PROGRAMMING LANGUAGE 2 (SPM 3112)

PROCEDURES & FUNCTIONS (PART 1)

NOOR AZEAN ATAN
MULTIMEDIA EDUCATIONAL DEPARTMENT
UNIVERSITI TEKNOLOGI MALAYSIA

Topics

- **General Procedure**
- **Event Procedure**

Introduction

- **What is procedure?**
  - **Procedure → a self-contained routines within a larger program that carry out specific tasks.**
  - **A sequence of statements that perform a specific task.**

- **Why we need procedures?**
  - **Organize code in program**
  - **More manageable?**
  - **Code are easier to maintain?**

Introduction

- **When we need procedures?**
  - **When need to repeat the same process over & over in a program.**
  - **Procedure can be called many times → but appears in the code once.**

Procedure

- **The format for the procedure is:**

Scope Sub ProcedureName (Argument)

Statement

End Sub

Scope – Procedure's access area (Public or Private)

Sub & End Sub – keywords for creating procedure

ProcedureName – Procedure name (must be unique)

Argument – data exchange

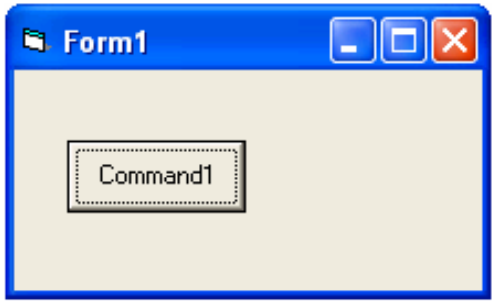
Procedure

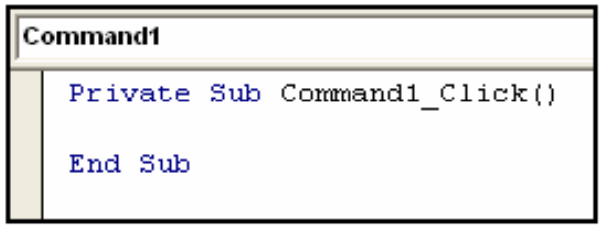
- **Ways of creating procedure?**
  - **By using the format**

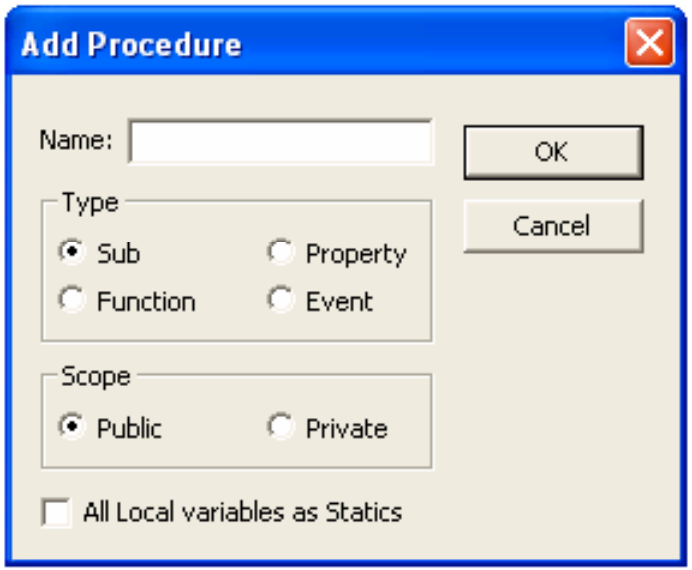
    Scope Sub ProcedureName (Argument)

    Statement

    End Sub
  - **Tool < Add Procedure**
  - **By double clicking objects**







Procedure

Example:

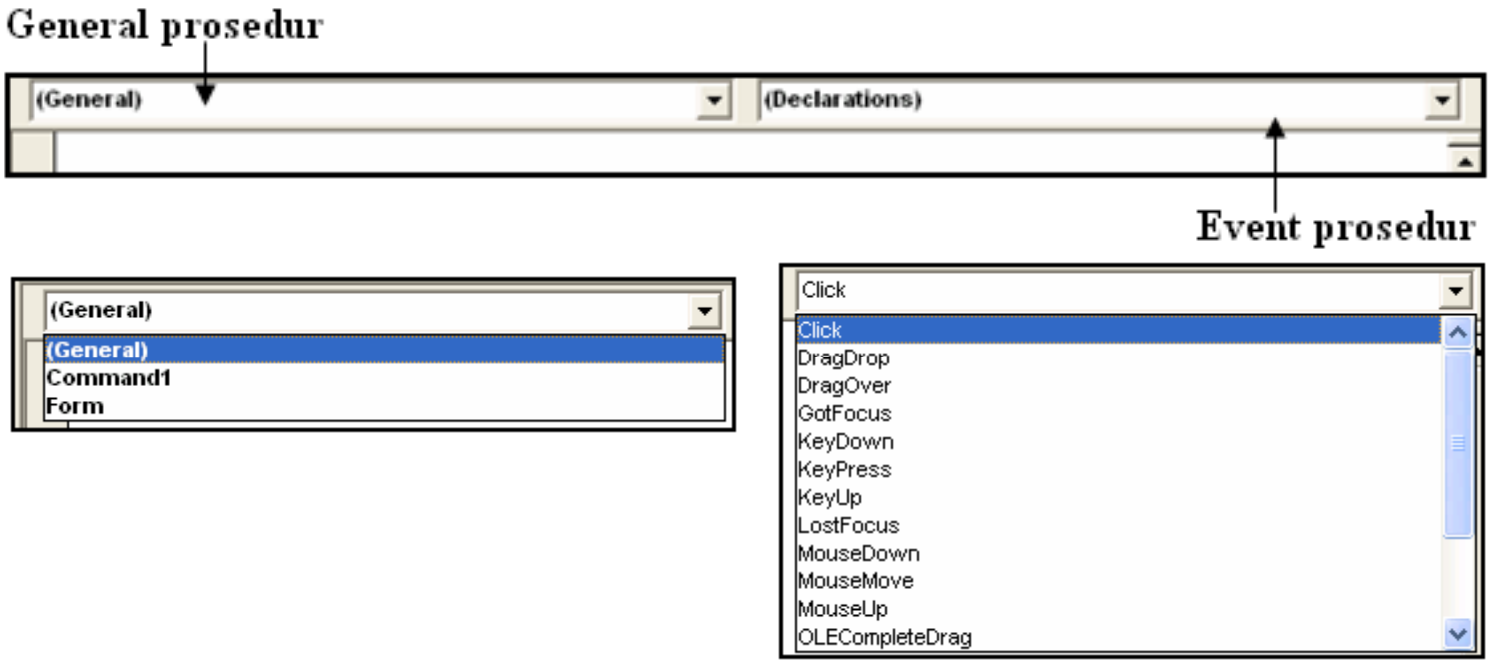
```
1. Private Sub Command1_Click( )
        Msgbox (“Hello Apa Khabar?”)
   End Sub

2. Private Sub Message_Click( )
        Print "Hellooooooo!!!!!"
   End Sub

3. Private Sub Form_Load()
        Frame1.Visible = True
   End Sub
```

Type of Procedures

- **There are two types of procedures:**
  - **General procedure**
  - **Event procedure**



Type of Procedures

The differences?

- **General procedures**
  - **Not associated with events**
  - **Procedure executes when called by another procedure**

- **Event procedures**
  - **Associated with object events**
  - **Procedure executes when the associated event occurs**

Test Yourself

Which of the following is general or event procedure?

```
1. Private Sub Command1_Click( )
     Msgbox ("Hello Apa Khabar?")
   End Sub

2. Private Sub Message( )
     Print "Hello"
   End Sub

3. Private Sub Form_Load()
     Total = 0
   End Sub
```

General procedures

- **Two levels of procedure scope**
  - ***Private*: Can be called from any other procedures on a form**
  - ***Public*: Can be called from any procedures on any form in the project**

General procedures

- Procedures are called from within another procedure
  - by using the called procedure's name
- Some procedures have no arguments
- Others, → need to specify arguments when calling the procedure
  - Arguments can be constants, expressions, or variables

General Procedures

General procedures are divided into:

- **Sub**
- **Function**

The differences?

- **Sub**

  → **No return value**
- **Function**

  → **Return value**

General Procedure

Example:

1. Sub

```
Private Sub Message( )
        Print "Hello"
End
Private Sub Command1_Click( )
        Call Message
End Sub
```

No value, just call

2.Function

```
Private Function luassegitiga(lebar As Integer, tinggi As Integer)
        luassegitiga = (lebar * tinggi)
End Function
Private Sub Form_Click( )
        luas = luassegitiga(2, 4)
        Print luas
End Sub
```

Call with value

Steps to Create General Subroutine

1. Make sure code window is active
2. Place cursor outside any other procedures
3. Type the procedure header line

   Use this syntax:
   Private Sub name_of_sub(arguments and types here)
4. VB will put in the *End Sub*
5. You type the body of the procedure
6. View or edit by clicking on the General object and selecting the procedure

   To give the procedure a PUBLIC scope, use
   Public Sub name_of_sub(arguments and types here)

Passing arguments ("parameter passing")

- The values of arguments → "passed" to/from an subroutine or function

- When variables are used as inputs
  - Variable types must match in
    - the calling procedure
    - in the called procedure

  - Order of variables must match between the calling procedure and the called procedure

More on Parameter Passing *ByVal*

- Called procedure sets up new memory locations
- Value of parameter is copied into the new locations
- Contents of the original variable does not change

```
Sub Add_And_Print(ByVal sX As Single, ByVal sY As Single)
   sX = sX + sY
   Print sX
End Sub

Private Sub Command1_Click()
   Call Add_And_Print(Text1.Text, Text2.Text)
End Sub
```

More on Parameter Passing *ByRef*

- ***ByRef*** **is the default in VB**
- **Parameters → can also be passed *ByRef***
- **Called procedure uses same memory location for the variable**
- **Assigns a new name for that location, uses that new name within the procedure**
- **Contents of the original variable may be changed**

```
Sub Add_And_Change (ByRef sX as Single, ByRef sY as Single)
    sX = sX + sY
    Print sX
End Sub
```

More on Parameter Passing ByRef

```
Private Sub Add_And_Change (ByRef nom1 As Integer, ByRef
    nom2 As Integer)
Dim total As Integer

total = nom1 + nom2
MsgBox total

nom1 = nom1 + 5
nom2 = nom2 + 5
End Sub
```

Event Procedures

- Associate with control or object events
- Event procedures will execute when the event occur
- Example of event procedures are:
  - Click
  - Key press
  - Drag & Drop
  - Mouse State (Down, Up & Move)
  - etc

Event Procedures

Key Press

```
Private Sub Text5_KeyPress(KeyAscii As Integer)
name = Text4.Text
If KeyAscii = vbKeyReturn Then
   MsgBox ("Hi Miss@Mr " + name + "  Welcome to my menu!!")
   End If
End Sub
```

Event Procedures

Drag & Drop

```
Private Sub Picture1_DragDrop(Source As Control, X As Single, Y
    As Single)
Picture1.Picture = LoadPicture(App.Path & "\
    Landscapeafter.jpg")
Image1.Visible = False
MsgBox ("Yes Yes.. Correct!!!!!!")
End Sub
```

Event Procedures

Mouse State

- **Mouse Down - Any mouse button is pressed**
- **Mouse Up - Any mouse button is released**
- **Mouse Move -The mouse pointer is moved to a new point on the screen.**

Event Procedures

```
Private Sub Form_MouseDown(Button As Integer, Shift As
   Integer,
    X As Single, Y As Single)
   lakaran = True
   CurrentX = X
   CurrentY = Y
End Sub
```

Event Procedures

```
Private Sub Form_MouseMove(Button As Integer, Shift As
   Integer, X As Single, Y As Single)
If lakaran = TrueThen
Line -(X, Y)
End If
DrawWidth = 6
End Sub
```

Event Procedures

Mouse State

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X
    As Single, Y As Single)
lakaran = False
End Sub
```