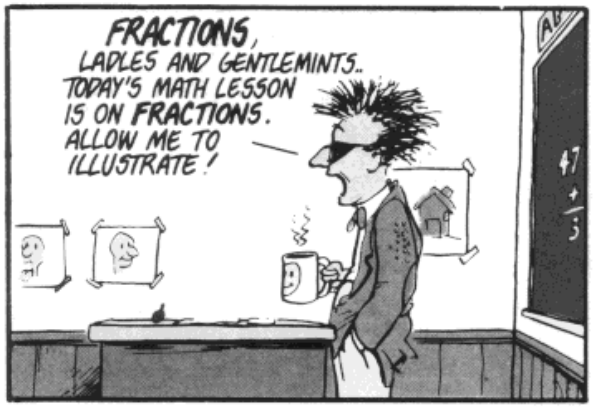
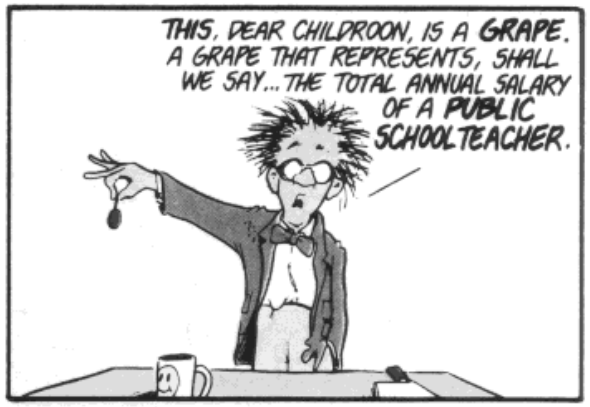
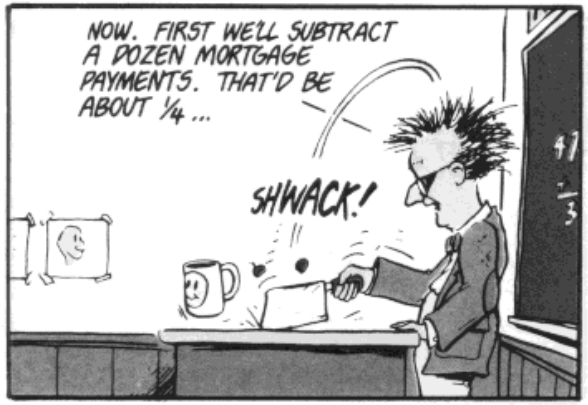
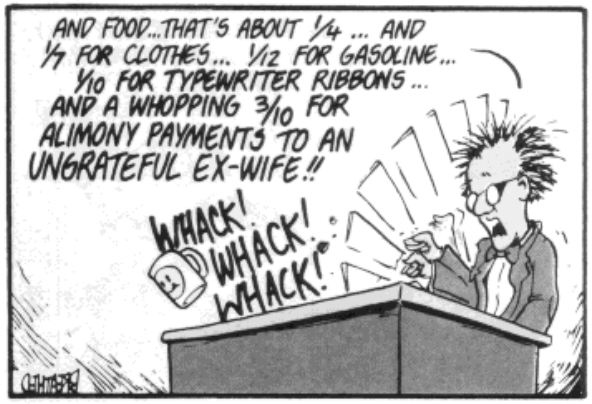
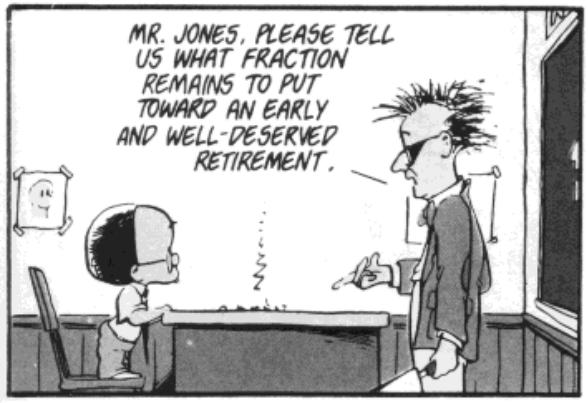
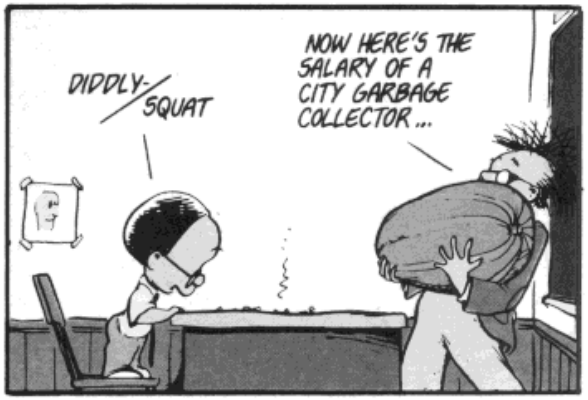

FRACTIONS, LADIES AND GENTLEMINTS.. TODAY'S MATH LESSON IS ON FRACTIONS. ALLOW ME TO ILLUSTRATE!
47 + 1/3


THIS, DEAR CHILDROON, IS A GRAPE. A GRAPE THAT REPRESENTS, SHALL WE SAY... THE TOTAL ANNUAL SALARY OF A PUBLIC SCHOOLTEACHER.


NOW. FIRST WE'LL SUBTRACT A DOZEN MORTGAGE PAYMENTS. THAT'D BE ABOUT 1/4 ...
SHWACK!
47 + 1/3


AND FOOD...THAT'S ABOUT 1/4 ... AND 1/7 FOR CLOTHES... 1/12 FOR GASOLINE... 1/10 FOR TYPEWRITER RIBBONS... AND A WHOPPING 3/10 FOR ALIMONY PAYMENTS TO AN UNGRATEFUL EX-WIFE!!
WHACK! WHACK! WHACK!


MR. JONES, PLEASE TELL US WHAT FRACTION REMAINS TO PUT TOWARD AN EARLY AND WELL-DESERVED RETIREMENT.


DIDDLY-SQUAT
NOW HERE'S THE SALARY OF A CITY GARBAGE COLLECTOR...

Doubles

Integers are fine so long as none of the values involve decimals. The accuracy problem that you would have run into with division in the previous lesson is that, unless the remainder was a even number, you would have no way of representing a decimal with an integer variable.

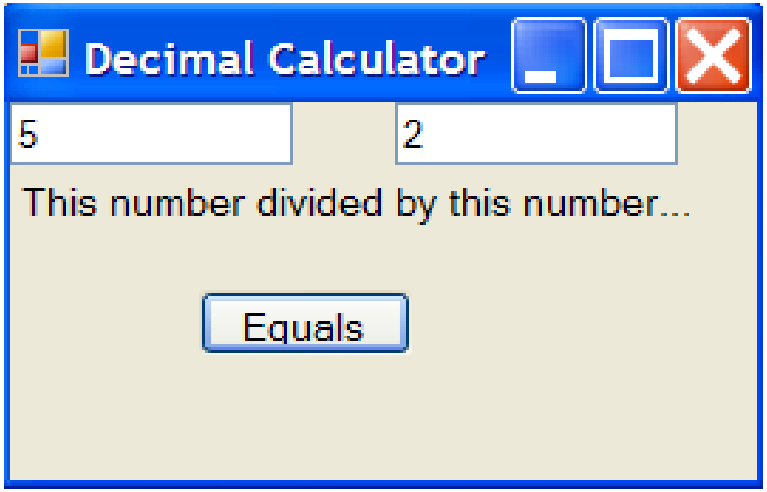
Doubles variables are able to represent decimal values between… well, lets just say some pretty big numbers.

To declare a variable as a double….

```
Dim numOne As Double
```

Decimal Calculator

```
Private Sub btnEquals_Click(ByVal sender As
    Dim numOne As Double
    Dim numTwo As Double
    Dim answer As Double
    numOne = Val(txtNum1.Text)
    numTwo = Val(txtNum2.Text)
    answer = numOne / numTwo
    MsgBox("Answer is " & answer)

End Sub
```



Getting GUI

The buttons and forms that a user interacts with when using a program is calld the Graphical User Interface or GUI (pronounced gooey)

A good GUI provides many hints as to how a program is used just by the way the various components are laid out.

Many GUI’s that are used in software mimic the design used in everyday life.

The modern day calculator is an example of good GUI design .

Principles of Good GUI Design

Uncluttered layout

Well sized controls

Logical order (left to right, up to down order)

Symbols where appropriate

Trip Calculator

miles a unit of length equal to 1760 yards

kilometers One kilometer is equivalent to 1,000 meters or 0.62 miles

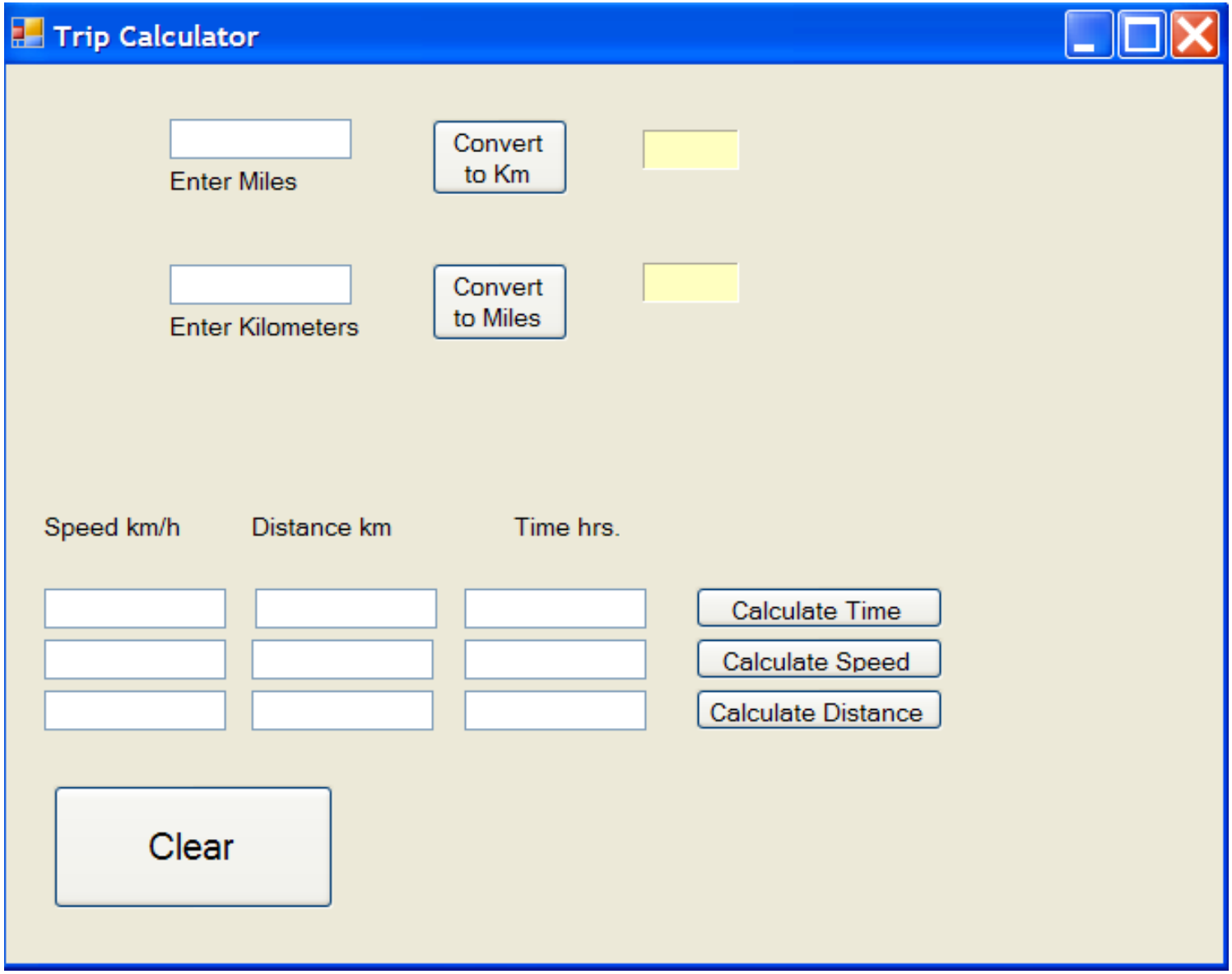
In this project we will create a program that will allow a user to calculate time, distance and speed. Additionally we will provide a handy miles/kilometers converter.

Since we want decimal accuracy in this calculator we will use Double variables for all calculation.

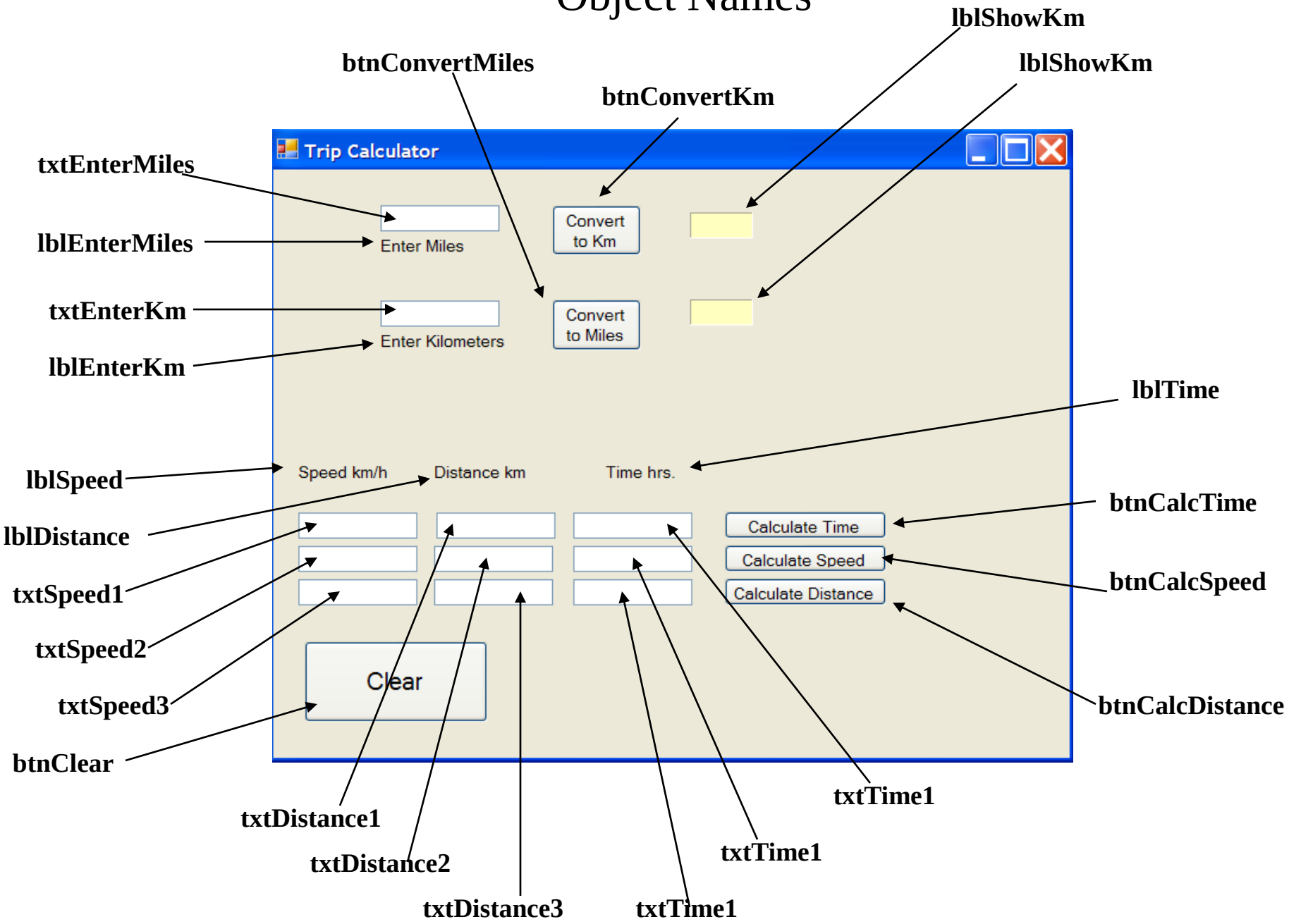
Generally information is entered left to right, up to down reflecting the style in which we read and write.

We will label all input boxes and provide clear buttons.

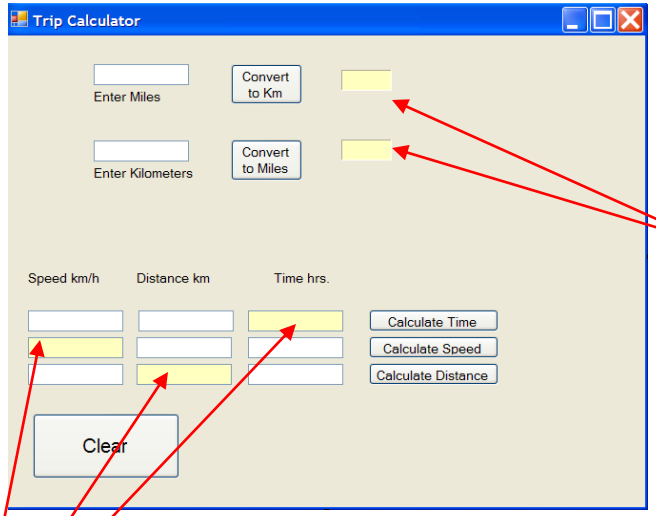
The GUI


Trip Calculator
Enter Miles
Convert to Km
Enter Kilometers
Convert to Miles
Speed km/h
Distance km
Time hrs.
Calculate Time
Calculate Speed
Calculate Distance
Clear

Object Names



ReadOnly Property



Often, if a component is meant to display info rather than allow the user to enter info, a programmer will use a label instead of a text box.

This is the case in the mile/kilometer converter section where the results are displayed.

In the distance/time/speed calculator section text boxes are used for the both the input and the display.

To denote a display text box, the back colour has been set to a pale yellow.

Additionally, the ReadOnly property of each of the 3 display text boxes has been set to True.

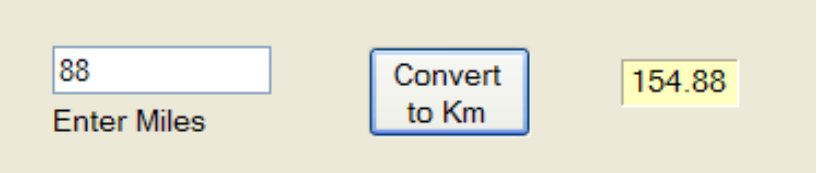
This means that the user will not be able to add or modify any text in these 3 boxes.

Pale Yellow background and ReadOnly property set to true, guides the user to understanding the function of the program.

Matching Variable Names With The TextBox Names

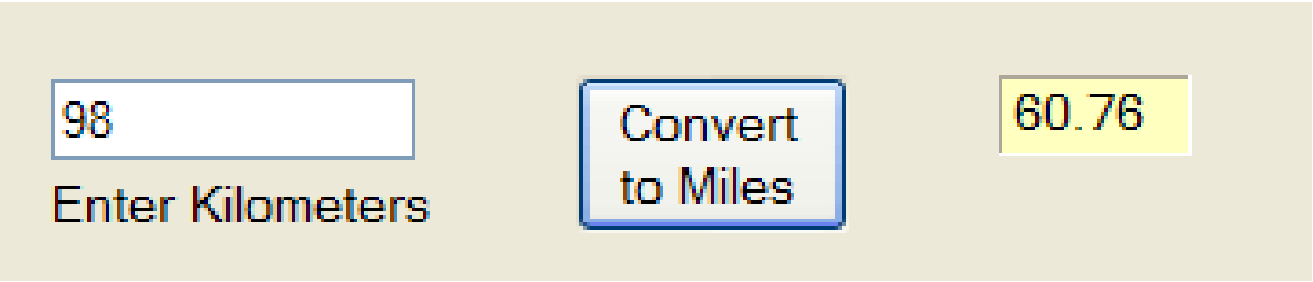
Now we have the task of declaring variables to hold the information which the user will be entering. Here is another example of where conventions can help make reading the code easier.
For each variable use the name of the text box that the information is coming from.
For instance, the text box txtEnterMiles will contain information that should be assigned to a Double variable called enterMiles.
txtEnterKm could send its information to a Double variable called enterKm and so on.

```
Private Sub btnConvertKm_Click(ByVal sender
    Dim enterMiles As Double
    Dim km As Double

    enterMiles = Val(txtEnterMiles.Text)
    km = enterMiles * 1.76
    lblShowKm.Text = km
End Sub
```



ConvertMiles Code

```
Private Sub btnConertMiles_Click(ByVal
    Dim enterKm As Double
    Dim miles As Double

    enterKm = Val(txtEnterKm.Text)
    miles = enterKm * 0.62
    lblShowMiles.Text = miles
End Sub
```



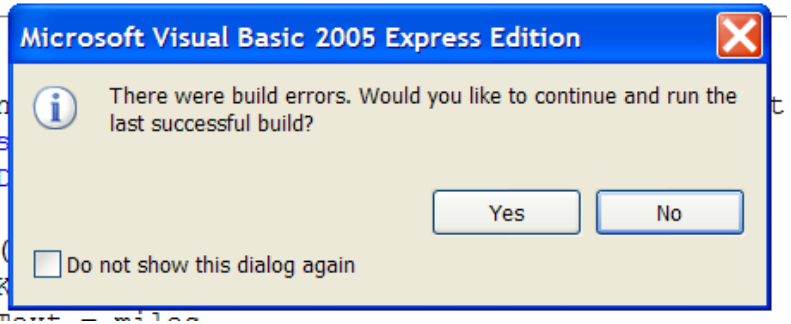
Syntax Errors

There are two types of errors in programming. Syntax and Logic.

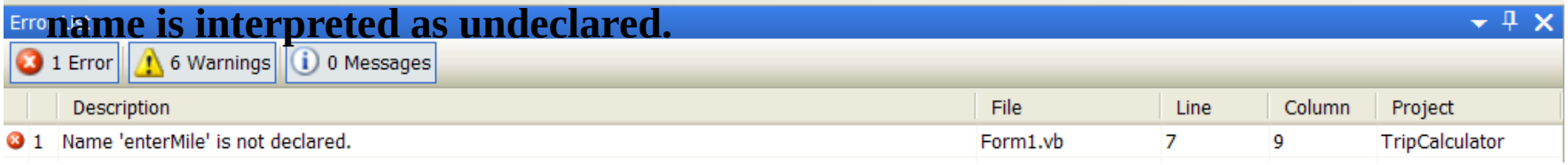
Syntax Errors – errors that prevent the compiler from running.
These are easy to spot. The compiler displays error messages and won't allow the program to continue until they are fixed.

Simply forgetting the 's' on the variable enterMiles causes an error message to show up.
Errors show up on the fly as underlines even before the project is run.

```
    enterMile = Val(txtEnterMiles.Text)
    km = enterMiles * 1.76
    lblShowKm.Text = km
End Sub

Private Sub btnCon
    Dim enterKm As
    Dim miles As D

    enterKm = Val(
    miles = enterK
```



Read the error messages carefully to help fix problems. Sometimes the message needs a little deciphering such as below when a misspelled variable name is interpreted as undeclared.



	Description	File	Line	Column	Project
1	Name 'enterMile' is not declared.	Form1.vb	7	9	TripCalculator

Logic Errors

Logic errors occur when the program seems to run okay but the data that is returned from the program is flawed or different from the intent of the programmer.

Using the wrong formula in a program, dropping decimal places, forgetting brackets in compound statements are all examples of logic errors.

Logic errors can be very tricky to find especially if the output is close to what is desired.

The key to avoiding logic errors is to test with ***known values.***

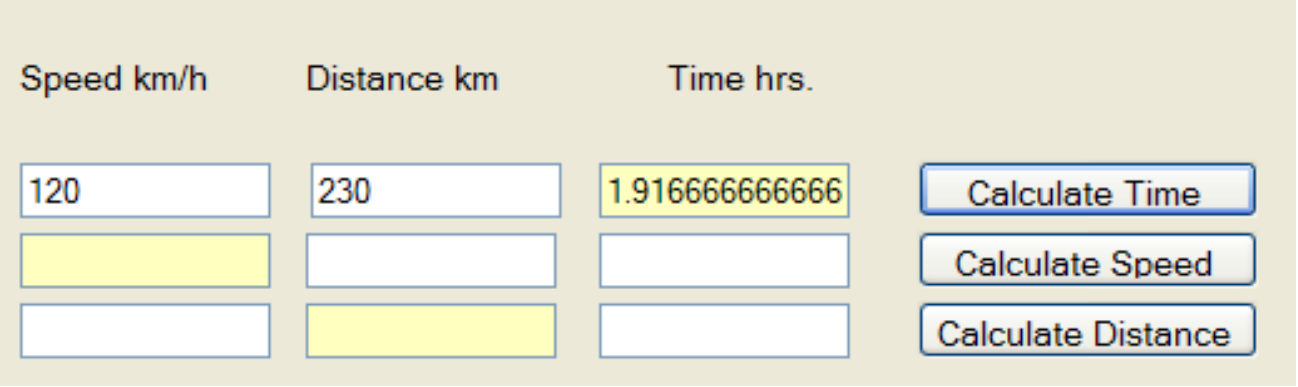
For instance, we know that the formula for converting km to miles is:

miles = km * .62.

Therefore, if we run our program with an input of 100 in txtEnterKm we should expect to see 62 show up in the lblShowMiles label box.

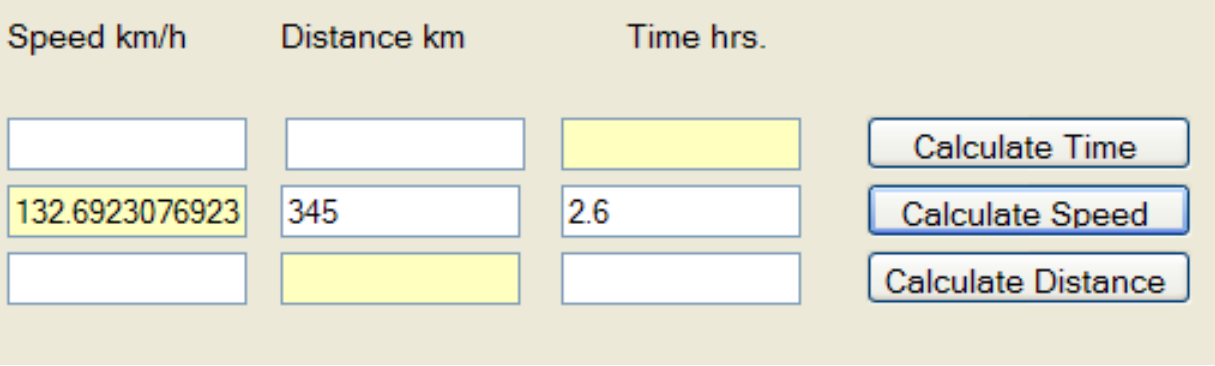
btnCalculateTime

```
Private Sub btnCalculateTime_Click(ByVal sender
    Dim speed1 As Double
    Dim distance1 As Double
    Dim time1 As Double

    speed1 = Val(txtSpeed1.Text)
    distance1 = Val(txtDistance1.Text)
    time1 = distance1 / speed1
    txtTime1.Text = time1
End Sub
```



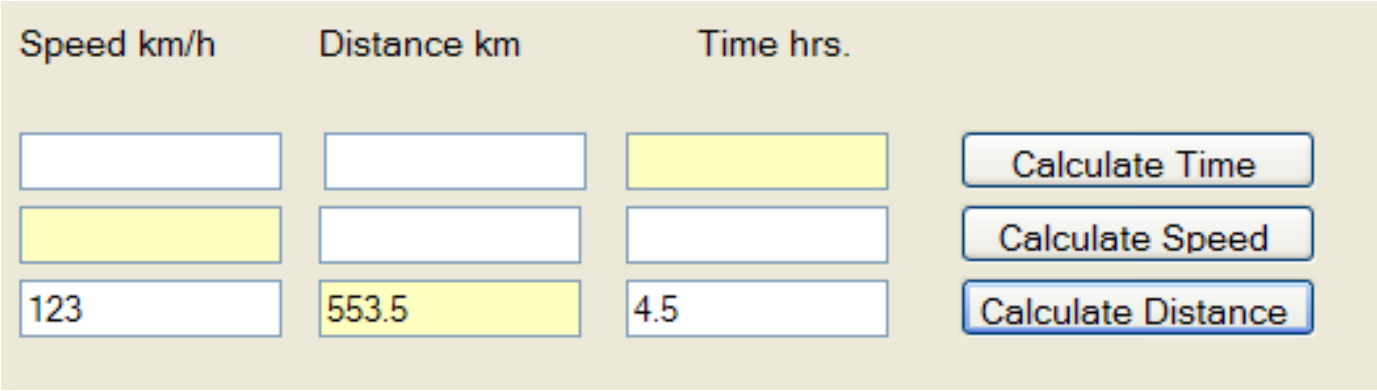
btnCalculateSpeed

```
Private Sub btnCalculateSpeed_Click(ByVal
    Dim speed2 As Double
    Dim distance2 As Double
    Dim time2 As Double

    distance2 = Val(txtDistance2.Text)
    time2 = Val(txtTime2.Text)
    speed2 = distance2 / time2
    txtSpeed2.Text = speed2
End Sub
```



btnCalculateDistance

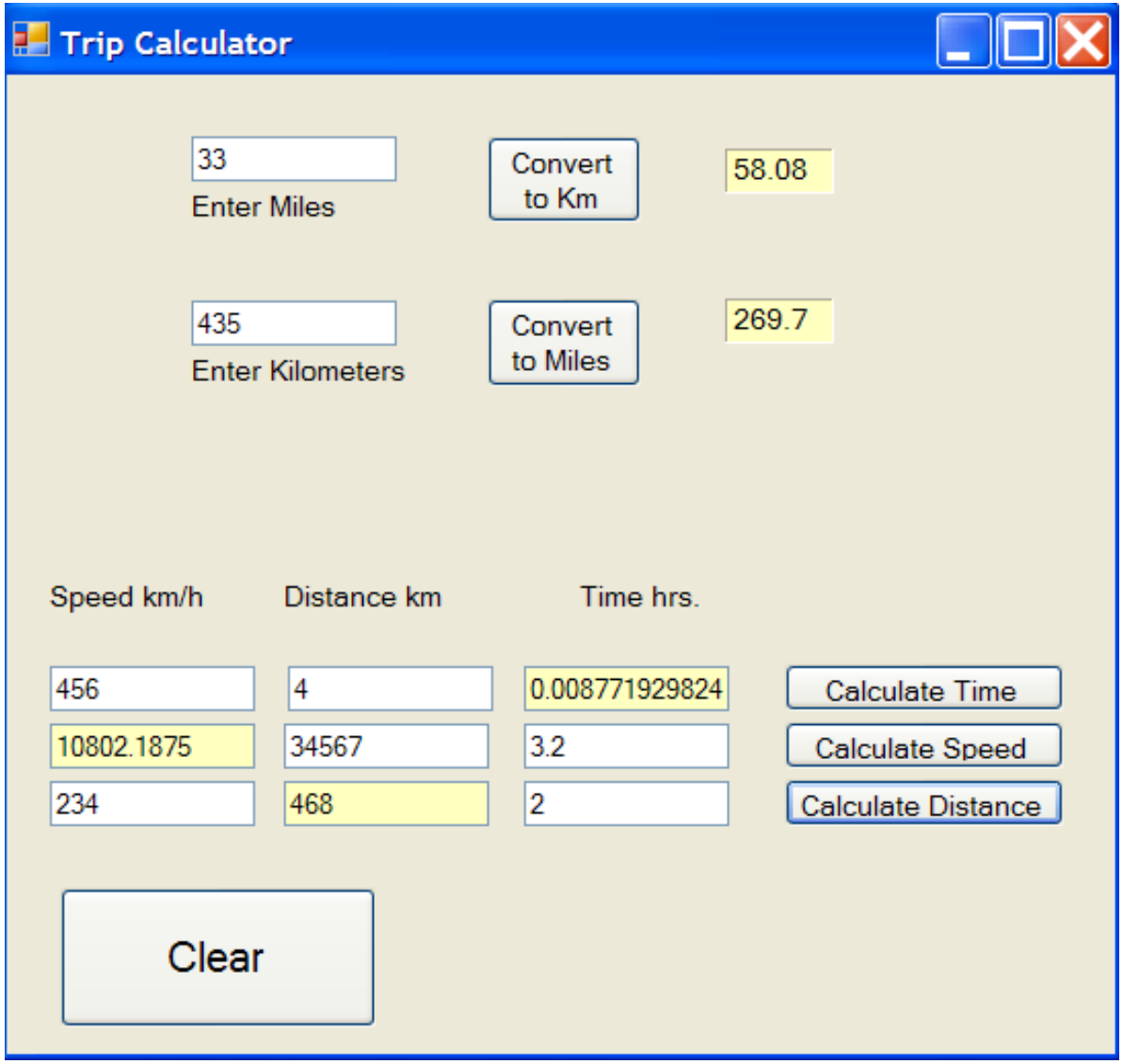
```
Private Sub btnCalculateDistance_Click(ByVal
    Dim speed3 As Double
    Dim distance3 As Double
    Dim time3 As Double

    time3 = Val(txtTime3.Text)
    speed3 = Val(txtSpeed3.Text)
    distance3 = time3 * speed3
    txtDistance3.Text = distance3
End Sub
```



btnClear

```
Private Sub btnClear_Click(ByVal
    txtEnterMiles.Text = ""
    txtEnterKm.Text = ""
    txtSpeed1.Text = ""
    txtSpeed2.Text = ""
    txtSpeed3.Text = ""
    txtDistance1.Text = ""
    txtDistance2.Text = ""
    txtDistance3.Text = ""
    txtTime1.Text = ""
    txtTime2.Text = ""
    txtTime3.Text = ""
    lblShowKm.Text = ""
    lblShowMiles.Text = ""
End Sub
```

Finished Trip Calculator

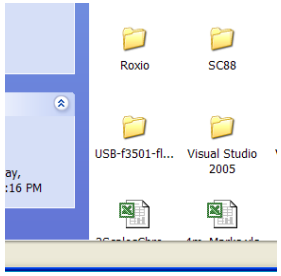


Have Project…will travel

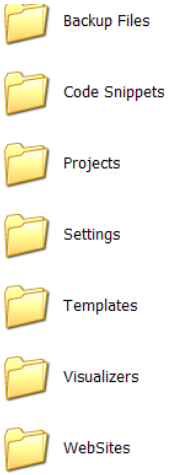
In order to save and take your project off the computer it was created on, follows these steps:

Understand that to open up the project on another computer, that computer will also have to have VB.Net installed on it. (preferably the same release)

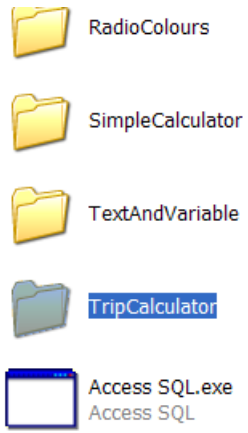
1. Find your Visual Basic Folder



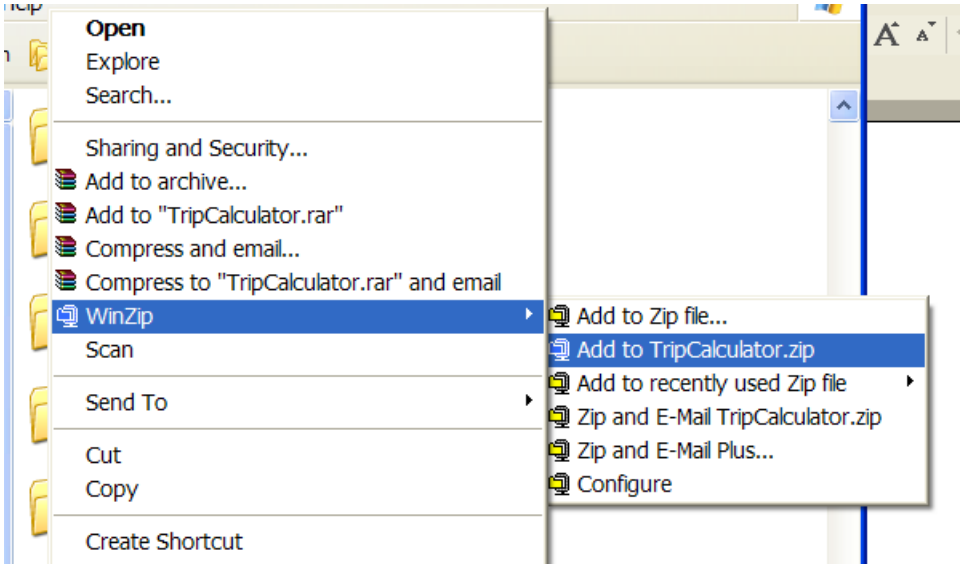
2. Go to the Projects Folder.



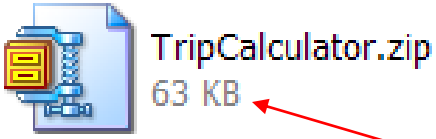
Zipping and Sending



When you discover the project folder rght-click and zip the entire contents of the project folder. Don't try and go into the project folder and send individual files…you will end up frustrated and possibly in tears.



Email and Unzip



A zipped file can be easily sent as an Email attachment.

To unzip the file at its destination simply right-click the zipped file and choose “Extract to here”