CENTRE FOR DIPLOMA STUDIES

MULTIMEDIA CONCEPT

LABORATORY INSTRUCTION SHEET

Course Code	**DAT 21103**
Lab Practical Title	**PHASES IN DEVELOPING A VISUAL BASIC APPLICATION**
Lab Practical	**2**

Learning Outcome(s)

At the end of this practical session, you should be able to:

Follow steps in 3 phases of VB.Net application development efficiently.

Identify syntax, run-time, and logic errors during application development.

Guided Task 2.1: Very Busy (Vb) Mail Order

Case Study:

"If you don't have time to look for all those hard-to-find items, tell us what you're looking for. We'll send you a catalog from the appropriate company or order for you. We can place an order and ship it to you. We also help with shopping for gifts; your order can be gift wrapped and sent anywhere you wish."

– Company Director –

Shorten company title to "VB Mail Order". Include this name on title bar of first form of each project for this case study.

First job:

- Project that will display name and telephone number for contact person for customer relations, marketing, order processing and shipping departments.
- Button for each department.
- When user clicks on button for a department, display name and telephone number for contact person in 2 labels.
- Identifying labels - Text "Department Contact", "Telephone Number".
- Button for Exit.
- Label at bottom of form that holds your name.
- Test Data (Figure 2.1):

Department	Department Contact	Telephone No
Customer Relations	Tricia Mills	500-1111
Marketing	Michelle Rigner	500-2222
Order Processing	Kenna De Voss	500-3333
Shipping	Eric Andrews	500-4444

Figure 2.1

Solve problem using following phases:

Phase 1 Analysis

In Microsoft Word, list the analysis of application's requirement as follow:

Step 1.1: List all i**nput** or data to be entered in application. List will answer question, "Do application need to get some input from user?"

Step 1.2: List all **desired output** or result from application. List will answer question," Do application need to display anything?"

Step 1.3: List all **required process** to implement in application to obtain desired output. List will answer question, "How will application get desired output? Any event required? Any calculation required in event?"

Phase 2 Design and Plan

Complete step 2.3 in this phase based on following information.

Step 2.1: **Sketch user interface**

Based on information in Phase 1, sketch Figure 2.2 in Microsoft Word.

Step 2.2: **List basic Properties** for each object planned at Figure 2.2.

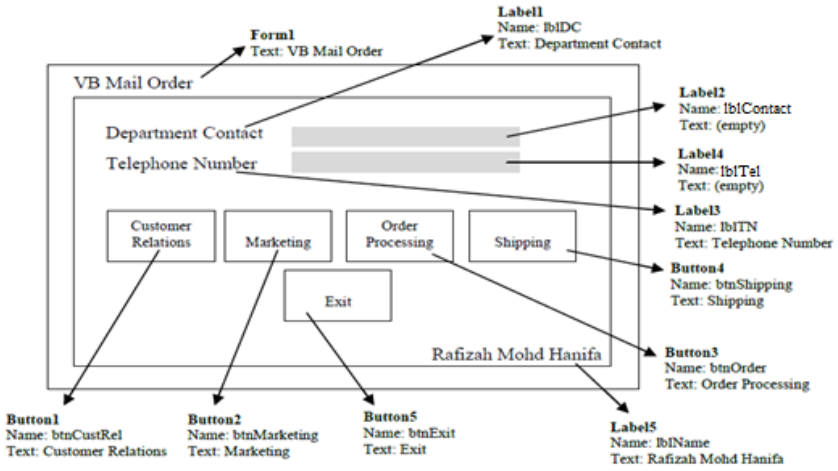

Figure 2.2

Step 2.3: **Write algorithm**

Algorithm used to determine which events required at specific object and step-by-step plan for those events. In Microsoft Word, complete this algorithm in order to complete your plan for event for each button.

Create following events in *frmHello*:

a. Event Click for *btnCustRel*:
   i. Display …….. at *lblContact.Text*
   ii. Display …….. at *lblTel.Text*
b. Event Click for `btnMarketing`
   i. Display …….. at `lblContact.Text`
   ii. Display …….. at `lblTel.Text`

c. Event Click for `btnOrder`
   i. Display …….. at `lblContact.Text`
   ii. Display …….. at `lblTel.Text`

d. Event Click for `btnShipping`
   i. Display …….. at `lblContact.Text`
   ii. Display …….. at `lblTel.Text`

e. Event Click for `btnExit`
   i. Close form using `Close` Method

Phase 3 Implement

Step 3.1: **Creating User Interface**

3.1.1 **Creating a new Visual Basic program**

File>New>Project. Choose Visual Basic language before creating project.

3.1.2 **Naming a project**

Select Windows Forms Application from Templates area.

Set Name to "MailOrder". Select OK. This should create a new window application project, which includes a single startup form and is contained within a "MailOrder" solution, as shown in the Solution Explorer window.

3.1.3 **Placing objects control onto user interface**

Place following object controls by dragging objects from Toolbox and dropping them onto form (Figure 2.3):

i. 5 buttons
ii. 5 labels

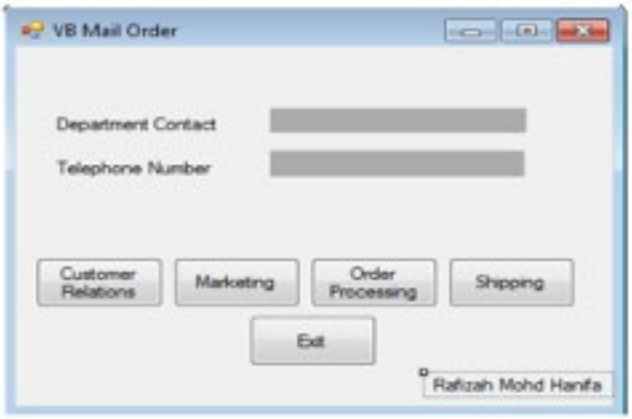

Figure 2.3

Step 3.2: **Setting Properties**

3.2.1 **Change default name** of each object at Name property. Use descriptive names as planned in Guided Task 2.1, Phase 2 (Figure 2.4).

3.2.2 **Set text** of each object at Text property (Figure 2.4).

3.2.3 **Set other attributes** of object if needed such as Font Type, Back Color, Font Color, etc. to make interface looks better.

Object	Properties
Form	Text: VB Mail Order
Label1	(Name): lblDC Text: Department Contact
Label2	(Name): lblContact Text:
Label3	(Name): lblTN Text: Telephone Number
Label4	(Name): lblTel Text:
Label5	(Name): lblName Text: Rafizah Mohd Hanifa
Button1	(Name): btnCR Text: Customer Relations
Button2	(Name): btnMarketing Text: Marketing
Button3	(Name): btnOP Text: Order Processing
Button4	(Name): btnShipping Text: Shipping
Button5	(Name): btnExit Text: Exit

Figure 2.4

Step 3.3: **Writing program code**

Double click on each object which involve an event and write suitable statements in event procedure as planned (algorithm) in Guided Task 2.1, Phase 2 (Figure 2.5).

Step 3.4: **Running an application**

Debug>Start Debugging. Click all buttons to check output. Make sure same as planned.

Button	Coding

Customer Relations	`Private Sub btnCR_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCR.Click`
	`lblContact.Text = "Tricia Mills"`
	`lblTel.Text = "500-1111"`
	`End Sub`

Marketing	`Private Sub btnMarketing_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnMarketing.Click`
	`lblContact.Text = "Michelle Rigner"`
	`lblTel.Text = "500-2222"`
	`End Sub`

Order Processing	`Private Sub btnOP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOP.Click`
	`lblContact.Text = "Kenna De Voss"`
	`lblTel.Text = "500-3333"`
	`End Sub`

Shipping	`Private Sub btnShipping_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnShipping.Click`
	`lblContact.Text = "Eric Andrews"`
	`lblTel.Text = "500-4444"`
	`End Sub`

Exit	`Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit.Click`
	`Me.Close()`
	`End Sub`

Figure 2.5 Program code for each event procedure

Guided Task 2.2: Application with Syntax Error

Step 1: Create a new project named `syntaxerror`.

Step 2: Add a button and a textbox to form. Leave **Name** properties as Button1 and Textbox1.

Step 3: Double click button to access its code, and type a line of code as follows:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
                          ByVal e As System.EventArgs) _
                          Handles Button1.Click

    Textbox2.Text = "Debug"

End Sub
```

When finish typing line, VB.NET puts a blue wiggly line under Textbox2.

Step 4: Hold mouse over **Textbox2**, a yellow tool tip appears, as shown below:

```
Textbox2.Text = "Debug"
```

Name 'Textbox2' is not declared.

Error occurs because no textbox named Textbox2.

Step 5: Change code as follows:

```
strText = "Debug"
TextBox1.Text = strText
```

VB.NET puts wiggly line under all occurrences of `strText`.

Step 6: Hold mouse over variable `strText` and you'll see the "not declared" tip again.

What does this mean?

Step 7: Change code as follows:

```
Dim strText As String
strText = "Debug"
TextBox1.Text = strText
```

Wiggly lines disappear. Why?

Step 8: Change code as follows:

```
strText = "Debug"
TextBox1.Text = strText
Dim strText As String
```

Wiggly lines appear. Why?

Step 9: Run application.

Step 10: **Error List** window (Figure 2.6) lists report of error (If can't see Error List window, View> Error List):

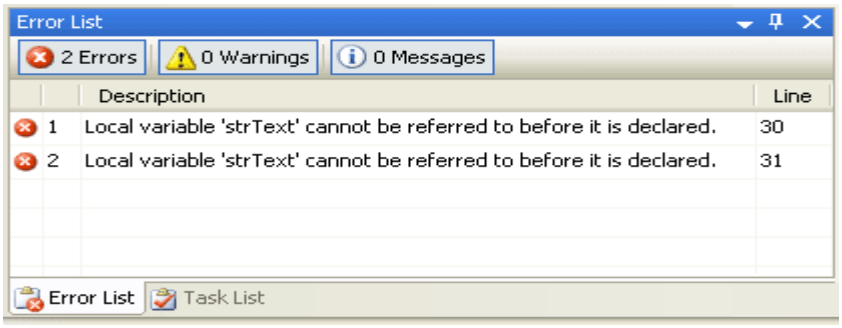

Figure 2.6 Error List window

Description of error - "**Local variable '`strText`' cannot be referred to before it is declared**". Double click left icons. VB.NET will highlight error in code. If still can't see Error List window, look at bottom left (Figure 2.7). Click Error List tab:

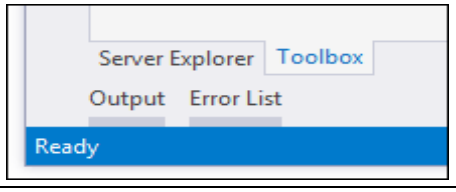

Figure 2.7 Error List tab

Step 11: Move **Dim …** line back to top. Blue wiggly lines go away and Error List erased.

Guided Task 2.3: Application with Run Time Error

Step 1: Create new project named runtimerror.
Step 2: Add a button and a textbox to form. Leave **Name** properties to **Button1** and **Textbox1.**
Step 3: Double click button to access its code, and type a line of code as follows:

```
Dim Num1 As Integer
Dim Num2 As Integer
Num1 = 10
Num2 = 0
TextBox1.Text = CInt(Num1 / Num2)
```

Step 4: Run application. Click button. Error message (Figure 2.8) will pop up. Why?

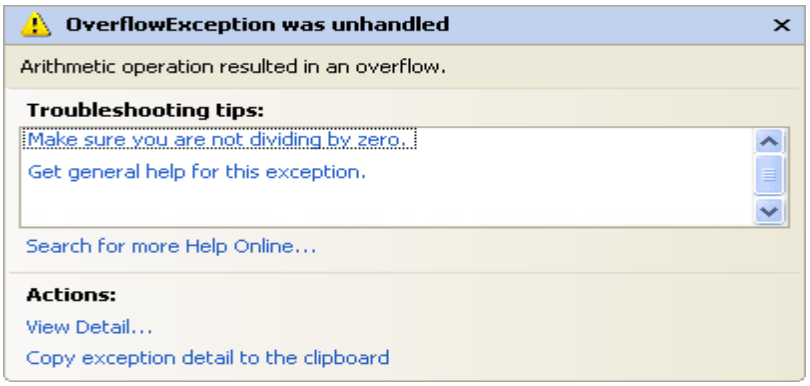

Figure 2.8 Error message

Step 5: Click Break button to stop application from running. Remove Cint. Run. Click button. See output. Why?

Guided Task 2.4: Application with Logic Error

Step 1: Create new project named logicerror.
Step 2: Add a button and a textbox to form. Leave **Name** properties to **Button1** and **Textbox1.**
Step 3: Double click button to access its code, and type a line of code as follows:

```
Dim x As Integer
Dim y As Integer
Dim answer As Integer
x = 10.5
```

```
y = 3
answer = x * y
TextBox1.Text = answer
```

Step 4: Before running application, what do you expect to get?

Step 5: Run application. Click button. See output. Why?

Step 6: Stop application. Assign 10.6 to variable x. Run application. Click button. See output. Why?

Step 7: Stop application. Change data type of variable x to Double. Run application. Click button. See output. Why?

Step 8: Stop application. Change data type of variable y to Double. Run application. Click button. See output. Why?