

**Creating an event
driven computer
program using
Visual Basic.NET**

Level 2

Notes

for

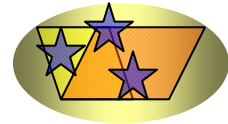
City & Guilds

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.

Copyright ©1999 Tench Computing Ltd

Microsoft, Windows, Windows NT or other Microsoft products referenced herein are either the trademarks or registered trademarks of the Microsoft Corporation. Other trademarks for products referenced herein are also acknowledged.

All rights are reserved and no part of this training manual may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the purchase of a licence.

This training manual is sold subject to licence and on condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without the prior consent of Tench Computing Ltd in any form of binding or cover other than that in which it is issued and without a similar condition being imposed on the subsequent purchaser. Any program listings within this training manual may be entered, stored and executed in a computer but they may not be reproduced for publication.

Contents	Page
Visual Basic.NET development environment	1
Toolbars	
Standard toolbar	3
Toolbox	5
Components of a program	
Event procedures	8
Properties	8
Solution Explorer	9
Form design window.....	9
Code window	10
Properties Window	10
Chapter 1	
Objectives.....	11
Demo program.....	11
Create a new project	12
Rename and save the form file.....	13
Solution Explorer window	15
Change the Text property of the form.....	15
Properties	16
Change the Name property of the form	17
Change the size of the form	18
Save the files	18
Add the Exit button to the frmDemo form	18
Toolbox window.....	19
Place the Exit button on the form.....	19
Change the Name property of the Exit button	20
Change the Text property of the Exit button	20
Change the location of the Exit button.....	20
Change the Font properties of the Exit button	21
Add the other buttons to the frmDemo form	21
Change the Name properties	22
Change the Text properties	22
Resize the buttons.....	22
Change the Font properties.....	22
Add a TextBox to the frmDemo form	22
Change the properties of the TextBox.....	22
Attach code to the objects	24
Attach code to the Exit button.....	24
Code for the btnExit_Click() procedure.....	25
Set the startup form.....	25
Execute the Demo program.....	26
Attach code to the Display button.....	26
Attach code to the Clear button	27
Execute the Demo program.....	27
Executable file (DEMO.EXE)	28
Build a form from a table	28
Questions 1	29

Chapter 2

Objectives.....	31
Scroll bar	31
MPH program	31
Create the form.....	32
Code of the MPH program.....	33
Execute the MPH program	33
Minimum and Maximum properties of the scroll bar	33
Value property of the scroll bar.....	33
Keyboard focus.....	34
Attach more code to the MPH program	34
Scroll bar properties	
Minimum and Maximum properties	35
SmallChange and LargeChange properties	35
CHOICE program	36
Enter the code of the CHOICE program	37
Execute the CHOICE program	37
Enter more code	38
Execute the CHOICE program	39
How the CHOICE program works	
Concatenation character	40
Comment	40
CheckedChanged() event procedure	40
Name conventions for controls	41
Questions 2	42

Chapter 3

Objectives.....	45
Arithmetic operators	45
MULTIPLY program.....	45
Enter the code of the MULTIPLY program	47
Execute the MULTIPLY program.....	47
Relational operators	50
Decision Statements	
If...End If construct.....	50
Select Case...End Select construct	50
Loops	
Do While...Loop	51
Do...Loop While	52
For...Next.....	52
ADD program	53
Execute the ADD program.....	55
How the ADD program works	55
Timer program	56
Execute the Timer program	57
How the Timer program works	57
Modify the Timer program	57
Execute the Timer program	58
Questions 3	59

Chapter 4

Objectives.....	61
GroupBox control.....	61
Enter the code of the Group program	63
Arithmetic operator precedence	67
Debugging	
Compiler.....	68
Option Explicit	68
Executable file (EXE)	69
Syntax errors.....	69
Reserved words	69
Data type mismatch	69
Logical errors	69
Run-time errors	70
Breakpoint and watches.....	70
Check the value of a variable	71
Add Watch	71
Sub Main()	72
Refresh() method.....	72
Idle time.....	72
GroupBox control.....	72
Select multiple objects.....	72
Lasso objects	73
Print a code listing	73
Print a form	73
Question 4	75

Chapter 5

Objectives.....	79
Menus.....	79
Menu program	79
Create the menu	80
Enter the code of the Menu program	83
Add a separator bar to a menu	84
Check marks	84
Make a menu item invisible	85
Questions 5	86

Chapter 6

Objectives.....	87
PictureBox control	87
Twip	87
Stand-alone application	87
Different types of picture files	
Bitmap files	87
Icon files.....	87
Move controls	87
MovePic program	88
Code the MovePic program.....	89
The MOON program.....	90
Enter the code of the MOON program.....	92
Questions 6	94

Chapter 7

Objectives.....	95
MessageBox class.....	95
Show() method	95
MESSAGE program	98
Enter the code	98
DIALOG program.....	100
Enter the code	100
Run the DIALOG program	102
InputBox() function	103
Add an InputBox to the DIALOG program	103
Other parameters of the InputBox() function	105
Custom dialog box.....	106
Design a custom dialog box	106
Standard dialog box properties.....	108
AcceptButton property	109
CancelButton property.....	109
Display and hide a custom dialog box	109
Questions 7	112

Chapter 8

Objectives.....	113
NUMBER program.....	113
The code of the NUMBER program.....	114
Modify the NUMBER program	115
TabIndex property	115
Cursor property	116
Access files	
Sequential text file.....	117
Open a sequential file for output	117
Open a sequential file for append	118
Open a sequential file for input	119
ADDRESS program.....	120
The code of the ADDRESS program	121
Enhance the ADDRESS program.....	122
Questions 8	124

Chapter 9

Objectives.....	127
OpenFileDialog control.....	127
ShowDialog() method.....	127
FileName property.....	128
Filter property	128
FilterIndex property	128
ShowReadOnly property	129
SaveFileDialog control.....	129
Overwrite property.....	130
CheckFileExists property	130
Dialogs program	130
Event procedure for multiple controls	132
WordWrap property	133
Perform a Click() procedure	134

Questions 9	135
Chapter 10	
Objectives.....	139
FontDialog control	139
ColorDialog control	140
FullOpen property	140
Font and Color program	141
Functions	
Int function	143
Val function	143
Str function.....	143
FormatNumber function	143
IsNumeric function	144
IsDate function	144
Logical operators.....	145
LOGICAL program.....	146
Testing	
Purpose of testing	148
Test data	148
Evidence of compliance.....	149
Logical program test data	149
Event/action chart.....	149
Add a control to the Toolbox.....	150
Remove a control from the Toolbox.....	150
Convert class	
ToString() method	151
ToInt32() method	151
Question 10	152
Chapter 11	
Objectives.....	155
PasswordChar property	155
Quote program	155
Console.WriteLine() method	162
Help facilities	162
Questions 11	164
Chapter 12	
ListBox control.....	169
Objective	169
List program	171
Objectives.....	171
Graphics	173
Objective	173
DrawRectangle() method.....	173
Pen object	173
CreateGraphics() method	173
Dispose() method	173
Drawtest program	174
SolidBrush object.....	176
FillRectangle() method	176
MouseMove() event.....	177

MouseDown() event	177
MouseUp() event.....	177
DrawEllipse() method	178
FillEllipse() method	179
DrawLine() method	180
Clear() method	182
Question 12	183
Sample assignment	184
Sample questions	187

Chapter 9

Objectives

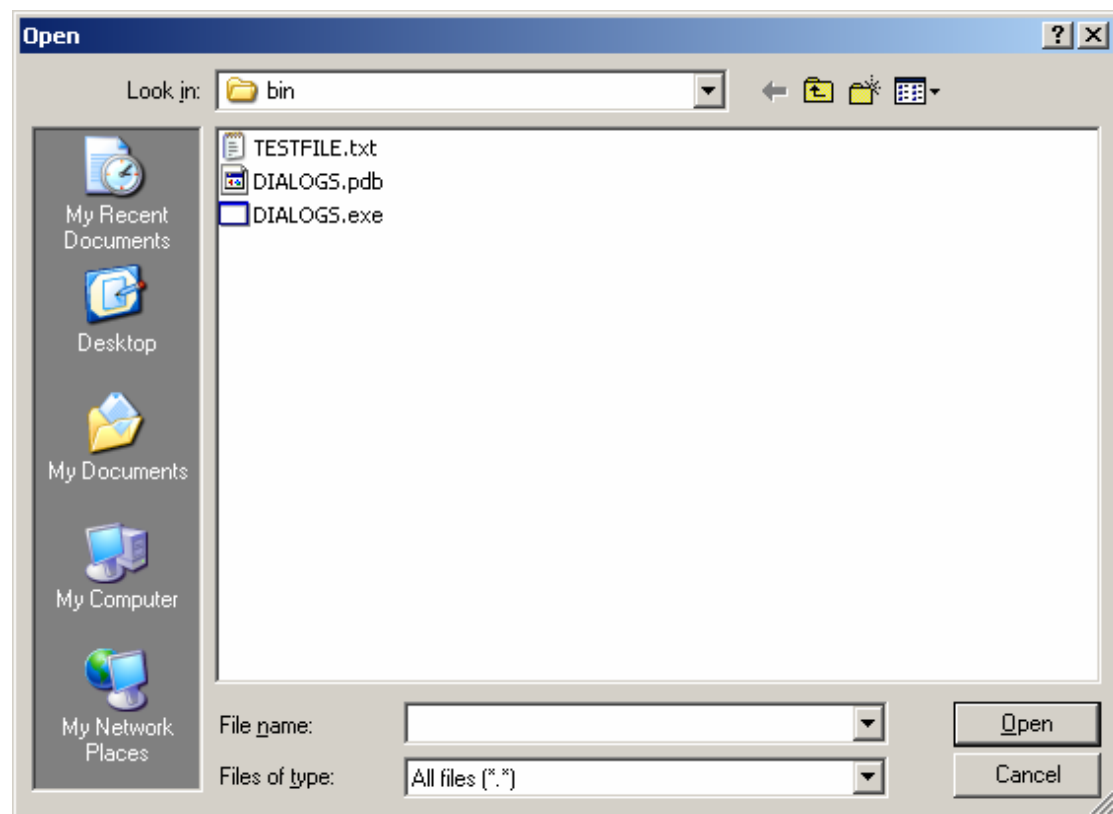
- Use the OpenFileDialog and SaveFileDialog controls
- Create, read from and write to sequential text files

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

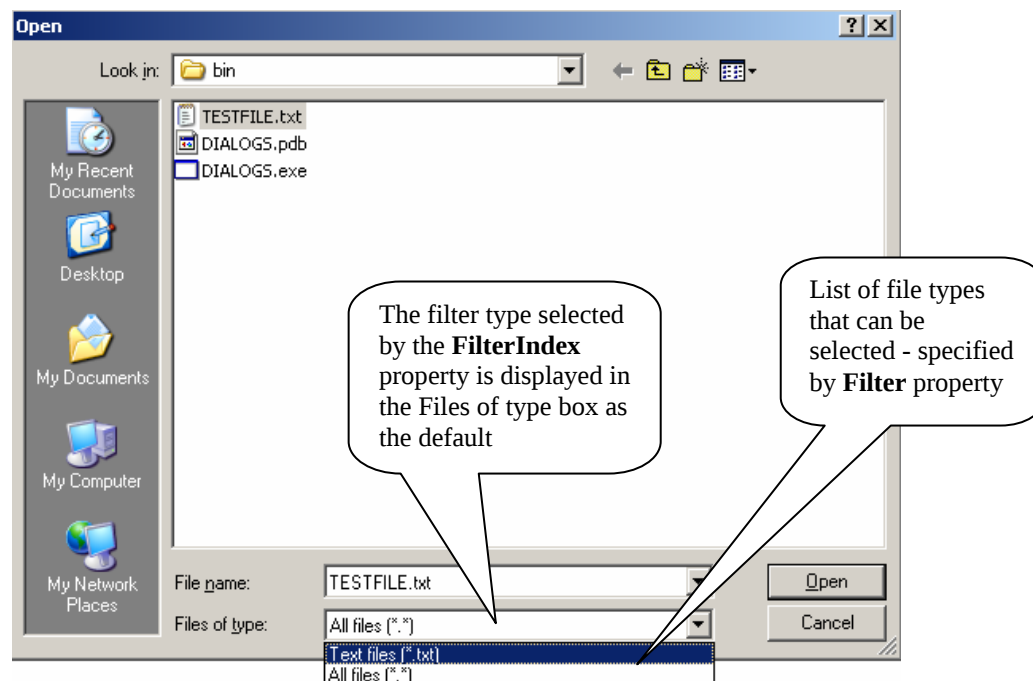
The **Filter** property is used with the `openFileDialog` to restrict the files displayed to only certain types of file. For example, to display only *.txt files in the dialog window the **Filter** property is set as below:

```
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

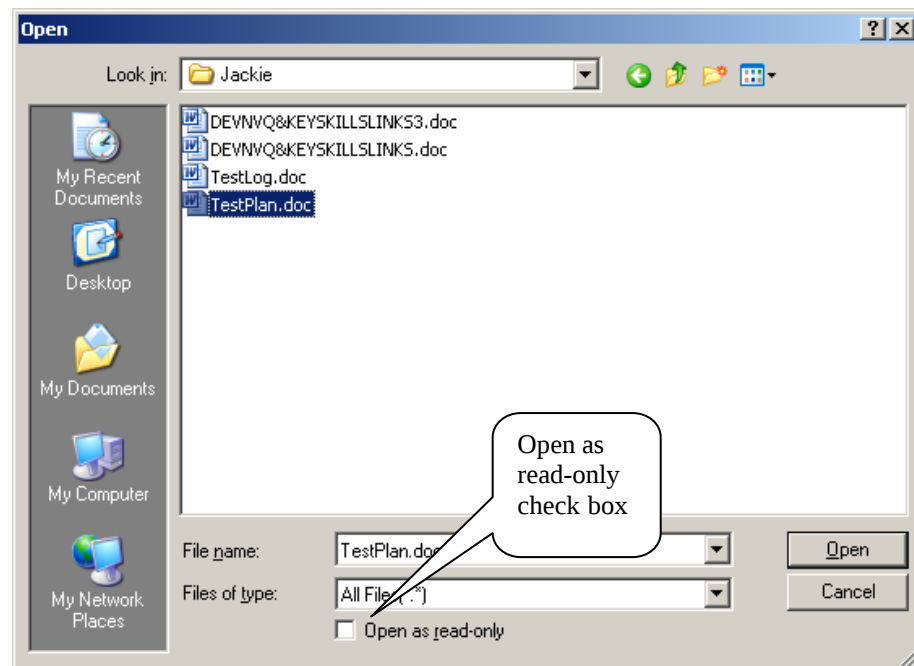
The index of the filter to be displayed to the user can be set using the **FilterIndex** property. The first filter has a value of 1.

```
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

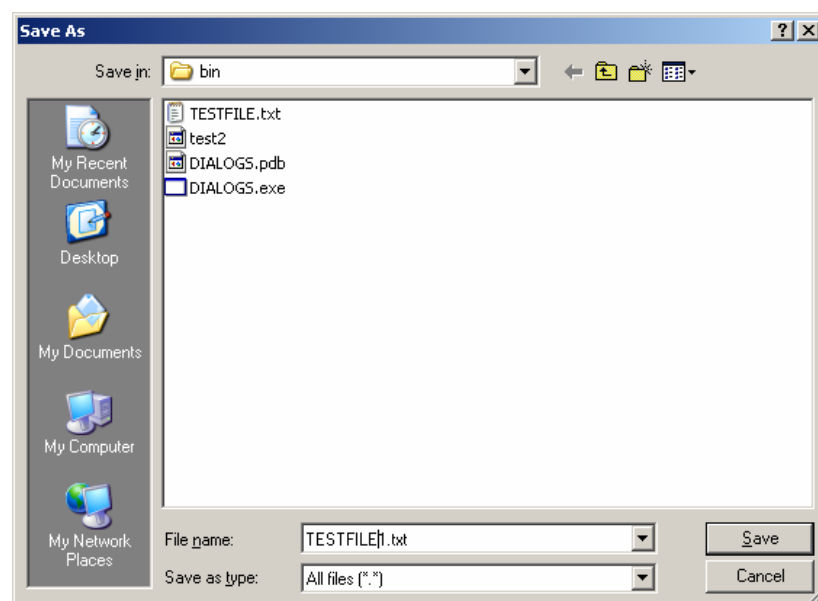
The ShowReadOnly property defaults to **False**. When this property is set to **True** the **Open as read-only** check box appears on the OpenFileDialog. If the user selects this check box the file will be opened as read only.



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**.

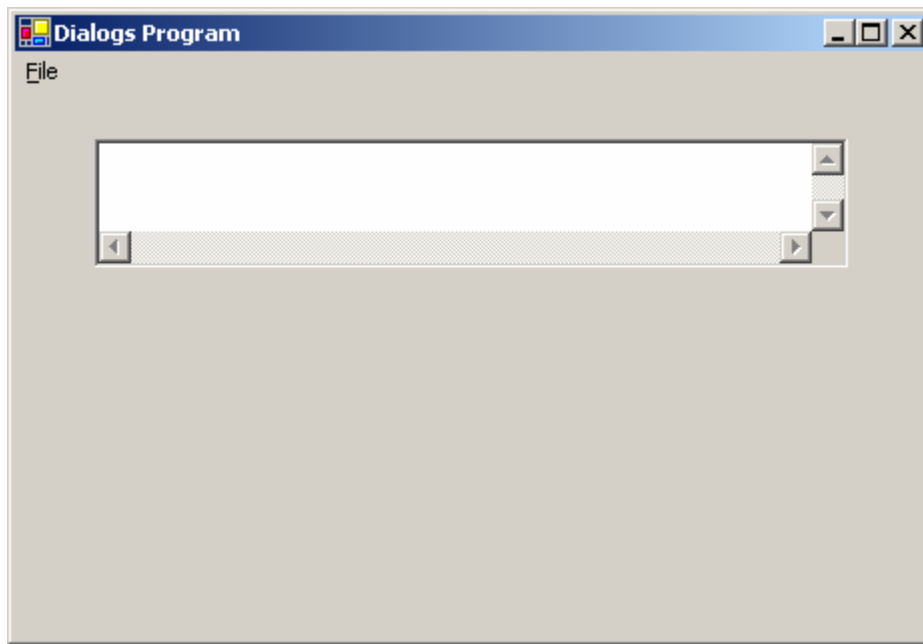
Build the form from the following table.

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 InitialDirectory 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 **Handles** 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.

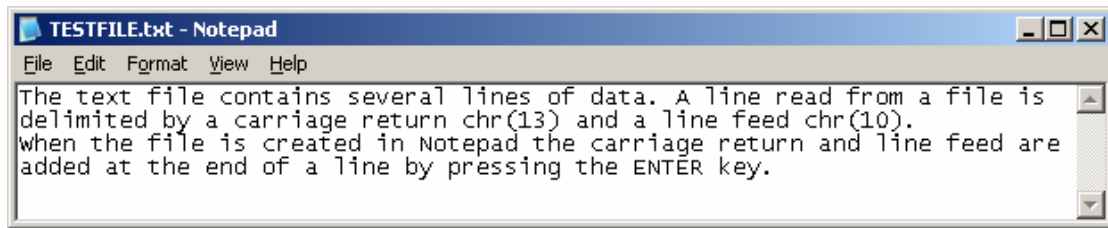
```

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

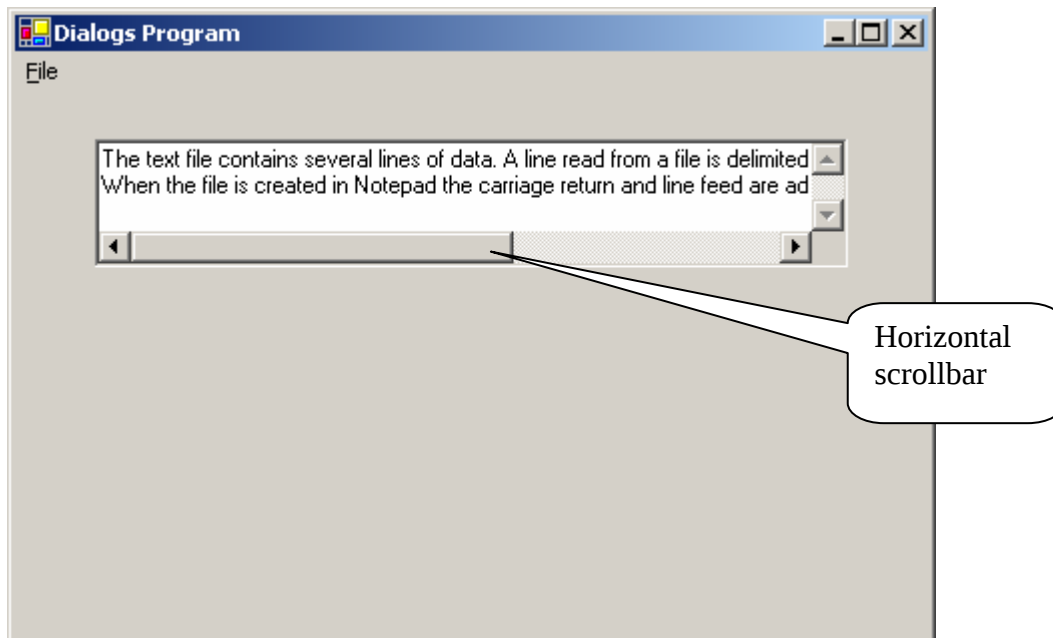
```

- 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 scrollbar.

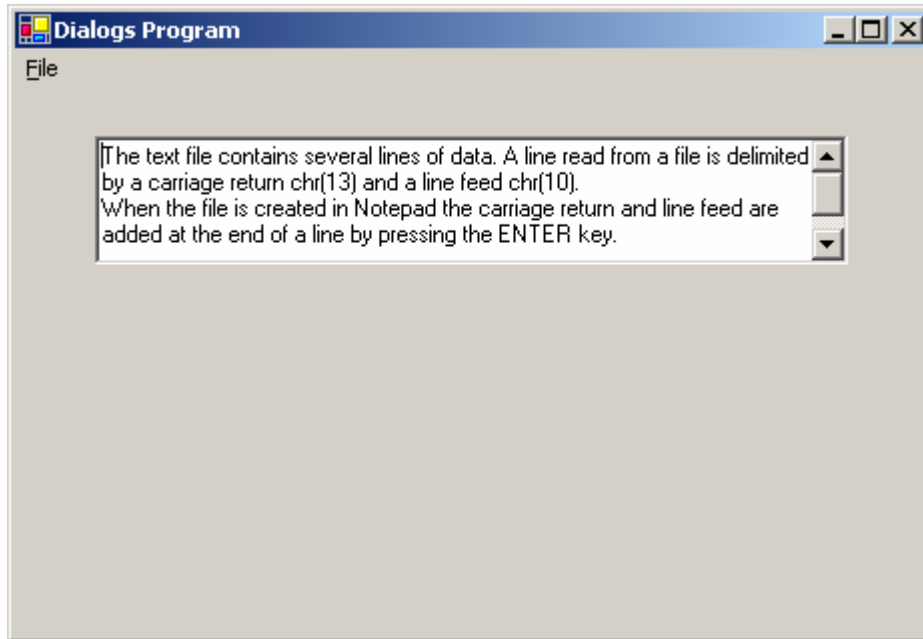


WordWrap property

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]