**COMPUTER SCIENCE 110 OUTLINE, 2012-2013 K. MACDONALD**

This course serves as an introductory course in computer programming using Visual Basic Express 2010 that should prepare students to take further programming courses in high school, university, or community college and to make an informed career choice in this area. Students will learn to create simple Windows programs.

# ***Outline***

Topics

Visual Basic Interface
Controls
Toolbox
Properties
Flow Chart
Variables
Data Types
Conditional Statements
Multiple Forms
Writing Expressions
Debugging
Documentation
Animated Objects
Sound

**EVALUATION**
Tests
3 Minor Projects
Major Project during last 3 weeks

**Final Mark**

| | |
|---|---|
| Average of Tests | 20% |
| Minor Projects | 30% |
| Final Project | 25% |
| Class Mark | 10% |
| Daily Assignments | 15% |

# Visual Basic

**Visual Basic** is used to create applications for Microsoft Windows. It includes tools that allow a programmer to create an application that has features similar to other Windows applications without having to write many lines of code.

Visual Basic is based on the BASIC programming language developed in the 1960s by John Kemeny and Thomas Kurtz at Dartmouth University. BASIC stands for Beginner’s All-Purpose Symbolic Instruction Code and was used by Kemeny and Kurtz to teach programming to their students.

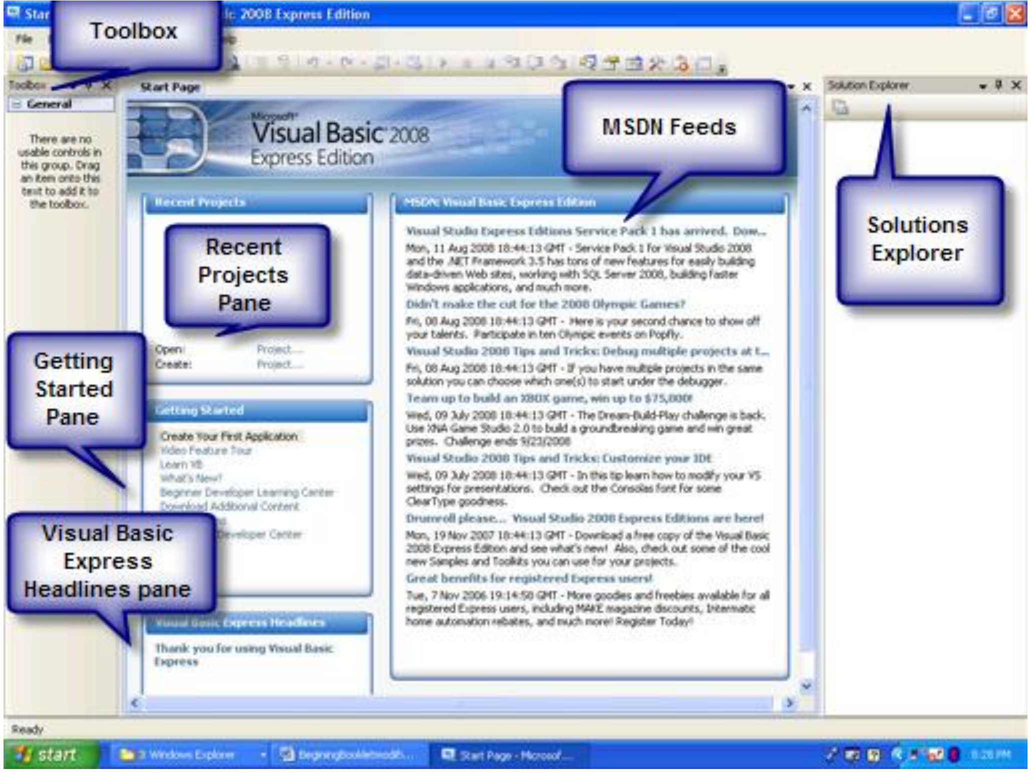
In 1975, Bill Gates and Paul Allen developed a version of BASIC especially for the Altair personal computer. With the success of this new version of BASIC, Gates and Allen founded Microsoft Corporation. BASIC then evolved to QuickBasic, a structured language that made programming easier for the rapidly growing number of personal computer users. In 1985, the Windows GUI (Graphical User Interface) was introduced. In 1992, Microsoft used QuickBasic and a program called Ruby to develop Visual Basic, an ***object-oriented programming*** environment for creating Windows programs.

Visual Basic programs are **event-driven**. An *event* is a way in which the user can interact with an object, such as using the mouse to click on a button in a dialog box. An ***event-driven program*** waits for an event to occur before executing any code, then only code for the current event is executed.

# Getting Started

1. **Start Visual Basic** by clicking on the **Microsoft Visual Basic Express Edition on the Program menu or** **Shortcut**.

2. The next page to show up is the Start Page.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Recent Projects

Open: Project...
Create: Project...

Getting Started

Create Your First Application
Video Feature Tour
Learn VB
What's New?
Beginner Developer Learning Center
Download Additional Content

Visual Basic Express Headlines

Thank you for using Visual Basic Express

MSDN: Visual Basic Express Edition

Visual Studio Express Editions Service Pack 1 has arrived. Dow...
Mon, 11 Aug 2008 18:44:13 GMT - Service Pack 1 for Visual Studio 2008 and the .NET Framework 3.5 has tons of new features for easily building data-driven Web sites, working with SQL Server 2008, building faster Windows applications, and much more.

Didn't make the cut for the 2008 Olympic Games?
Fri, 08 Aug 2008 18:44:13 GMT - Here is your second chance to show off your talents. Participate in ten Olympic events on Popfly.

Visual Studio 2008 Tips and Tricks: Debug multiple projects at t...
Fri, 08 Aug 2008 18:44:13 GMT - If you have multiple projects in the same solution you can choose which one(s) to start under the debugger.

Team up to build an XBOX game, win up to $75,000!
Wed, 09 July 2008 18:44:13 GMT - The Dream-Build-Play challenge is back. Use XNA Game Studio 2.0 to build a groundbreaking game and win great prizes. Challenge ends 9/23/2008

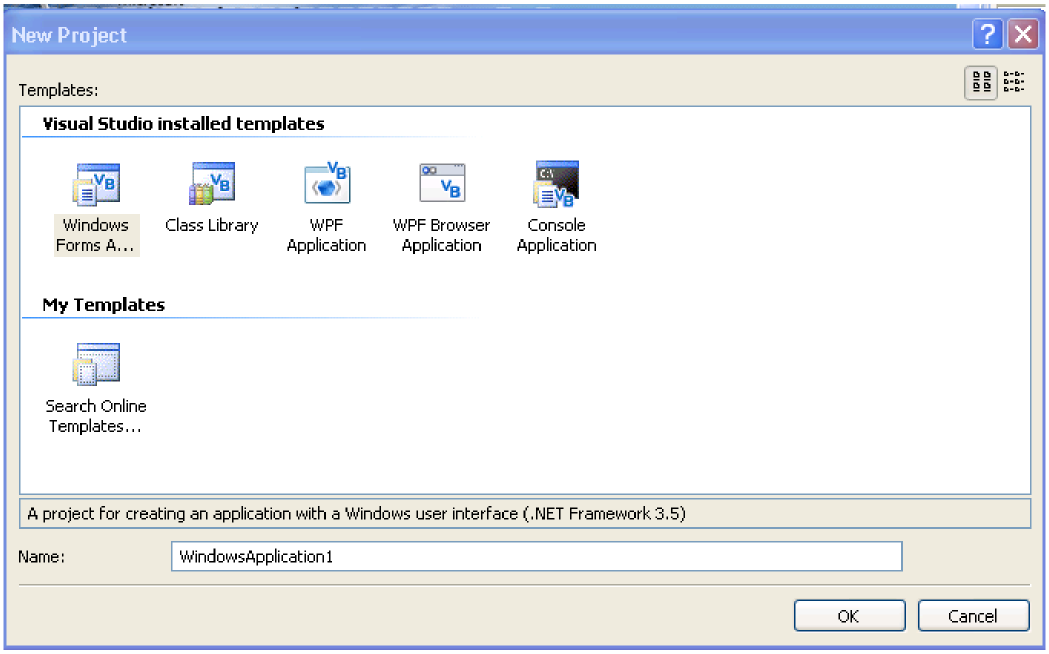
Visual Studio 2008 Tips and Tricks: Customize your IDE
Wed, 09 July 2008 18:44:13 GMT - In this tip learn how to modify your VS settings for presentations. Check out the Consolas font for some ClearType goodness.

Drumroll please... Visual Studio 2008 Express Editions are here!
Mon, 19 Nov 2007 18:44:13 GMT - Download a free copy of the Visual Basic 2008 Express Edition and see what's new! Also, check out some of the cool new Samples and Toolkits you can use for your projects.

Great benefits for registered Express users!
Tue, 7 Nov 2006 19:14:58 GMT - More goodies and freebies available for all registered Express users, including MAKE magazine discounts, Intermatic home automation rebates, and much more! Register Today!

The start page contains the following information:
**Toolbox:** When expanded the toolbox contains all the controls for creating the applications.
**Recent Projects Pane:** Lists all the recent projects that have been opened.
**Getting Started Pane:** Allows the user to get help with projects by hyper linking to programmers or seeing a list of how to articles.
**Visual Basic Express Headlines Pane:** This is where you receive up to date information about VB from Microsoft if you have a live internet connection.
**MSDN Feeds:** By default this feed will come from the MSDN VB 2008 Express RSS feeds. This feed can be changed through the tools and options page.
**Solutions Explorer:** This features lists the files and components in your project.

3. Once you click on Create new project the following templates screen will appear. We will be using the Windows Form Applications for our projects.



New Project

Templates:

Visual Studio installed templates

Windows Forms A... | Class Library | WPF Application | WPF Browser Application | Console Application

My Templates

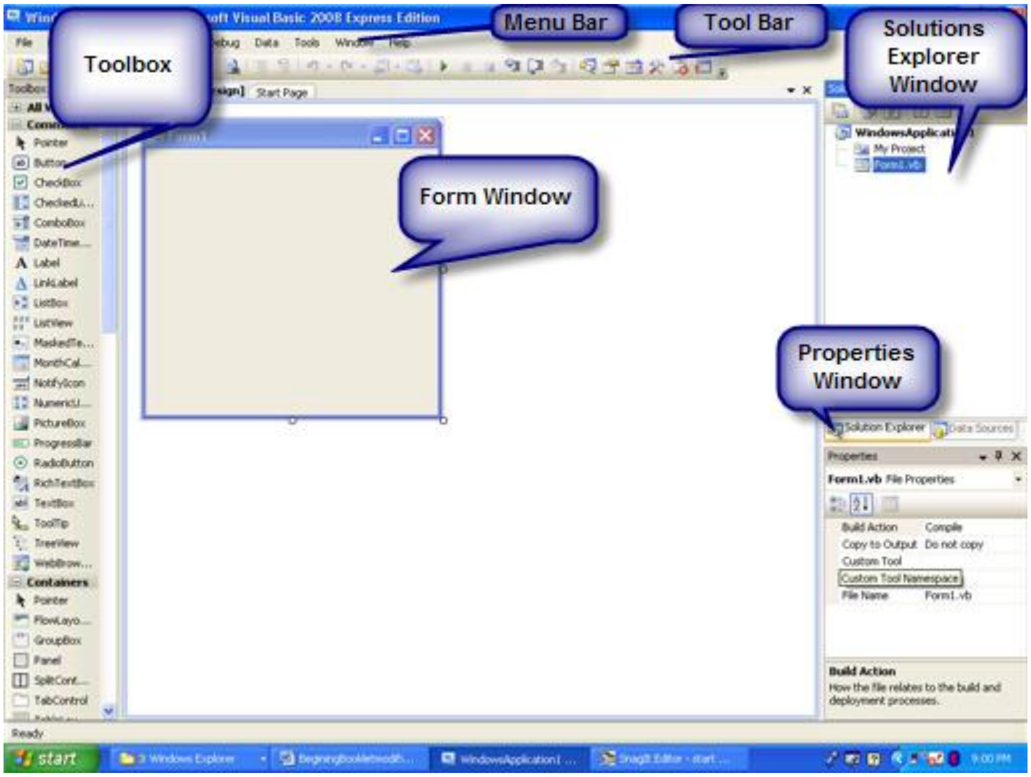
Search Online Templates...

A project for creating an application with a Windows user interface (.NET Framework 3.5)

Name: WindowsApplication1

OK Cancel

4. Your screen should look similar to the picture below. Examine the various parts. If any of them are missing, they can be restored by clicking on View—Then selecting the appropriate part. If the Form Windows does not show, double click on the Form name in the Project Window. Although these Windows can be moved around the screen, it is advisable to leave them in the configuration shown below.



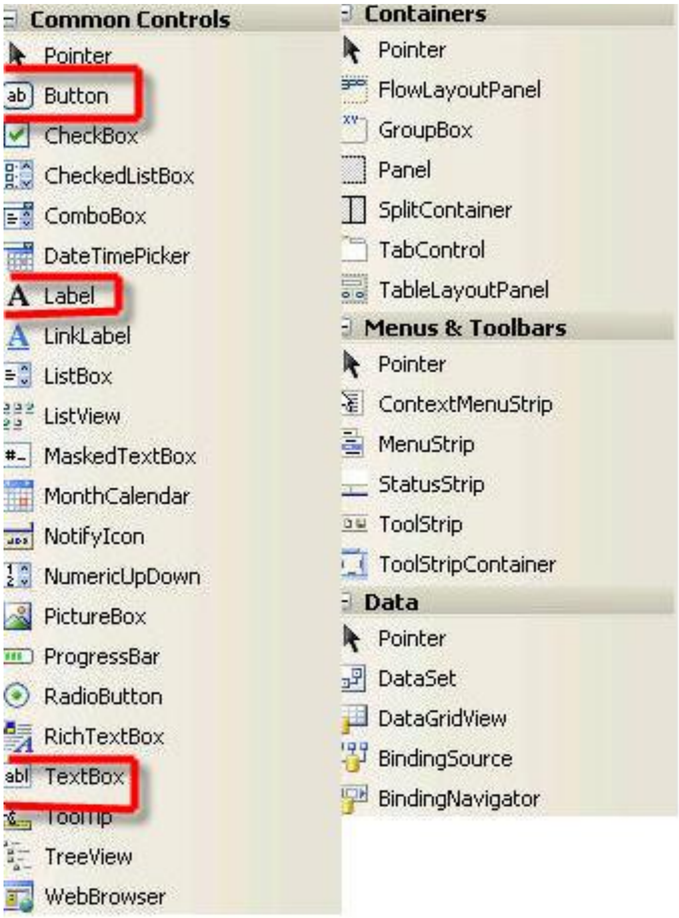
## The Form Window

This is where you design the Windows for your program.  You can add additional forms by clicking on the New Form icon (or selecting New Form from the File menu).

### The Solutions Explorer Windows (The Project Window VB6)

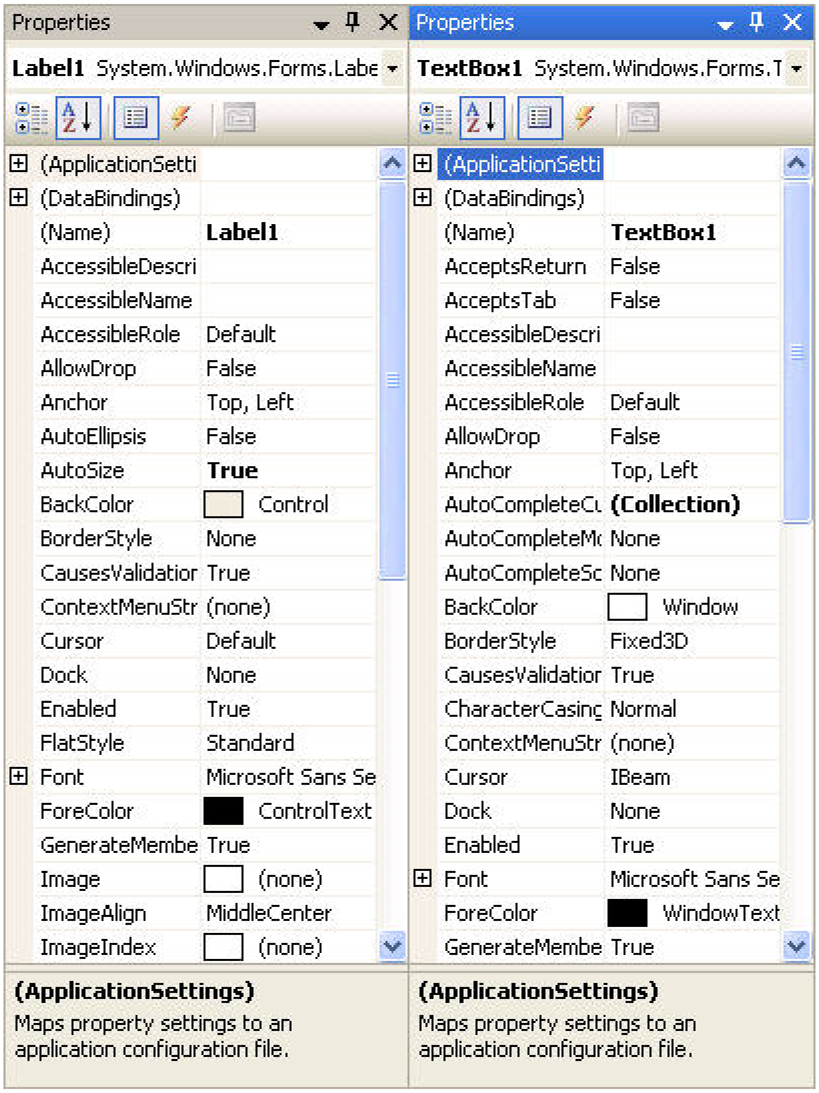
Forms are saved as projects.  If you need only one form in your program, then that project will contain only a single form.  The project window lists the form names used in your project.

**The Toolbox** As you can see there are many different controls that can be used in a VB project. The most commonly used one are boxed in red.



You can select various tools and put them on the form.  This can be done by clicking once on the object and then clicking and dragging across a section on the form, or by double clicking on the tool and then moving it to the point you wish.  By clicking on the object on the form, you will get handles which indicate that that tool is the active one and you can then resize it, move it, or set properties on it.

**<u>The Properties Window</u>** Here is an example of two different objects with some of their properties. Some are the same while others are totally different.



Properties

**Label1** System.Windows.Forms.Labe

| | |
|---|---|
| ⊞ (ApplicationSetti | |
| ⊞ (DataBindings) | |
| (Name) | **Label1** |
| AccessibleDescri | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| Anchor | Top, Left |
| AutoEllipsis | False |
| AutoSize | **True** |
| BackColor | □ Control |
| BorderStyle | None |
| CausesValidatior | True |
| ContextMenuStr | (none) |
| Cursor | Default |
| Dock | None |
| Enabled | True |
| FlatStyle | Standard |
| ⊞ Font | Microsoft Sans Se |
| ForeColor | ■ ControlText |
| GenerateMembe | True |
| Image | □ (none) |
| ImageAlign | MiddleCenter |
| ImageIndex | □ (none) |

**(ApplicationSettings)**
Maps property settings to an application configuration file.

Properties

**TextBox1** System.Windows.Forms.T

| | |
|---|---|
| ⊞ (ApplicationSetti | |
| ⊞ (DataBindings) | |
| (Name) | **TextBox1** |
| AcceptsReturn | False |
| AcceptsTab | False |
| AccessibleDescri | |
| AccessibleName | |
| AccessibleRole | Default |
| AllowDrop | False |
| Anchor | Top, Left |
| AutoCompleteCu | **(Collection)** |
| AutoCompleteMo | None |
| AutoCompleteSc | None |
| BackColor | □ Window |
| BorderStyle | Fixed3D |
| CausesValidatior | True |
| CharacterCasing | Normal |
| ContextMenuStr | (none) |
| Cursor | IBeam |
| Dock | None |
| Enabled | True |
| ⊞ Font | Microsoft Sans Se |
| ForeColor | ■ WindowText |
| GenerateMembe | True |

**(ApplicationSettings)**
Maps property settings to an application configuration file.

One of the characteristics of an object is that it has a “state.” This state is made up of a number of properties. Visual Basic lists the properties of objects in a Properties window. At any one time, the window shows the properties of only one object. First click on the object to select it, then go to the properties window and the properties for that object are displayed. If no object is selected, then the properties for the form are displayed.

## <u>Placing the Objects</u>

***Textboxes*****—**These are boxes that contain text.  A textbox can display numbers, letters, or a mixture of both.  You use textboxes to accept INPUT from people running your program.

***Labels*****—**You use a label object to tell something to people running your program—in other words, to print something.

***Buttons*** **–** The button controls are buttons that users click as they are running your program. When a user clicks on one of these buttons, something (an action) happens.  The caption of a button, such as "Exit" or "Quit," explains what that action will be.

## Naming Conventions

The way you name the controls placed on a form is important because the names are used to refer to the controls in the program code**.**  Meaningful names make understanding the program code easier.

Textbox names start with the prefix "txt".  The prefix for labels is "lbl", for Command (Formerly cmd) Buttons, "btn", and for forms "frm".  The prefix is followed by one or more words describing the function of the control.  No spaces are used.  Each new word is capitalized.  Here are some examples:

txtLastName
lblBirthPlace
txtUnpaidBalance
btnDisplayPicture
frmDisplay

Here is the list of prefixes:

### Prefix Abbreviations for Control Names

| **Prefix** | **Control** |
|---|---|
| cbo | Combo Box |
| chk | Check Box |
| btn | Button |
| dir | Directory list box |
| drv | Drive list box |
| fil | File list box |
| fra | Frame |
| frm | Form |
| grd | Grid |
| hsb | Horizontal scrollbar |
| img | Image |
| lbl | Label |
| lin | Line |

| | |
|---|---|
| lst | List box |
| mnu | Menu |
| ole | OLE client |
| opt | Option button |
| pic | Picture Box |
| shp | Shape |
| tmr | Timer |
| txt | Text box |
| vsb | Vertical scrollbar |

## Setting Properties for Forms,Textboxes, Labels, and Commands

### Labels

**Text**—This is the text that the label will display
**Name**—The name of the label—eg. lblBirthplace—The Name is used to identify the label in code.

### Textboxes

**Name**—The name of the Textbox—eg. txtLastName—used in code
**Text**—The text that you wish to see displayed, if any.  Often this is just deleted as the user will fill in the textbox when the program is run.

### Commands / Buttons

**Text**—Words you wish to see displayed on the command button.
**Name**—the name of the button—eg  btnExit—used in code

### Forms

**Name**—the name of the form—eg. frmTitle—used in code.  This name will appear in the Project Window.  It will also be used as the filename when you save the form.
**Text**—This text will appear in the Title Bar of the form.

## Steps in Creating a Program

This is a **three-step process**.

1. Select the object and arrange on the form.
2. Set the properties of the form and the objects.
3. Write the code.

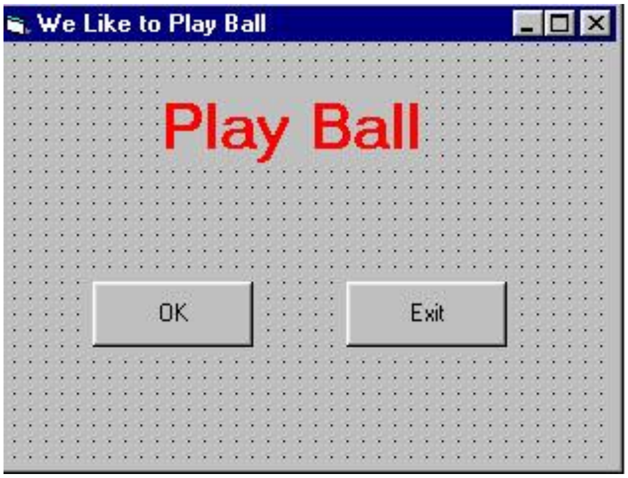
## To Write the Code—Attaching Actions to Objects

Double click on the object to get to the Code Window. Notice that Sub and End Sub are already there for you. Insert your code in between. For example, the code for the Exit button would be **Me.Close**. This command should be indented to make your code easier to read.

Double clicking on the form name in the Project Window will return you to the form. Then just double click on the next object you wish to attach code to.

## Our First Program

Let's write a program to display "**Play Ball**" in large red letters when we hit a command button called **OK**.



***Step 1***—Place a **label** on the form, as shown, large enough to hold the words **Play Ball**. Place 2 **buttons** on the form as shown.

***Step 2***--Set the properties for the form and the two objects as follows:

**Form**
Set **text** to **We Like to Play Ball**
Set **Name** to **frmPlay**

**Label**
Set **text** to **Play Ball**
Set **name** to **lblPlayBall**
Set the **font** to **MS Sans Serif Bold 24 pt**
Set **forecolor** to **red**
Set **visible** to **false**
Set **Autosize** to **true**

**1st Button**
Set **name** to **btnOK**
Set **text** to **OK**

**2nd Button** Set **name** to **btnExit**
Set **text** to **Exit**

***Step 3***—Write the code. **Double click** on the **Exit** command. Tab to indent, and insert this code between the sub and end sub commands:

```
Me.close
```

**Close** the code window to return to the form, and **double click** on the **OK** command. Enter this line of code between the sub and end sub commands.

```
lblPlayBall.Visible = True
```

**Run your program. Test the OK and Exit buttons.**

**Saving a Project**—Select **File, Save Project As**

**Select your own folder on the fileserver with your login name on it. For each Visual Program you wish to save, create a new folder within your folder in which to store all the files associated with your project. This is necessary because most programs consist of several files and it is important to keep them together.**

You will first be asked to save the form—Yes—The default filename should be frmPlay.—Make sure to put it in a new folder within **your folder**.
Then you will save the project—Yes—Name it **PlayBall** and put it in **the same folder**.

## Using Variables and Making Calculations

### Using Variables

Visual Basic programs are designed to deal with a large amount of data stored in the computer. This data is not entered by you the programmer. Instead, it comes from:

- Users entering information
- Data gathered as the program runs

A good variable name should reflect something about the information it represents. There are some rules to follow as you create variable names. These names may be up to forty characters (letters or numbers) long. You should also follow the Intercap method to name variables, using capital letters to show how words or portions of words are put together.

**Eg.** *SpeedOfTrain*
*LastName*
*DateOfBirth*

## Variable Declaration

This means you will have to declare any variables before you use them in the appropriate sub program. When you declare a variable you are telling the program the name of the variable and the type of data it will hold.

**Eg.** *Dim LastName as String*
*Dim Amount as Currency*
*Dim Quantity as Integer*
*Dim Degrees as Single, Frequency as Double*

## Types of Data

In the beginning, we will use three types of data. If you declare a variable incorrectly, you will get an error message when you try to run the program.

***String*** This type of variable is used to store words. More correctly, it could be defined as anything inside quotes.
Eg. “Miramichi” or “******” are both strings.

***Single*** This type of variable is used to store numbers, including decimal places, to an accuracy of about 8 digits.

***Integer*** This type is used to store whole numbers only, in the range –32,768 to +32,767

## The Other Visual Basic Data Types

| Datatype | Description and Range |
|---|---|
| Boolean | A Datatype that takes on one of two values only: True or False |

| | |
|---|---|
| Byte | Positive numeric values without decimals that range from 0 to 255. |
| Currency | Data that holds dollar amounts from -$922,337,203,685,477.5808 to 922,377,203,685,477.5807. The four decimal places ensure that proper rounding can occur. |
| Date | Holds date and time values. The date can range from January 1, 100, to December 31, 9999. |
| Decimal | A new Datatype not yet supported in Visual Basic except in a few advanced situations. The **Decimal** Datatype represents numbers with 28 decimal places of accuracy. |
| Double | Numeric values that range from -1.79769313486232E+308 to 1.7979769313486232E+308. The Double Datatype is often known as double-precision. |
| Integer | Numeric values with no decimal point or fraction that range from -32, 768 to 32,767. |
| Long | Integer values with a range beyond that of Integer data values. Long data values range from -2,147,483,648 to 2,147,483,647. Long data values consume more memory storage than integer values, and they are less efficient. |
| Object | A special Datatype that holds and references objects such as controls and forms. |
| Single | Numeric values that range from -3.402823E+38 to 3.402823E+38. The Single Datatype is often called single-precision. |
| String | Data that consists of 0 to 65, 400 characters of alphanumeric data. Alphanumeric means that the data can be both alphanumeric and numeric. String data values may also contain special characters such as ^, %, and @. Both fixed length strings and variable – length strings exist. |
| Variant | Data of any Datatype, used for control and other values for which the Datatype is unknown. |

Prefixes to Maintain Accurate Datatype

The data types highlighted are the one we will be using:

| **Prefix** | **Datatype** | **Example** |
|---|---|---|
| bln | Boolean | blnIsOverTime |
| byt | Byte | bytAge |
| cur | Currency | curHourlyPay |
| dte | Date | dteFirstBegan |
| dbl | Double | dblMicroMeasurement |
| int | Integer | intCount |
| lng | Long | lngStarDistance |
| obj | Object | objSoundClip |
| sng | Single | sngYearSales |
| str | String | strLastName |
| vnt or var | Variant | vntControlValue |

## Making Calculations

**<u>Assignment Statements</u>**

A value is assigned to a variable. Only the variable being assigned the value can appear on the left side of =. the calculations (expressions) appear on the right side.

**Eg.**

Interest = Principal * Rate * Time
AvgSpeedPerHour = DistanceTravelled / Hours
TestAverage = (Test1 + Test2 + Test3)/3
Area = 3.14 * Radius ^ 2

**Order of Operations**

In writing assignment statements, it is important to understand the order of operations (the order in which calculations will be performed).

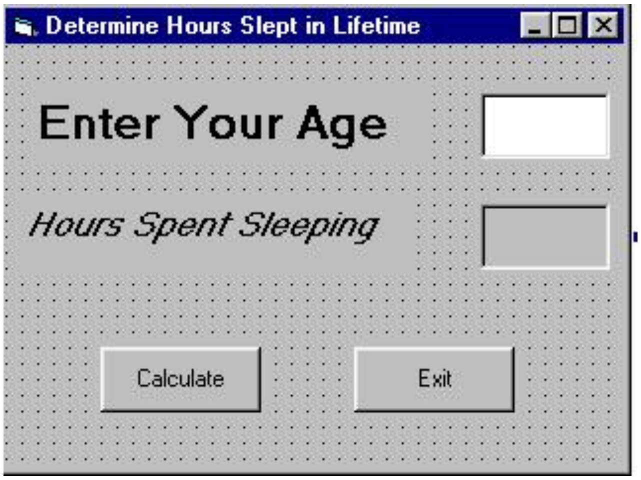
| | | | |
|---|---|---|---|
| 1. | Whatever is inside brackets | ( ) | |
| 2. | Exponents | 5 ^2 would be 5 squared | |
| 3. | Multiplication and Division | * | / |
| 4 | Addition and Subtraction | + | - |

Go to **http://msdn.microsoft.com/en-us/library/fw84t893(VS.80).aspx**
for the complete list of operators.

## A Program with Calculations

**This Program will Calculate the TotalHours you have slept in a Lifetime—assuming 8 hours per day**

**First, design a form similar to the one below**. You should have 3 labels (for the words written and to display the answer), 1 textbox (to enter your age), and 2 command buttons.



**Next, Set the Properties as shown in the table below:**

| | | |
|---|---|---|
| Form | Name<br>Text | frmAge<br>Determine Hours Slept in Lifetime |

| | | |
|---|---|---|
| Label1 | Text<br>Font | Enter Your Age<br>18 |
| Textbox | Name<br>Text | txtAge<br>(delete Text1) |
| Label2 | Text<br>Font | Hours spent Sleeping<br>14 |
| Label3 | Name<br>Text<br>Border Style | lblTotalHours<br>(delete the Text)<br>1—Fixed Single |
| Command1 | Name<br>Text | btnCalculate<br>Calculate |
| Command2 | Name<br>Text | btnExit<br>Exit |

**Next, enter the code for the Command Buttons**—Remember, double click on the command button to take you to the code section.

```
Private Sub btnCalculate_Click()
        Dim Age As Single, HoursPerDay As Single
        Dim TotalHours As Single, DaysPerYear As Single
        Age = Val(txtAge.Text)
        HoursPerDay = 8
        DaysPerYear = 365
        TotalHours = Age * HoursPerDay * DaysPerYear
        lblTotalHours.Text = TotalHours
        txtAge.Focus()
  End Sub


Private Sub btnExit_Click()
  Me.close
End Sub
```

## Run the Program

You should now be able to enter your age (or someone else's). If you then click on the Calculate button, it should display the total numbers of hours spent sleeping.

## Add Another Button to the Form

Add a third command button to the form whose purpose is to clear the textbox and label so that you can enter someone else's age. Name this command **btnClear** and set the caption to **Clear**. The code for the command is

```
      txtAge.Clear()
      lblTotalHours.text =String.Empty
      txtAge.Focus()
```

## Save Your Program

Select **Save Project As**. Within your folder, create a new folder called **Age**. First save the form—it should be called **frmAge**. Then save the project—call it **Sleep**.

## **Explanation of Code**

- The Dim statements on the first two lines declare the required variables as numeric.
- *Age = Val(txtAge.Text)* takes the Text property from the textbox, txtAge, and assigns it to the variable *Age*. *Val* converts the text property (a text property is a string value) to a numeric value. Think of Val for Value.
- The next two lines just assign values to the variables.
- *TotalHours = Age * HoursPerDay * DaysPerYear* is an assignment statement which performs the calculations on the right side of the equal sign and then assigns that value to the variable *TotalHours*
- *lblTotalHours.Caption = TotalHours* sets the Text property of *lblTotalHours* to the value of the variable *TotalHours*

## **Setting Focus**

Focus is a procedure which will be carried out as the last command of the Clear and Calculate buttons. This means that after you hit the Clear button, control will be transferred to the textbox txtAge, ready for you to enter another value. After you hit the calculate button, control will be passed to the Clear button. You can then just hit enter to clear the values.