CHAPTER 7 APPLICATION PROGRAMMIN USING VB & VB.NET

Figure 7.1. Application made by Visual Basic.

Examine Figure 7.1, it an application to show a table of collection of data. Someday you may find such application. The above application may be quickly made by using Visual Basic, without any difficulties in handling table, menu, button etc. All we need to do is click, drag, placed, arrange position then do a little bit of coding, an application may be made.

The standard of competence in programming with VB and basic VB.Net consists of three (3) basic competences. In this book, each basic competence contains the material, and the exercise. The summary is written at the end of each chapter and followed by exercise. In this chapter, the basic competence covers Visual Basic foundations, data access and manipulation in Visually Basic, and applying COM technology. Prior to study this competence, please review the operation system, the principle of problem solving, the programming algorithm as well as supporting materials especially mathematics.

THE OBJECTIVES

After studying this chapter, the reader is hoped to be able to:

- Explain on the foundation of Visual Basic.
- Data access and manipulation using Visual Basic.
- Applying COM technology.

7.1. Foundation of Visual Basic

Visually Basic (VB) is one of the computer's programming. The VB programming language, that was developed by Microsoft since 1991, is the development from its predecessor namely BASIC programming language (Beginner's All-purpose Symbolic Instruction Code) that was developed in the 1950 's. VB is one of the development tool in Windows environment. In developing application, Visual Basic uses Visual approach to design the form of user interface, whereas the coding uses Basic language that is very easy to study. Visually Basic is becoming a famous tool for both beginners and developers. However, the major drawback of VB is in its slow performance as compared to other programming language. Although, using current fast processors and large memory, such problem is no longer important.

Visually Basic is run on Microsoft Windows operation system. To start Visual Basic, we need to select Start -> Programs -> Microsoft Visual Studio 6 -> Microsoft Visual Basic. The starting early Visual Basic will appear like the Picture 7,2.

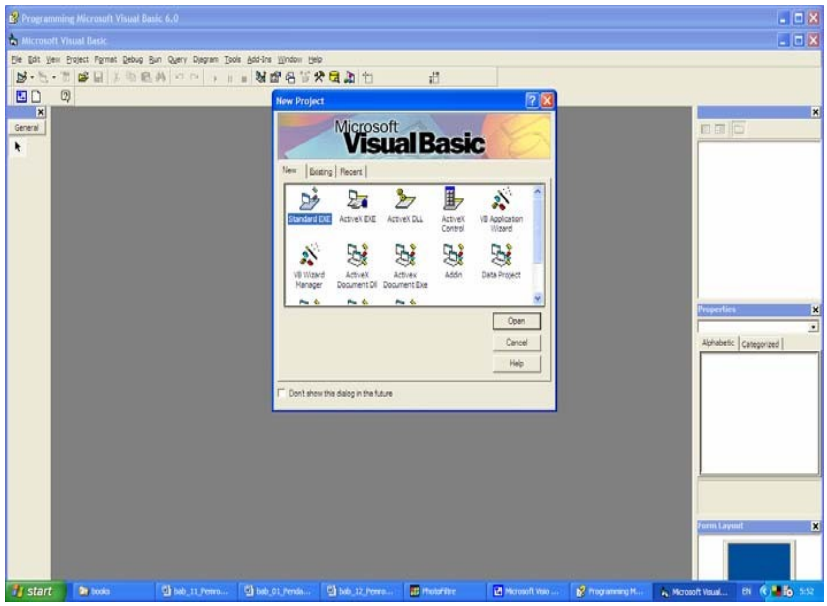


Figure 7.2. Starting Page of Visual Basic.

In Figure 7.2, we are asked to select the type of project that will be built. For a beginner, standard project .EXE is a good choice. After selecting Standard .EXE, we will see the following page.

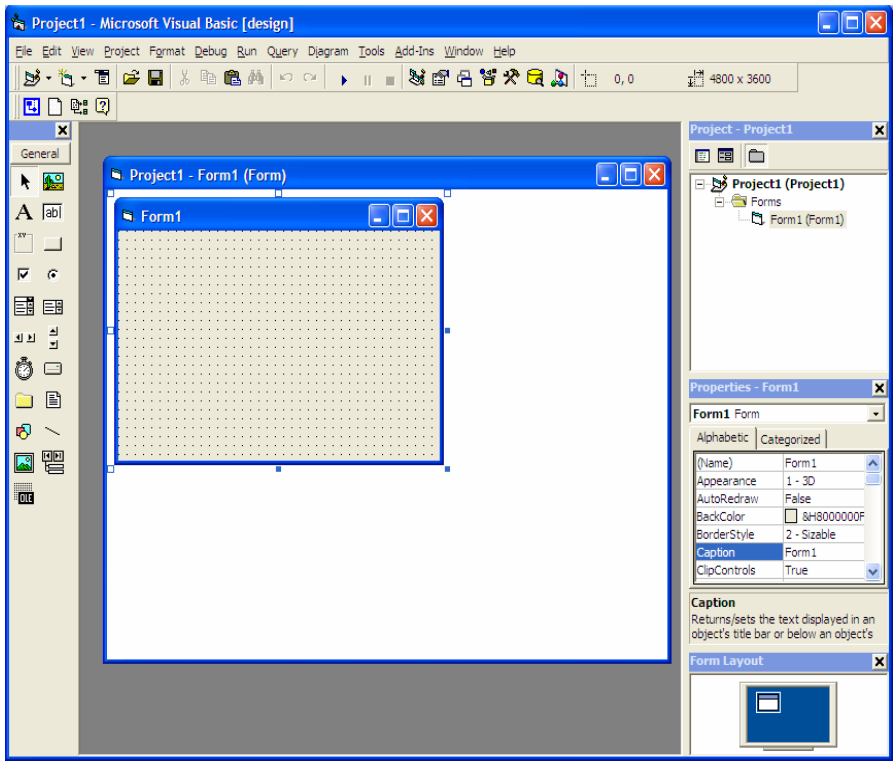


Figure 7.3. Starting page for Standard.EXE.

Prior to work with Visual Basic, it would be a good idea to know the working environment (IDE) of Visual Basic. The Visual Basic IDE (Integrated Development Environment) is the integrated environment within which a programmer is developing their application. Using IDE, a programmer may create interface, do coding, carry out testing and debugging as well as compile the program into executable. A good understanding of IDE will help the programmer to make their job more efficient. Figure 7.4 shows the Visual Basic IDE.

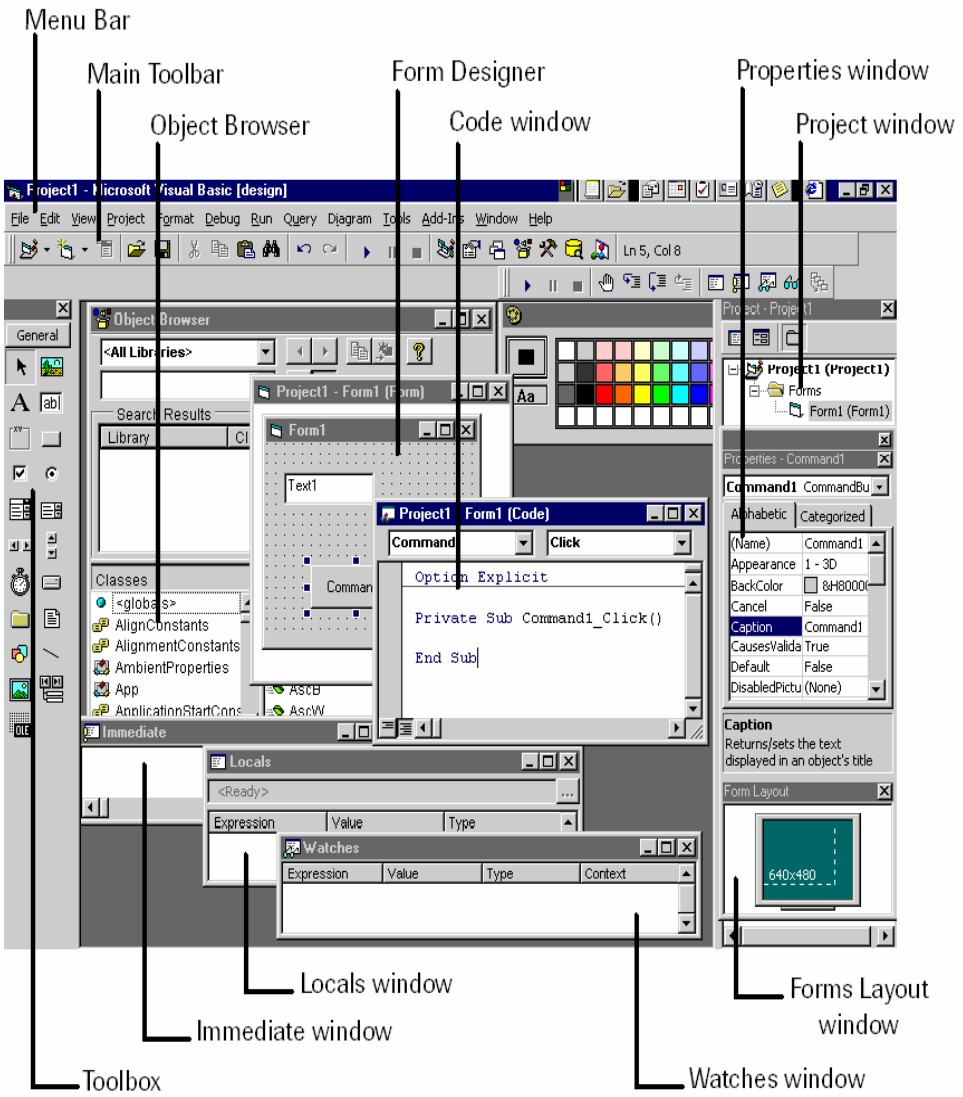


Figure 7.4. Visual Basic IDE

- Menu Bar, used to select certain task such as save project, open project etc.
- Main Toolbar, used quickly invoke certain task.
- Project Window, shows all modules in your application. You may use icon Toggle Folders to show modules in the windows in a group or sort based on

name. You may use Ctrl+R to show project windows, or use icon Project Explorer.

- Form Designer Window is where you may design the user interface of your application. The windows is like canvas for a painter.
- Toolbox Window contains the components that could be used to develop user interface.
- Windows Code is where you write your code. You may active the windows using Shift-F7.
- Properties Window lists object properties that being used. For example, you may change the foreground color and the background color. You may invoke the Properties Windows using F4 button.
- Color Palette window is a short cut facility to change object color.
- Form Layout window shows particular form during run-time.

Toolbox Window is an important windows. From Toolbox Windows, you may select the objects in form to built the user interface.

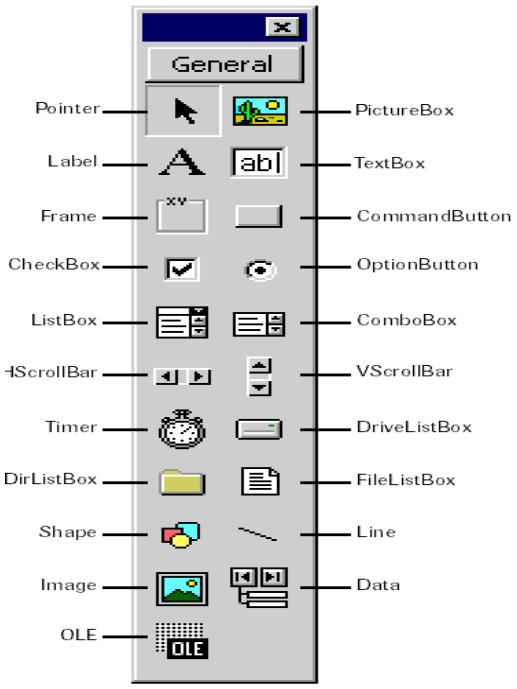


Figure 7.5. Toolbox in VB 6.

- Pointer is not a control tool; use Pointer icon to select existing control in the form.
- PictureBox is a control to show image using format: BMP, DIB (bitmap), ICO (icon), CUR (cursor), WMF (metafile), EMF (enhanced metafile), GIF, and JPEG.
- Label is a control to show fixed texts.
- TextBox is a control that contains string that could be changed by the user. It can be single or multiple lines.
- Frame is a control that used as container for other control.

- CommandButton is a control that can be found in almost every form, and may be used to trigger an certain process event as clicked by the user.
- CheckBox is used to provide selection process, such as, yes / no, true / falses.
- OptionButton is often used to provide options on certain selection.
- ListBox contains user selectable items. More than one item may be selected depending on the property of MultiSelect.
- ComboBox is a combination between TextBox and ListBox where input data may be carried out using typing or selection.
- HScrollBar and VScrollBar is used to create a stand alone scrollbar.
- Timer is to active background process in a specific interval. It is non-visual control.
- DriveListBox, DirListBox, and FileListBox is often used to create dialog box related to files.
- Shape and Line is present form, such as, line, square, circles, oval.
- Image function similar to image box, but could not be used as container for other control. It is interesting to know that control image uses much lower resource than PictureBox.
- Data is used for data binding.
- OLE may be used to link to external programs, such as, Microsoft Excel, Word, etc.

7.1.1. Main GUI based programming principles.

In principle there are two main parts in the application development using VB, that is: visual design and event-driven programming.

Visual Design

In Windows environment, user-interface plays an important role, as the application will have to interact with users via user-interface without realizing that it is supported by instruction program to support the visual interface and the process. In Visual programming, application development starts with user interface creation. To design user interface, the most needed knowledge is only the understanding of type and usage of control and basic knowledge on how to draw an object.

Form and control is the basic user interface elements of Windows based applications. In VB, these elements are known as object as it may be manipulated similar to an object. An object is a combination of code and data that could be treated as one entity. An object has a certain properties and methods, and will react to some external events such as physical objects.

As shown in Figure 7.6, a car is a physical object that has property, method and event. One of its property is color. Car's color property is usually determined prior to its fabrication. If we don't like the color of our car, we can always change it by re-painting the car. Similarly in VB, property of a control is usually determined when object was made (when put in one form), but we may change the property by giving new value. We may change property's value during design process via Properties windows or during run time through program code. Some properties are only available during design time, some available during runtime.

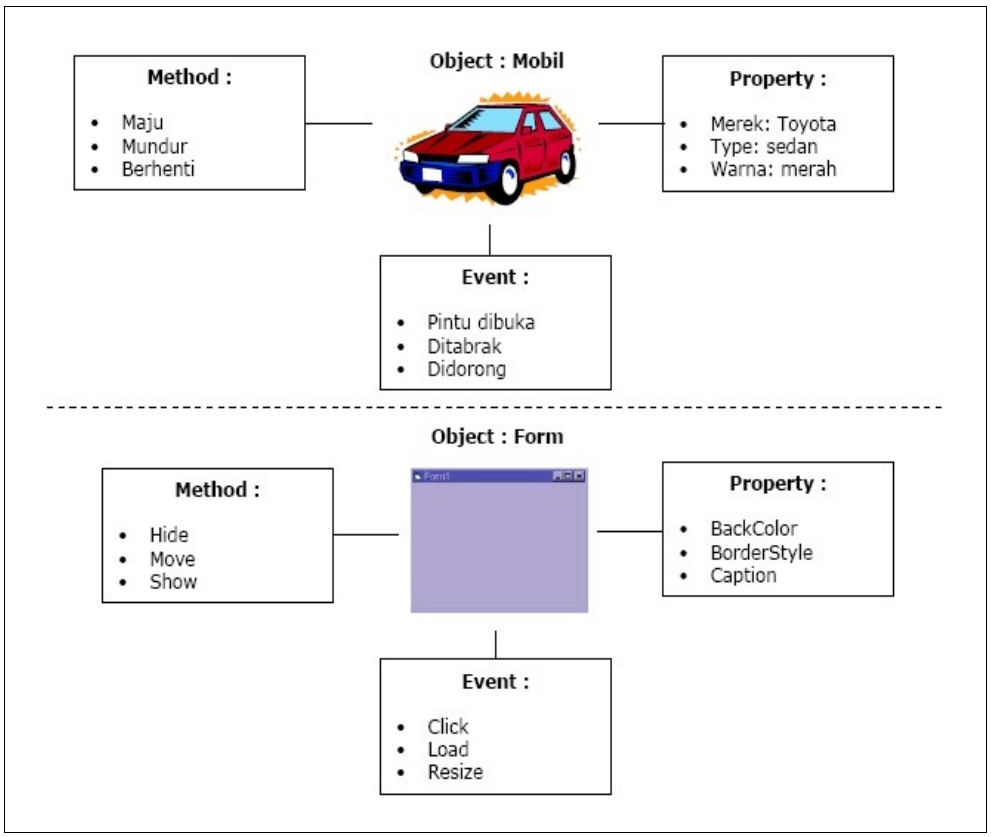


Figure 7.6. Object, Property, Method and Event

Event-Driven Programming

Application programming is not easy, we should follow a methodology. The application program made with VB is not a monolithic program that has only one sequential pass. When we create a program using VB, we must first determine the user interaction of the application. Or in other words, we must define the reaction of every action taken by te user, such as, mouse click, mouse double-click, keyboard button, and so on. This concept is known as Event-Dri ven Programming, because program flow is decided by an event triggered by the user. Our application program must react to the external condition / event, and user action defines the program flow.

7.1.2. Data type, Variable and Constant

In general, data type, variable, and constant in Visual Basic is not different than that described in Chapter 5. Please review Chapter 5 to provide some idea on this topics. The main different is in the declaration syntax. Examine the following example.

Example 7.1. Declaration example of variable, constant, and data type.

```
Dim speed As Double
Dim timeElapsed As Double
Dim NumberStudent as Integer = 10
Dim velocity as Single
Dim Nama as String
Const phi as Single = 3.14
```

In Example 7.1, variable is declared using Dim while constant is declared Const. We can also set the initial value of a variable right after its data type declaration.

One of benefit if using GUI based programming such as Visual Basic is the availability of objects that could be treated as data type. Examine the following example.

Example 7.2. Example object data type usage.

```
Dim   frm As Form
Dim   midfrm As MDIForm
Dim   ctrl As Control
Dim   obj As Object
Dim   inv As frmInvoice
Dim   txtSalary As TextBox
Dim   wrk As Excel.Worksheet
```

In the above Example 7.2, the Form statement, MDIForm, Control and Object are the controls owned by Visual Basic. Whereas frmInvoice is the form made by us the programmer and named frmInvoice. Visual Basic enables us to use object from outside Visual Basic. Examine the above last line. We use Excel worksheet in our program.

7.1.3. Operator

Operator is a symbol used in programming language to carry out an operation against data. Operator's symbol may be a character or certain word. In Visual Basic, there are three (3) main operator groups, namely, arithmetics operator, comparative operator and logic operator.

- **Arithmetics Operator**

Arithmetics operator is used to perform mathematical operation on data. Arithmetics operator notation is shown in the following table.

Tabel 7.1. Arithmetics Operator

Symbol	Mathematical operation	Example
^	power	5 ^ 2 result 25
*	multiplication	5 * 2 result 10
/	divide	5 / 2 result 2.5
\	divide	5 \ 2 result 2
Mod	Remains of division	5 Mod 2 result 1
+	addition	5 + 2 result 7
-	subtraction	5 – 2 result 3
&	string combining	5 & 2 result 52

Notation / symbol for this operator has a hierarchical level, meaning that if two or more operator are used together, then the operator with higher hierarchy would be executed first before the lower hierarchy. The hierarchical level from the highet is as follows ^, * and/,\, mod, + and -. The operator * and / is equal level. Likewise the operator + and -. Examine the following example.

Example 7,3. Example of the hierarchical of Arithmetics operator.

```
5 * 2 + 3 = 13
4 ^ 2 - 5 = 11
```

In the first example, the sign * (multiplication) has a higher level than + sign (addition) so that multplication operation is carried out before the adding. The result is 10 plus 3, not 5 times 5. While in the second example, sign ^ (power) has a higher level than – (subtraction) so that the result would be 11 (from 16 minus 5). To change the

arithmetic sequence, it may be done by using parentheses. Examine the following example.

Example 7.4. The example of parentheses usage to arrange hierarchical operator.

5 * (2 + 3) = 25
4 ^ (2 – 5) = 0,015625

In this example, we use the same numbers and operators as the previous example, but using parentheses to change calculation sequence. In the first example, the first process is adding 2 and 3, then multiply by 5. The final result is 25, as compared with the previous example, that is 13. In Example two, the first process is 2 minus 5 then used as power of 4, i.e., 4 ^ 3. The result is 0,015625, big different from the previous result that is 11. From the two examples, it could be seen that parentheses have a higher level than arithmetics operator. If there is more then one parenthesis then the most inner parentheses will be processed first. Examine the following example.

5 * ((2 + 2) / 8) = 2,5

- **Comparative Operator**

Comparative Operator is used to do data comparison operation. Symbol used are as follows.

Table 7.2 Comparative Operator

Symbol	Comparative Operator	Example
<	Less then	5 < 2 results FALSE
>	Larger than	5 > 2 results TRUE
<=	Less then or equal	5 <= 2 results FALSE
>=	Larger then or equal	5 >=2 results TRUE
=	equal	5 =2 results FALSE
<>	Not equal	5 <> 2 results TRUE

- **Logical Operator**

Logical operator is used to compare two comparisons. Symbol used are as follows.

Table 7.3. Logical operator

Symbol	Logical Operator	Example
Or	Atau	(5<2) Or (5>2) results TRUE
And	Dan	(5<2) And (5>2) results FALSE
Not	Tidak	Not (5<2) results TRUE

7.1.4. Control Program Structure

Control program structure also known as control structure is an implementation of algorithm for branching structure and looping structure. Please review Chapter 5 for algorithm structure.

- **Branching Structure**

Branching Structure in Visual Basic may be done by using If... Then and Select... Case. When there is not too many branches, If... then may used. Whereas Select... Case is used in the case of many branching. Please examine the following example of using If ... Then.

Example 7.5. If .. Then.

```
' Single line branching without Else
If x > 0 Then y = x

' Single line branching with Else
If x > 0 Then y = x Else y = 0

' Single line branching with : and Else
If x > 0 Then y = x: x = 0 Else y = 0

' Branching written in more than one line
If x > 0 Then
      y = x
      x = 0
Else
      y = 0
End If

' Using Block IF
If x > 0 Then
      y = x
ElseIf x < 0 Then
      y = x * x
Else
```

```
      x = -1
End If
```

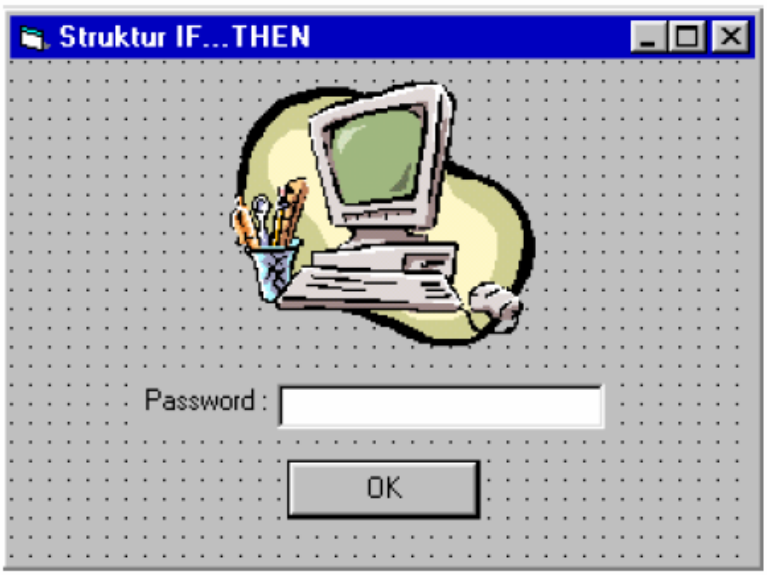
Example 7.6. Select ... Case.

```
Dim position As Integer          'Pilihan
position = CInt(txtPosition.Text)
Select Case position
   Case 1
     txtOutcome.Text = "Menang"
   Case 2
     txtOutcome.Text = "Kalah"
   Case 3
     txtOutcome.Text = "Seri"
   Case Else
     txtOutcome.Text = "Tidak bertanding."
End Select
```

Examine the following application of using If... Then and Select... Case.

Example 7.7. Application program with if... then.

Activate VB 6 then create a form as follows.



Arrange properties for the following respective objects.

Object	Properties	Value
Form5	Caption StartUpPosition	Struktur IF…THEN 2 – CenterScreen
Image1	Stretch Picture Visible	True Komputer.wmf False
Label1	Caption	Password :
Text1	PasswordChar Text	* <kosong>
Command1	Caption Default	OK True

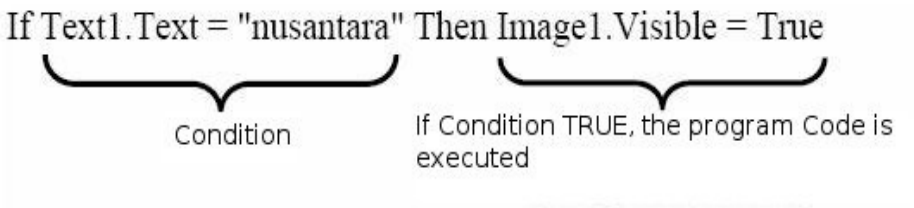
Open code windows and using the code editor write the source code as follows.

```
Private Sub Command1_Click()
   If Text1.Text = "nusantara" Then Image1.Visible = True
End Sub
```

Click on menu Project > Project1 Properties then click General tab. Change the Startup Object to Form 5. Try to run Project1:

- Type any text in the TextBox then click OK or press Enter, there will be nothing.
- Type “nusantara” in TextBox then click OK or press Enter, a computer picture will emerge.

Code explanation:



Modified the source code as follows

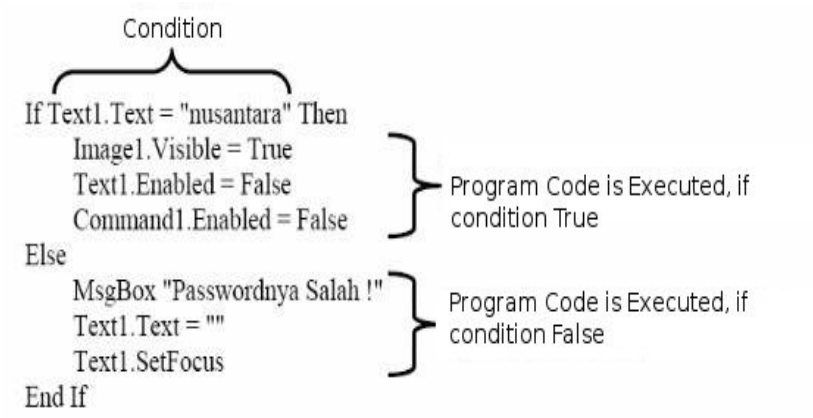
```
Private Sub Command1_Click()
If Text1.Text = "nusantara" Then
    Image1.Visible = True
    Text1.Enabled = False
    Command1.Enabled = False
Else
```

```
        MsgBox "Passwordnya Salah !"
        Text1.Text = ""
        Text1.SetFocus
    End If
    End Sub
```

Try re-run Project1:

- Type any text in TextBox then click OK or press Enter, shown a box with text “Passwordnya salah! ”.
- Type “nusantara” in TextBox then the click OK or press Enter, a computer picture will emerge. TextBox and OK switch is disabled and cannot be used.

Source code explanation:



Additional notes:

- The text “nusantara” must be in lower case. Remember: data string is case sensitive!
- To make password not case sensitive, modify the following statement using.

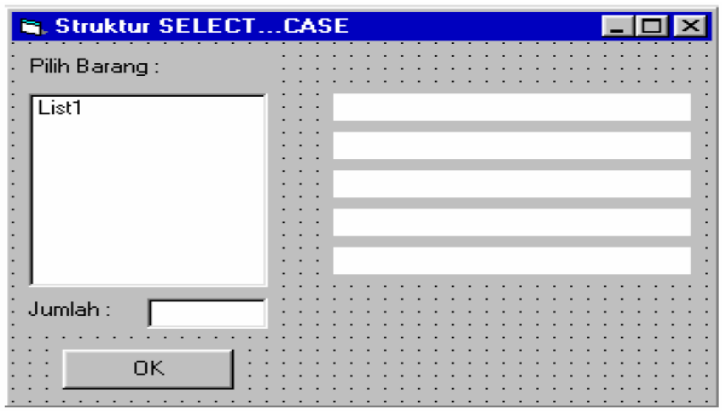
  If LCase (Text1.Text) = "nusantara" Then

  LCase fuction is to convert all string in Text1.Text to lower case, although user may input it in capital letters.

Example 7.8. Application program with Select... Case.

Activate VB 6 through Start button, then create the following Form.

Set properties for each object as follows.



Object	Properties	Value
Form6	Caption StartUpPosition	Struktur SELECT…CASE 2 – CenterScreen
Label1	Caption	Pilih Barang :
List1	-	-
Label2	Caption	Jumlah :
Text1	Text	<kosong>
Command1	Caption	OK
Label3-7	Name BackColor Caption	lblBarang, lblHarga, lblJumlah, lblDiskon, lblTotal Palette : <putih> <kosong>

Open code windows and in the code editor type the source code as follows.

```
Private Sub Form_Load()
    List1.AddItem "Disket"
    List1.AddItem "Buku"
    List1.AddItem "Kertas"
    List1.AddItem "Pulpen"
End Sub
Private Sub Command1_Click()
    Dim harga As Currency, total As Currency
    Dim jumlah As Integer
    Dim diskon As Single
    Dim satuan As String
    If List1.Text = "" Then
        MsgBox "Anda belum memilih barang !!"
```

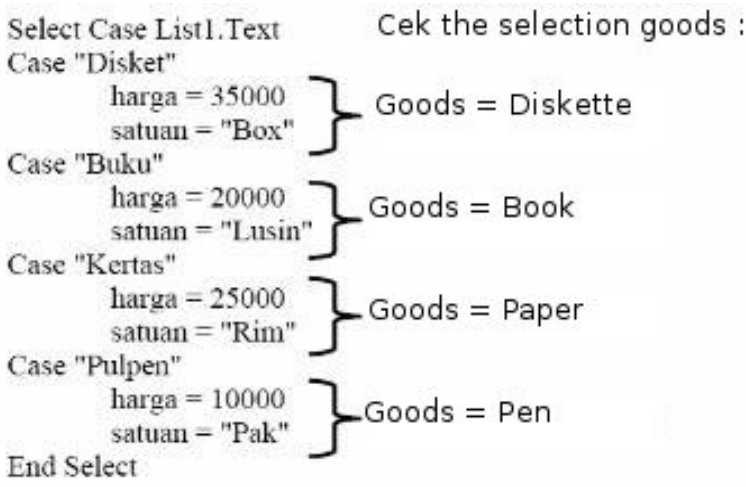
```
        List1.ListIndex = 0
        Exit Sub
    End If
    If Text1.Text = "" Then
        MsgBox "Anda belum mengisi jumlah barang !!"
        Text1.SetFocus
        Exit Sub
    End If
    Select Case List1.Text
        Case "Disket"
            harga = 35000
            satuan = "Box"
        Case "Buku"
            harga = 20000
            satuan = "Lusin"
        Case "Kertas"
            harga = 25000
            satuan = "Rim"
        Case "Pulpen"
            harga = 10000
            satuan = "Pak"
    End Select
    lblBarang.Caption = "Barang : " & List1.Text
    lblHarga.Caption = "Harga : " & Format(harga,
    "Currency") & "/" & satuan
    lblJumlah.Caption = "Jumlah : " & Text1.Text & " " &
    satuan
    jumlah = Text1.Text
    Select Case jumlah
        Case Is < 10
            diskon = 0
        Case 10 To 20
            diskon = 0.15
        Case Else
             diskon = 0.2
     End Select
     total = jumlah * (harga * (1 - diskon))
     lblDiskon.Caption = "Diskon : " & Format(diskon, "0
     %")
     lblTotal.Caption = "Total Bayar : " & Format(total,
     "Currency")
End Sub
```

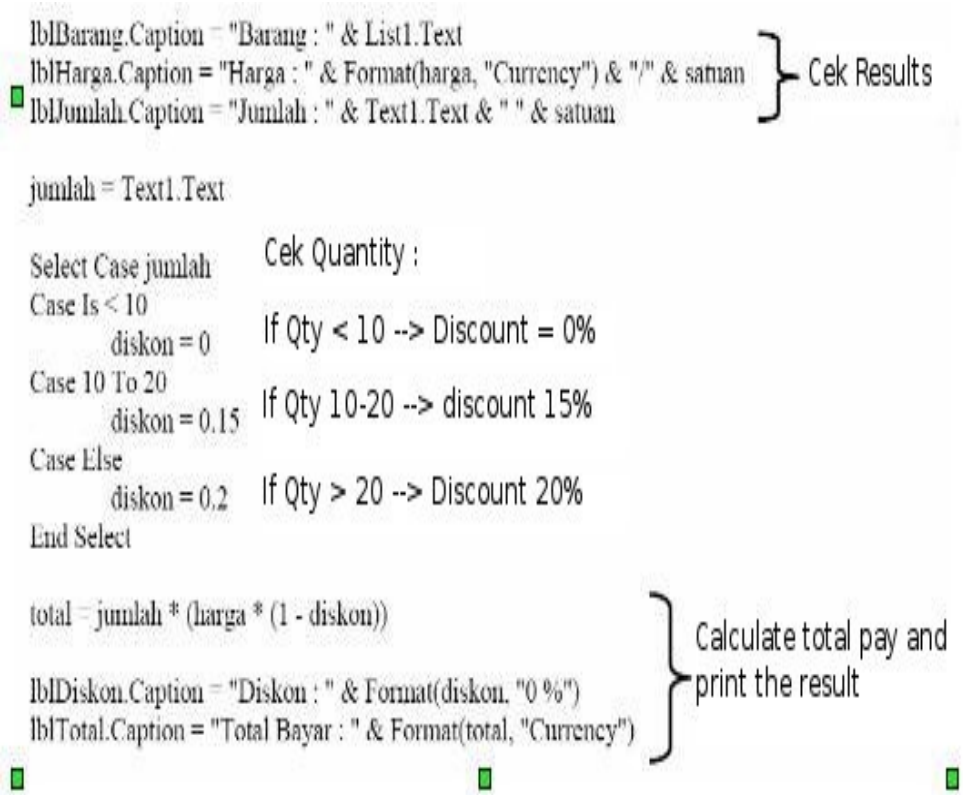
Click on menu Project > Project1 Properties then click General tab. Change Startup Object into Form6.

Please run Project1:

- List1 will be filled by name of items.
- Click OK, will be shown message box "Anda belum memilih barang!!"
- Click OK, the first item will be automatically selected. You may select other items.
- Click OK, will be shown message box “Anda belum mengisi jumlah barang!!!”
- Click OK, text1 will be the focus. Fill in the number of goods, for example: 10
- Clok OK, will shows item name, item price per unit, total item with its unit, discount and total price.
- Please change the name of items (in List 1) and total item (in Text1) then click OK.

Source code explanation:





Note:

- If the total item fills with other than numbers, an error message will be shown.
- To check whether Text1 is number only, add the following code:

```
If Not IsNumeric(Text1.Text) Then
    MsgBox "Isi jumlah barang harus angka !!"
    Text1.SetFocus
    Exit Sub
End If
```

- Loop Structure.

The mainly used loop structure in Visual Basic is For structure. In Visual Basic, the structure for this is known as For... Next. In general, the code is For... Next is as follows.

```
For counter = nilaiAwal To nilaiAkhir [Step increment]
    ' pernyataan yang akan diulang...
Next
```

The Step and increment statement may be reviewed in Chapter 5 and will not be discussed here. Examine the use of For .. Next in the following application.

Please examine the following For... Next

Example 7.9. Repetition using For ... Next.

```
Dim d As Single, count As Long
For d = 0 To 10 Step 2
    count = count + 1
Next
Print count
```

In Example 7,9, d is counter and we declare as single. We may use increment as 2. Increment value may be integer or floating point. However, floating point value sometimes give error results. How is the output from the program above? At the end the program, count value will be 5.

More flexible Loop structure then For .. Next is Do... Loop. Do... Loop could have different form. Examine the following example.

In Example 7.10. loop using For... Next.

```
Do While x > 0
    y = y + 1
    x = x \ 2
Loop

Do
    y = y + 1
    x = x \ 2
Loop Until x <= 0
```

In the first part of the Example 7.10 we use Do While... Loop. This is the same as While structure discussed in Chapter 5. The statement within Do While, will be executed if the condition to Do While is true.

The second part of Example 7,10, we use Do... Loop Until to do the looping. In this form to carry out the loop. In this form, looping is carried out until the Loop Until condition is true. Thus, as long as the Loop Until condition is false, loop will always be carried out. In other word, Do .. Loop is the opposite of Do While. Examine these two

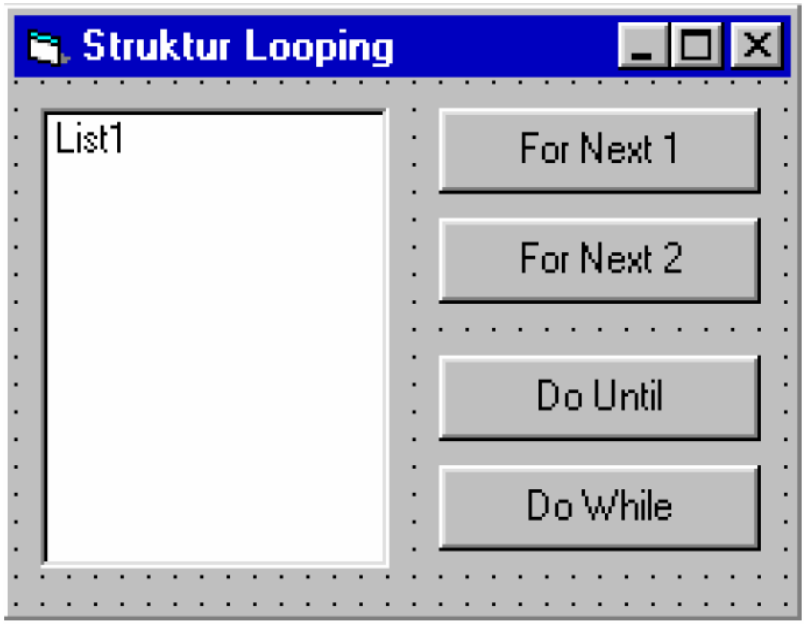
sections, if we initialize x = -4 and Y = 5, how is the result? In Do While, it will not give result as X less than 0 and looping is not carried out. Whereas in Do... Loop Until, the statement under Do will be executed. So that X value will be -2 and Y is 6.

The following is the application program example using a loop structure.

Example 7.11. Application program with loop structure.

Open VB and create the following form.

Set properties as shown in the following table.



Object	Properties	Value
Form7	Caption StartUpPosition	Struktur Looping 2 – CenterScreen
List1	-	-
Command1-4	Caption	For Next 1 For Next 2 Do Until Do While

Open windows code and in the Code Editor, type the following source code.

```
Dim i As Integer
Private Sub Command1_Click()
List1.Clear
 For i = 1 To 100
      List1.AddItem "Angka " & i
 Next i
 End Sub
 Private Sub Command2_Click()
 List1.Clear
 For i = 100 To 1 Step -2
      List1.AddItem "Angka " & i
 Next i
 End Sub
 Private Sub Command3_Click()
 List1.Clear
 i = Asc("A")
 Do Until i > Asc("Z")
      List1.AddItem "Huruf " & Chr(i)
      i = i + 1
 Loop
 End Sub
 Private Sub Command4_Click()
 List1.Clear
 i = Asc("Z")
 Do While i >= Asc("A")
      List1.AddItem "Huruf " & Chr(i)
      i = i - 1
 Loop
 End Sub
```

Run the program Press the four buttons in the form one at a time. Examine the output of the program.

7.1.5. Procedure and Function

There are several type of procedures used in Visual Basic:

- Sub procedure with no return value.
- Function procedure that returns value.
- Property procedure that could return the value and that refers to an object.
- Sub procedure

Syntax of sub procedure

```
[Private|Public][Static]Sub procedurename (argumen-argumen)
  statements
End Sub
```

Every time procedure is called, then statements between Sub and End Sub will be executed. The procedure's argument is a value passed during calling the procedure.

In Visual Basic, Sub Procedure can be split into two, namely,

- General Procedure, activated by the application.
- Event Procedure, activated by the system as to the response to event.

Example 7.12. Example of Sub procedure

In this example, we will create a sub procedure called CenterForm to show a half screen form, where x is the parameter in form of a form in the middle of the screen.

```
Sub CenterForm(x As Form)
  x.Top = (Screen.Height - x.Height) \ 2
  x.Left = (Screen.Width - x.Width) \ 2
End Sub

‘memanggil sub prosedur CenterForm
Private Sub Form_Load()
  Call CenterForm(Me)
End Sub
```

In Example 7.12, the sub procedure CenterForm needs the argument form. So that to call the sub procedure, one needs to include the argument. Examine the line that call CenterForm (Me). Me is the argument from sub procedure. In Visual Basic, Me refers to form where the source code is made.

- **Function procedure**

There are two kind of functions in Visual Basic, namely, Built-in Function and Function Procedure. In Visual Basic, many Built-in Functions are available for many purposes, such as, mathematical calculation, string manipulation, data type manipulation etc. In this section, we will discuss more detail on the built-in function. In the appendix, some frequently used built-in function is listed.

Although there are quite large collection of built-in functions, these functions are fairly general and may not suite programmer specific needs. Thus, for specific needs, we may create our own Procedure Function or we may create our own function.

Syntax of procedure function is

```
[Private|Public][Static]Function procedurename (argumen-argumen) [As type]
   statements
End Function
```

There are were three differences between function and procedure:

- In general, you could call a function by using function name on the right hand side in a statement of expression (returnvalue = function ()).
- Function has same data type as a variable. It sets the return data type.
- The return value is put into its function name, and some function may be part of a long expression.

Please examine the following function.

Example 7.13. Function Example.

The following function is to show the name of month in Indonesian from the input date. The needed input argument is x with data type as date.

```
Function Bulan(x As Date)
Dim sRet As String
Select    Case Month(x)
  Case   1: sRet = "Januari"
  Case   2: sRet = "Februari"
  Case   3: sRet = "Maret"
  Case   4: sRet = "April"
  Case   5: sRet = "Mei"
  Case   6: sRet = "Juni"
  Case   7: sRet = "Juli"
  Case   8: sRet = "Agustus"
  Case   9: sRet = "September"
  Case   10: sRet = "Oktober"
  Case   11: sRet = "Nopember"
  Case   12: sRet = "Desember"
  Case   Else
       sRet = "tidak sah"
  End Select
Bulan = sRet
End Function
```

7.2. Access and Database Manipulation with Visual Basic

One of the advantage of Visual Basic is in its ability to access and manipulate database since Visual Basic is developed by Microsoft that also made the Windows operating system. Thus, it has a quite complete set of support for database functionality.

Visually Basic provides many ways to access and to manipulate data. Examine the following figure.

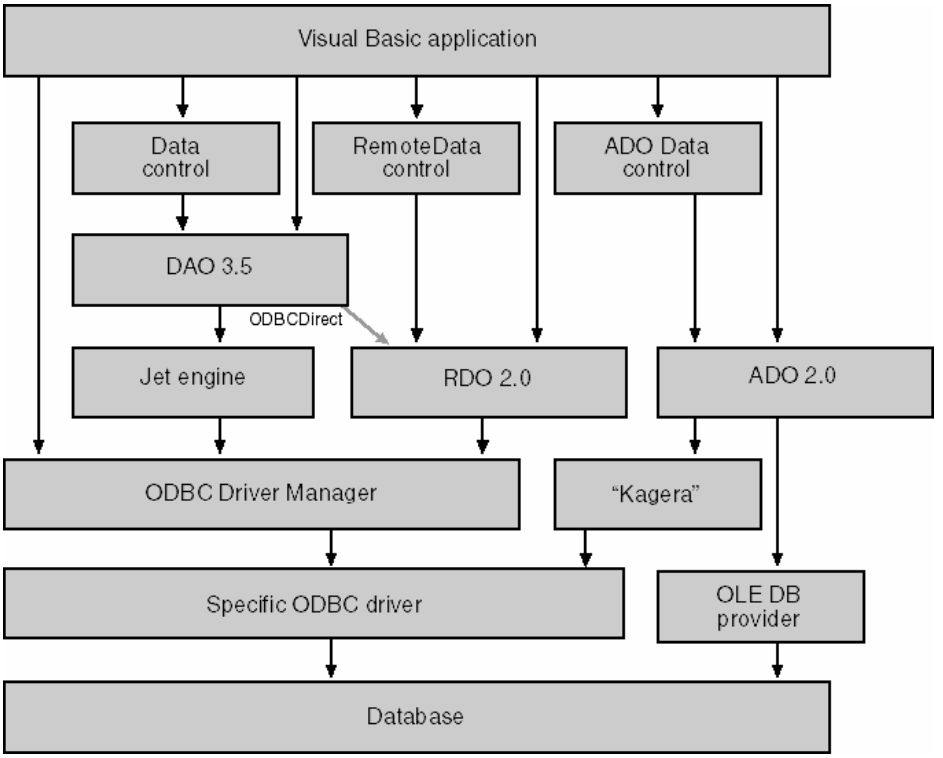


Figure 7.7. Many ways to access database in Visual Basic.

- **ODBC**

  ODBC, short for Open Data Base Connectivity, is a collection of functions that permit us to connect local or network database. ODBC is usually used to access various types of the database including, Ms FoxPro, Ms Access, Ms SQL Serve, Oracle or even data in raw data file format.

- **DAO**

  DAO, short for Data Access Object, is the interface for Microsoft Jet, the main driver behind MS Access. We could easily create a database with Ms Access

then manipulate it using Visual Basic via DAO. Since DAO has a direct connection to Ms Access, we could use DBMS functions from Visual Basic.

- **OLE DB**

  OLE DB is a technology to access low level database and is meant to replace the ODBC function. However, in its development, ODBC and OLE DB have some differences, namely, OLE DB based on COM technology and could be used to access non-relational database..

- **ADO**

  ADO (ActiveX Data Object) is a high level interface of OLE DB. ADO is developed on top of OLE DB to equip functions that is not available in OLE DB and facilitate programmer in making the application.

7.2.1. Create and Manipulate Database using ADO

The working method with database in Visual Basic is principally the same not depends on its access method. There are several stages needed to be able to work with database. In this section, we will use ADO as Needed by several stages to be able to work with the database. In this part we will use ADO as its technology is more flexible than other technology. It may always developed using other technology.

- **Connection using database.**

Connection using database means we connect to open the database and access the data in it. The following is an example of connecting to database biblio.mdb.

```
Dim cn As New ADODB.Connection
cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;"
_
    & "Data Source=C:\Microsoft Visual
Studio\Vb98\Biblio.mdb"
```

- **Access record data in database.**

A collection of data record in database, in ADO, is known as recordset. To work with recordset, we need access to table or view / query to see what is available in the database. Exampine the following example.

```
Const DBPATH = "C:\Program Files\Microsoft Visual
Studio\Vb98\NWind.mdb"
Dim cn As New ADODB.Connection, rs As New ADODB.Recordset
```

```
  cn.Open "Provider=Microsoft.Jet.OLEDB.3.51;Data Source=" &
  DBPATH
rs.Source = "Employees"
rs.Open , cn
```

In the above code, we use database Nwind.mdb as data source. Then as source recordset, we call “Employees” table. Variable rs is the recordset. After determining the source table, we may open the table using open statement.

After open the source table, we may access the data in it. The following is the source code example to access data in a table.

```
Dim i As Integer
For i = 0 To rs.Fields.Count
    Print rs.Fields(i).Name & " = " & rs.Fields(i).Value
Next
```

The above statement will print all row and column Name in the Employees table that is previously opened. The fastest method is by using For Each statement as follows,

```
Dim fld As ADODB.Field
For Each fld In rs.Fields
    Print fld.Name & " = " & fld
Next
```

- **Data manipulation in recordset.**

Data update on recordset may be done as follows.

```
rs.Update Array("FirstName", "LastName", "BirthDate", "HireDate"), _
    Array("John", "Smith", #1/1/1961#, #12/3/1994#)
```

While to add a recordset, the following statement may be used.

```
rs.AddNew
rs("FirstName") = "Robert"
rs("LastName") = "Doe
rs("BirthDate") = #2/5/1955#
rs.Update
```

To erase a record, it may be done by the following statement.

```
rs.Delete
rs.MoveNext
If rs.EOF Then rs.MoveLast
```

7.3. COM Technology

7.3.1. COM Concept

COM or Component Object Model is an infrastructure provided by Visual Basic to access objects or controls as long as it has interface accessible by Visual Basic.

To be able to use COM in Visual Basic, we may open Reference Dialog from Project menu select add Reference. The following windows as Figure 7,8 will be opened.

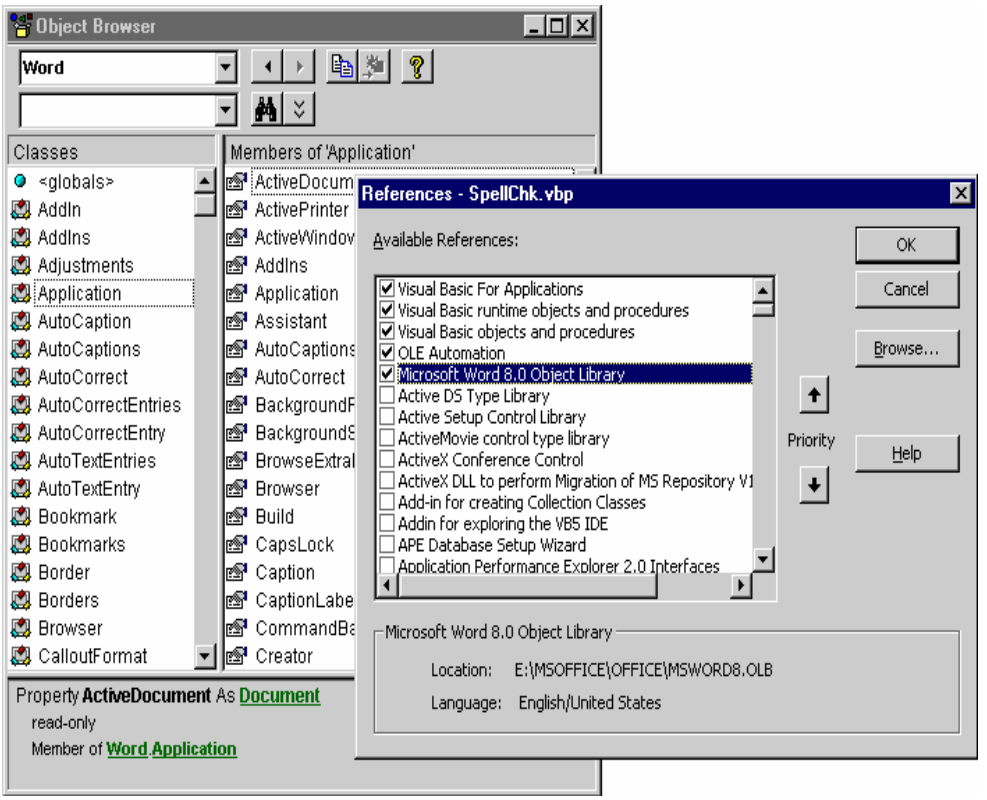


Figure 7.8. Reference Windows.

In Figure 7.8 select COM or other object that we'll like to use. In this example, we select “Microsoft Word 8,0 Object Library”. After click OK, we could use the choosen object in our program. The following is the example of object usage.

Example 7.14. Example of COM usage.

```
Private    Sub cmdCheck_Click()
     Dim  text As String
     Dim  suggestion As Word.SpellingSuggestion
     Dim  colSuggestions As Word.SpellingSuggestions
     ' menambahkan dokumen bila belum ada dokumen yang
terbuka.
     If MSWord.Documents.Count = 0 Then MSWord.Documents.Add
     text = Trim$(txtWord.text)
     lstSuggestions.Clear
     If MSWord.CheckSpelling(text) Then
          lstSuggestions.AddItem "(correct)"
     Else
          Set colSuggestions =
MSWord.GetSpellingSuggestions(text)
          If colSuggestions.Count = 0 Then
               lstSuggestions.AddItem "(no suggestions)"
          Else
               For Each suggestion In colSuggestions
                    lstSuggestions.AddItem suggestion.Name
               Next
          End If
     End If
End Sub
```

In the above code, we use one of the method from the previously opened object (“Microsoft Word 8,0 Object Library”). The method that we like to use is Spelling Suggestion. Please see the code in the variable declaration section. Using the same way, we may use method or function that we like to use from COM object that has been loaded.

In the Reference windows in Figure 7.8, there are a large number of COM that can be used. Please try to open it and examine its function one by one.

7.4. SUMMARY

In this chapter, you have studied the Visual Basic programming language. Start with variable, constant, data type and operator. Then continue with control structure, namely, branching and looping. Procedure and function usage is also provided to complete the foundation of Visual Basic programming.

In other section, we also studied access and data manipulation technique using ADO. This chapter is closed by studying how to open and use COM technology provided by Visual Basic.

7.5. EXERCISE

1. What is the result of the following Visual Basic expression:

   a. 3*4
   b. 7^2
   c. 1/ (2^3)
   d. 3 + (4*5)
   e. (5 - 3) *4
   f. 3* ((-2) ^5)

2. What is the result using the following mod expression.

   a. 6 Mod 2
   b. 14 Mod 4
   c. 7 Mod 3
   d. 5 Mod 5

3. Check the naming of variable in Visual Basic. Determine whether it is true or false.

   a. sales.2006
   b. room&Board
   c. fOrM_1040
   d. 1040B
   e. expenses?
   f. INCOME 2006

4. If a = 2, b = 3, and c = 4, what is the result of the following expression.

   a. (a*b) + c
   b. a*(b + c)
   c. (1 + b)*c
   d. a^c
   e. b^(c - a)
   f. (c - a)^b

5. Create a program to calculate the following expression.
   a. 7•8 + 5
   b. $(1 + 2•9)^3$
   c. 5.5% of 20
   d. $15 - 3(2 + 3^4)$
   e. 17(3 + 162)
   f. 4 1/2 - 3 5/8

6. Open Figure 5.6 and 5.7 in Chapter 5. Create the program in Visual Basic. Use Text Box control and Command Button in this exercise.

7. Open Figure 5.9 and 5.10 in Chapter 5. Create the program in Visual Basic.

8. Open Figure 5.15 and 5.17 in Chapter 5. Create the program in Visual Basic.