following menu:

Text	**Name**
&File	mnuFile
&Open	mnuFileOpen
&Save	mnuFileSave
Save &As	mnuFileSaveAs
-	mnuFileSep1
E&xit	mnuFileExit

The form should appear as shown when completed.



- Add the variable **bTextChanged** outside of any event procedure as shown below.

```
Public Class frmDialogs
    Inherits System.Windows.Forms.Form
'bTextChanged is True if user has changed text since the last
'file save operation
    Private bTextChanged As Boolean
```

- Enter the following code in the appropriate procedures.

```
Private Sub frmDialogs_Load()
'text not yet changed
        bTextChanged = False
End Sub

Private Sub mnuFileExit_Click()
   End
End Sub

Private Sub mnuFileOpen_Click()
'displays Open dialog box
     Dim nFileNo As Integer
     Dim sFileName As String
'OpenFileDialog1 is the default name of the OpenFileDialog
'control on this form
      Dim NextLine, AllLines As String
      AllLines = ""
'The files displayed will be in the current directory
'if another directory is required it can be set using
'the InitalDirectory property of the OpenFileDialog
     OpenFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*"
'The FilterIndex is used to select which text from the filter list
'is to be displayed in the Files of Type box on the Open dialog
     OpenFileDialog1.FilterIndex = 2
     nFileNo = FreeFile()
`Displays Open dialog box
     If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
```

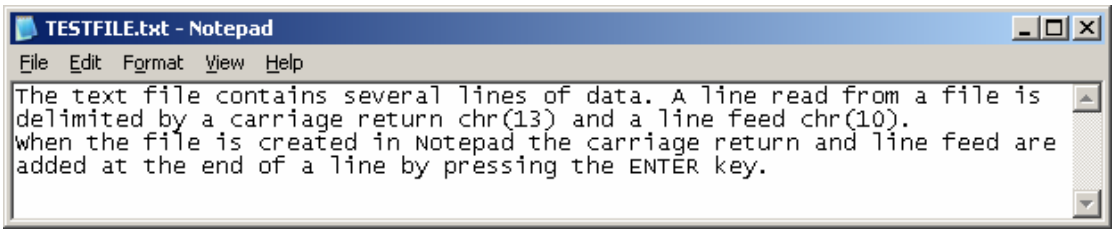
```
' OpenFileDialog1.FileName contains the filename of the file the
' user selected
         sFileName = OpenFileDialog1.FileName
         FileOpen(nFileNo, sFileName, OpenMode.Input)
         Do While Not EOF(nFileNo)
              NextLine = LineInput(nFileNo)
'The carriage return and linefeed are not read from the file
'so have to be inserted if required for the display
'of the text
              AllLines = AllLines & NextLine & Chr(13) & Chr(10)
         Loop
         txtUser.Text = AllLines
         FileClose(nFileNo)
    End If
End Sub
```

Event procedure for multiple controls

Note that for the mnuFileSaveAs_Click event procedure you must change the **Handle**s clause - `Handles mnuFileSaveAs.Click, mnuFileSave.Click.` The Handles clause specifies that the mnuFileSaveAs_Click event procedure will handle events for the mnuFileSaveAs_Click event and the mnuFileSave_Click event. This demonstrates how one event procedure can handle events from multiple controls. In this case it saves writing two event procedures, one for the **Save** menu option and one for the **Save As** menu option which would both contain the same code. When the user selects either the **Save** or the **Save As** menu option the mnuFileSaveAs_Click procedure is executed.

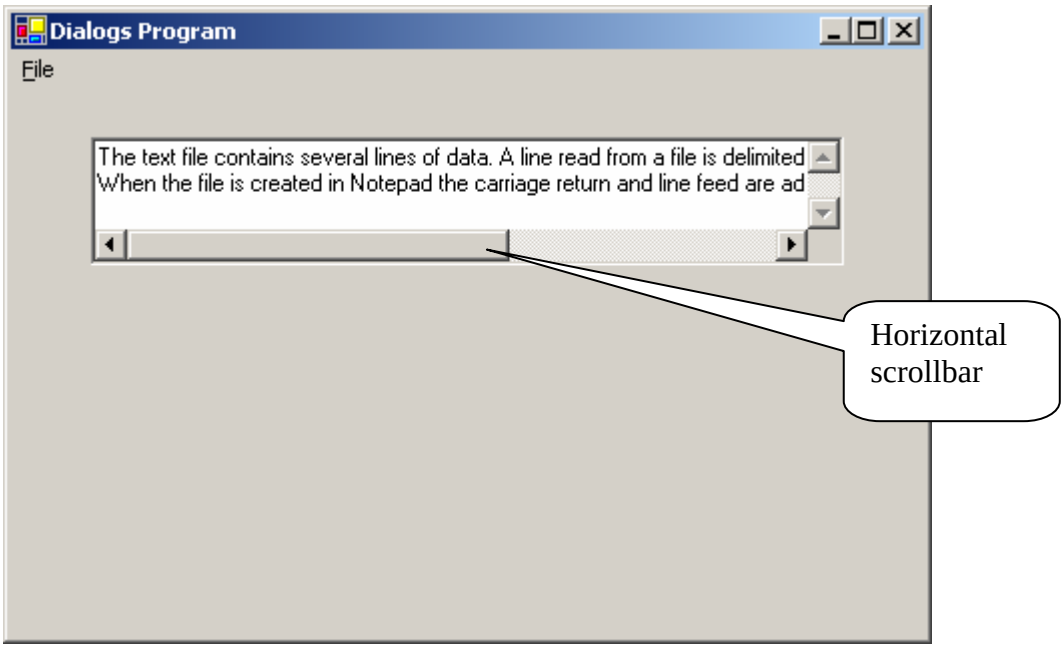
```
Private Sub mnuFileSaveAs_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles mnuFileSaveAs.Click, _
      mnuFileSave.Click
        Dim nFileNo2 As Integer
        Dim sFileName As String
'if code has not been saved then user can choose file to save in
     SaveFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*"
'The FilterIndex is used to select which text from the filter list
'is to be displayed in the Files of Type box on the Open dialog
        SaveFileDialog1.FilterIndex = 2
        nFileNo2 = FreeFile()
        ' displays Save dialog box
        If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
' saveFileDialog1.FileName contains the filename of the file the
' user selected
            sFileName = SaveFileDialog1.FileName
            If bTextChanged = True Then
                nFileNo2 = FreeFile()
                FileOpen(nFileNo2, sFileName, OpenMode.Output)
                'save txtUser data to file sFileName
                Print(nFileNo2, txtUser.Text)
                bTextChanged = False
                FileClose(nFileNo2)
            End If
        End If
End Sub

Private Sub txtUser_TextChanged()
'This procedure is executed when the text in the text box is changed
'by the user
    bTextChanged = True
End Sub
```

- Set the **Startup object** to **frmDialogs**.
- Create a file named **TESTFILE.txt** in NotePad to contain the text shown below. Make sure that you press the ENTER key after each paragraph.



- Test that the Dialogs program opens and saves text files correctly. Change the text in the text box and save the file and then check that the file has been saved correctly.

Note that when the file TESTFILE.txt is opened and displayed in the text box you cannot view all the text at the right unless you use the horizontal scroll bar.

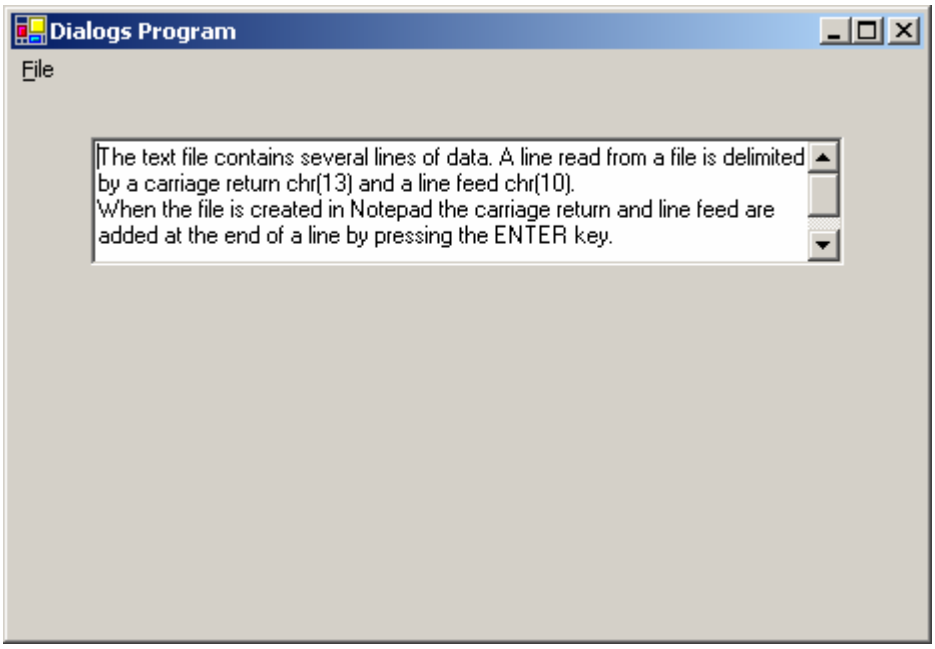


WordWrap property

The WordWrap property of a text box is set to **True** as the default. We set the WordWrap property of the text box to **False**. When the WordWrap property is set to True the text in the text box is wrapped at the right so that all the text is visible. When the WordWrap property is set to False the text is not wrapped at the right so that the text is not visible. This means that the **Scrollbars** property must be set so that a user can scroll the text so that it becomes visible. The Scrollbars property was set to **Both** so that a Vertical and Horizontal scrollbar appear on the text box.

- Set the **WordWrap** property of the text box **txtUser** to **True**.
- Set the **ScrollBars** property of the text box **txtUser** to **Vertical**.

Only the Vertical scrollbar is required when the WordWrap property is set so that a user can scroll the text up and down if the all the text is not fully displayed.



The text is wrapped around at the right so that it is visible. Only one scrollbar is needed – the **Vertical** scrollbar.

Perform a Click() procedure

You can perform an existing Click() procedure in your program from another event procedure. For instance to perform the event procedure **mnuFileSaveAs_Click()** from the **mnuFileExit_Click()** event procedure in the Dialogs program the following code would be used.

```
mnuFileSaveAs.PerformClick()
```

When the procedure **mnuFileSaveAs_Click()** is called it will be executed and then the program continues executing from the statement after the call in the calling procedure.

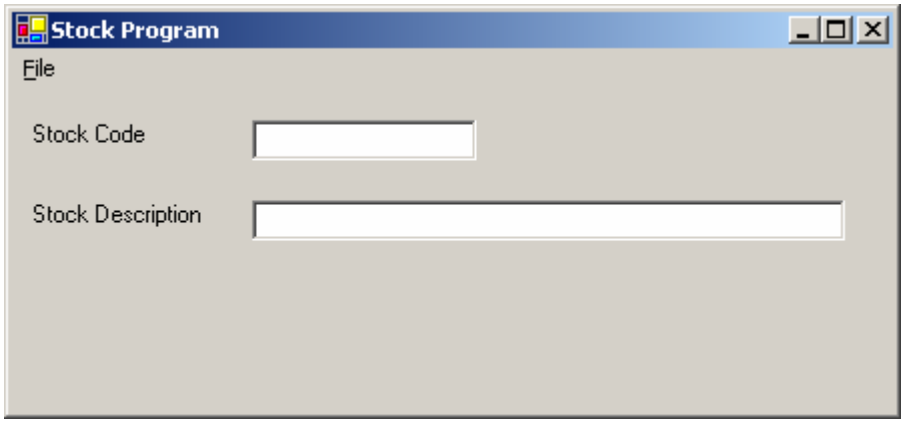
Questions 9

1. In the mnuFileExit_Click() event procedure of the Dialogs program add code to do the following:

   - if the contents of txtUser have been changed use a message box with Yes, No and Cancel buttons to inform the user that the text has changed and ask if the user wants to save the changes.
   - if the user presses the Cancel button then do not terminate the program.
   - if the user presses the No button then terminate the program.
   - if the user presses the Yes button then perform the mnuFileSaveAs_Click() event procedure and then terminate the program.

   Test your amended code thoroughly.

2. You are required to create and test a user interface to create and add stock records to a sequential text file. Your completed form should appear similar to the screen image shown below.



   - Save the project as **STOCK** and save the form as **FormSTOCK.vb**.
   - Declare an integer variable **TextFileNum** and a string variable **OpenFileName** in the code outside any procedure.
   - Set the **Text** property of the form to: **Stock Program**
   - Set the **Name** property of the form to **frmStock**
   - Set up a menu with the following options.

Option	**Name**
&File	mnuFile
&New	mnuNew
&Open	mnuOpen
&Close	mnuClose
-	mnuSep1
E&xit	mnuExit

- Draw, size and position two text box controls. Name the controls **txtStock** and **txtStockDesc**.
- Draw, size and position two label controls. Name the controls **lblStock** and **lblStockDesc**. Set their **Text** properties to **Stock Code** and **Stock Description** respectively.
- Draw, size and position a button. Name the button **btnAdd** and set the **Text** property to **Add** and the **Visible** property to **False**.
- Add an **OpenFileDialog** control to the form and set the **DefaultExt** property to **txt**. (This will add the extension **txt** if the user does not enter an extension for the file.)
- Code the **frmStock_Load()** procedure as follows:

  set the string variable **OpenFileName** to empty

  set the **Enabled** property of the **Close** menu option to **False**.
- Code the **mnuNew_Click()** procedure as follows:

  set up file filters to give a choice of **All files** or **.txt** files in the dialog

  set the **FilterIndex** property of the dialog to default to **All files**

  assign the **FreeFile** number to the variable **TextFileNum**

  set the **CheckFileExists** property of the **OpenFileDialog** to **False**

  display the OpenFileDialog

  set the **OpenFileName** variable to the dialog **FileName** property

  open the selected file as **Output**

  set the **Visible** property of the **btnAdd** button to **True** (the Add button is only visible when a record can be added)

  set the **Enabled** property of the **New** menu option to **False**

  set the **Enabled** property of the **Open** menu option to **False**

  set the **Enabled** property of the **Close** menu option to **True**.
- Code the **mnuOpen_Click()**procedure as follows:

  set up file filters to give a choice of **All files** or **.txt** files in the dialog

  set the **FilterIndex** property of the dialog to default to **All files**

  assign the **FreeFile** number to the variable **TextFileNum**

  display the OpenFileDialog

  set the **OpenFileName** variable to the dialog **FileName** property

  open the selected file as **Append**

  set the **Visible** property of the **btnAdd** button to **True** (the Add button is only visible when a record can be added)

  set the **Enabled** property of the **Open** menu option to **False**

  set the **Enabled** property of the **New** menu option to **False**

set the **Enabled** property of the **Close** menu option to **True**.

- Code the **mnuClose_Click()** procedure as follows:

  if a file is open

    close the file

    set the **Visible** property of the **btnAdd** button to **False**

    set the string variable **OpenFileName** to empty

    set the **Enabled** property of the **Open** menu option to **True**

    set the **Enabled** property of the **New** menu option to **True**

    set the **Enabled** property of the **Close** menu option to **False**

  otherwise

    display the error message "001 No file open"

- Code the **mnuExit_Click()** procedure as follows:

  if a file is open

    close the file

  terminate the program.

- Code the **btnAdd_Click()** procedure as follows:

  write the data in the Text properties of the text box controls to the file

  clear the data from the text box controls

  set the focus to the **txtStock** text box (**txtStock.Focus()**).

- Set the **Startup object** to **frmStock**.
- Create test data for the stock records, test the program for all the menu options, print the records in the file (use Notepad) and compare the output file with the test data created.
- Print a program listing and a printed copy of the form frmStock.

You should make sure that your program

conforms to the design specification

uses the most appropriate data type(s)

uses meaningful names when declaring variables

syntax is consistently indented to aid readability

is commented.

[This page is intentionally blank]