# 2010

# **VB.NET Programming**

# By mkaatr

This ebook is a quick introduction to VB.NET programming language. It explains the very basics of the language with screenshots showing what is expected to see during development process.

mkaatr http://www.mkasoft.com 10/13/2010



# **VB.NET Programming**

By mkaatr

## **Table of Contents**

Introduction	4
Chapter 1: Installation	5
Chapter 2: Understanding the IDE	32
Chapter 3: Understanding Buttons, and Textboxes	68
Chapter 4: Dialogs and Menus	109
Chapter 5: Understanding variables	133
Chapter 6: Variables again, group box, list box	139
Chapter 7: IF statement	150
Chapter 8: FOR statement	158
Chapter 9: Arrays	163
Chapter 10: Collections	169
Chapter 11: Functions	177
Chapter 12: ByVal & ByRef	
Chapter 13: Subroutines	189
Chapter 14: Do Loop	193
Chapter 15: Structures	200
Chapter 16: Modules	208
Chapter 17:Classes	218
Chapter 18: Classes Initialization and Finalization	242
Chapter 19: Classes and Inheritance	248
Chapter 20: Try & Catch	258

### Introduction

This ebook is the result of combining a number of tutorials available on the site. It gives basic information about using the language, however it does not cover every aspect of the language. There are lots to learn. I tried to put screen shots in most of the tutorials along with the source files embedded in the PDF file itself. If you still need to see how to do some stuff, check the website, there you can watch video tutorials which cover the same exact topics in this book.

Feedback is highly appreciated. If you have any comments, notes, or recommendations, send them to <u>notes@mkasoft.com</u>. Finally if you find this book useful, you might want to support this work and further tutorials by sending a small donation to <u>donation@mkasoft.com</u>.

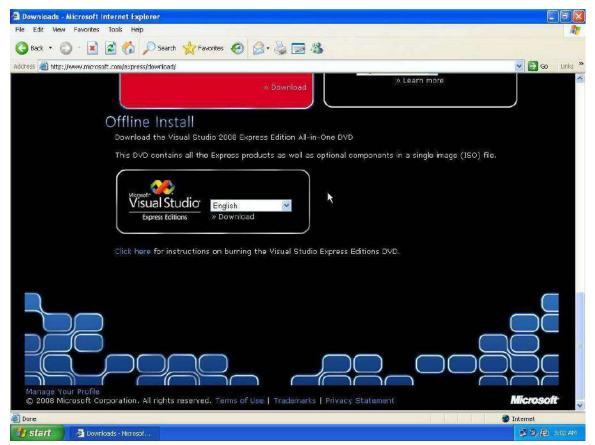
Thank you.

#### mkaatr

**Chapter 1: Installation** 

#### Installation

Installation of VB.NET should not be hard to do. The installation might be a little different since Microsoft updates the site from time to time. All you have to do is follow the steps below:



Scrool down and select offine install. You need to burn this ISO image into a disk before using it... or you can select web install for Visual Basic .net...



Select Visual Basic 2008 from the list...



The installation appears after a few seconds



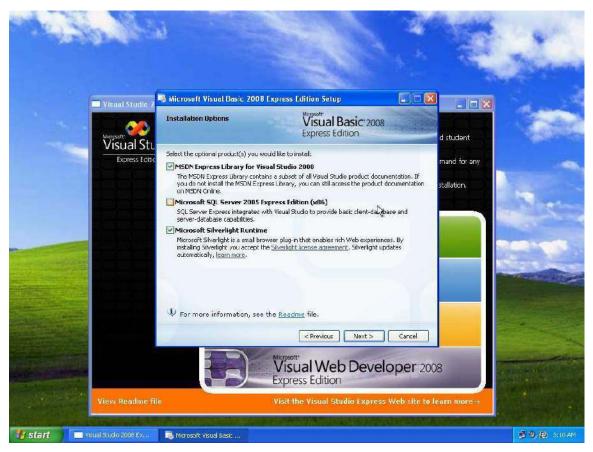
Wait for a while



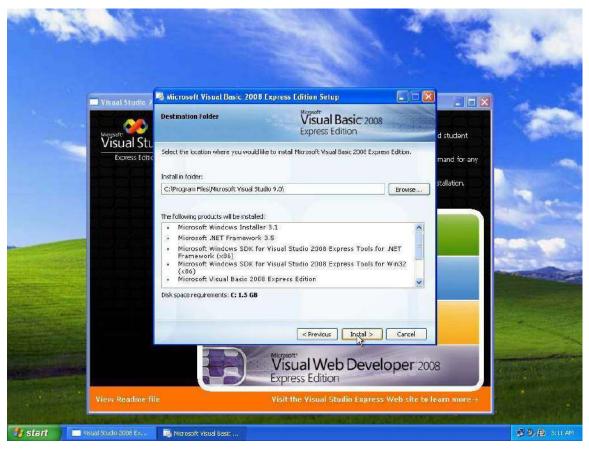
Click next...



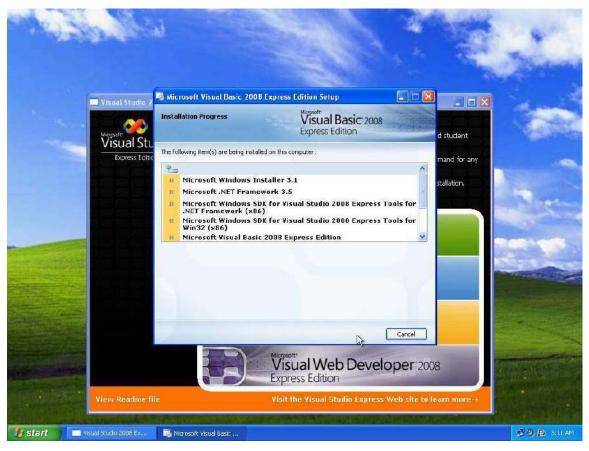
Read the agreement and agree about it... then click next...



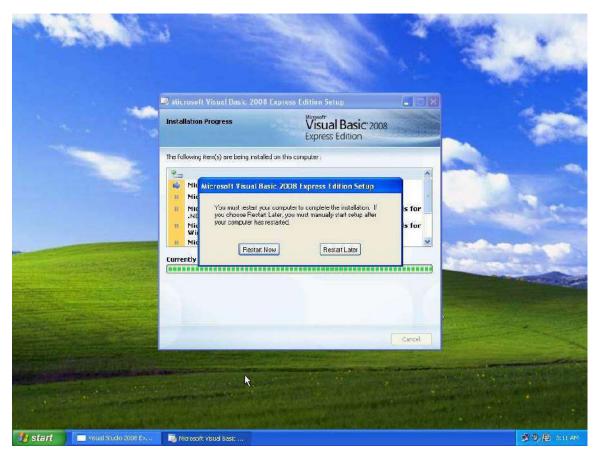
Select the MSDN (the help library) and Silver Light then click next...



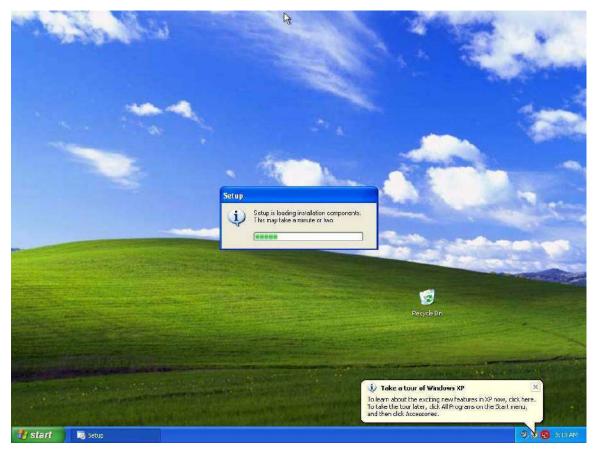
Click next...



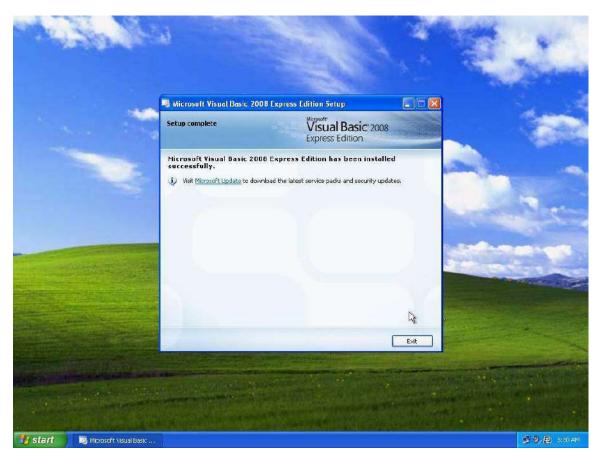
The installation starts...



After a while a restart will be required... click restart now...



Do not press or do anything... the installation continues automatically...



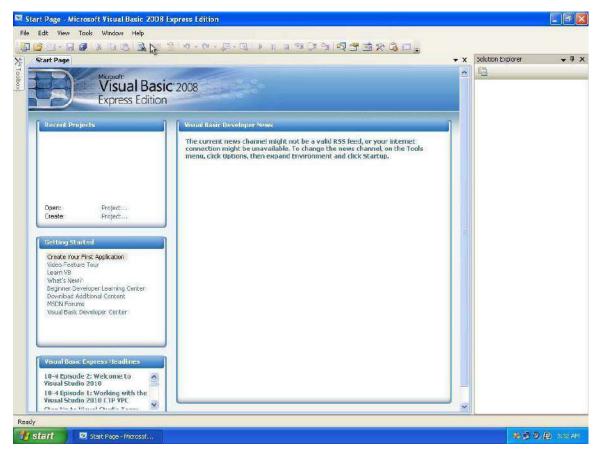
After the installation finishes, press exit... now let us test the VB.Net and write a simple application.



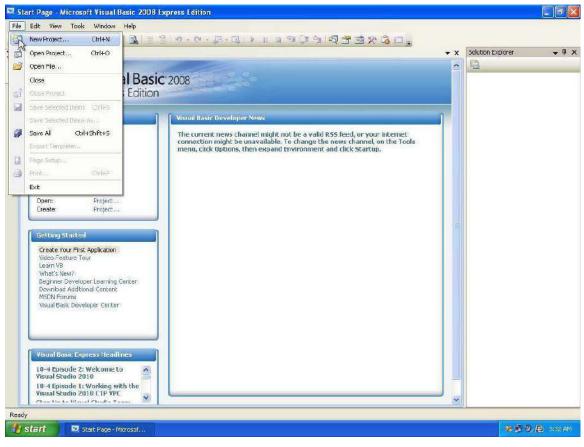
Go to All Programs->Microsoft Visual Basic 2008...



VB will start...



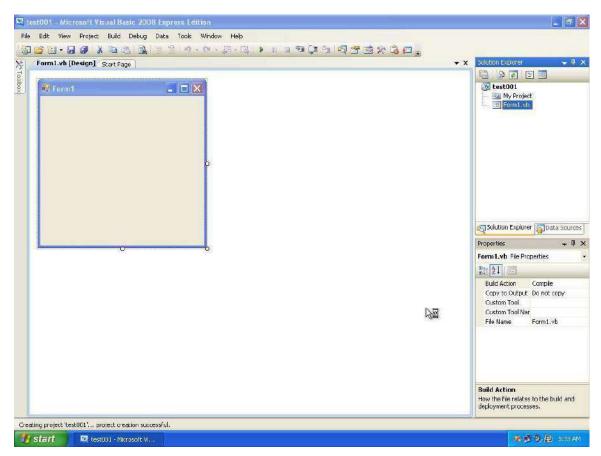
This is only a quick test... the next lessons explains what happens in detail... so for now this is the IDE (Integrated Development Environment) that you build your applications with...



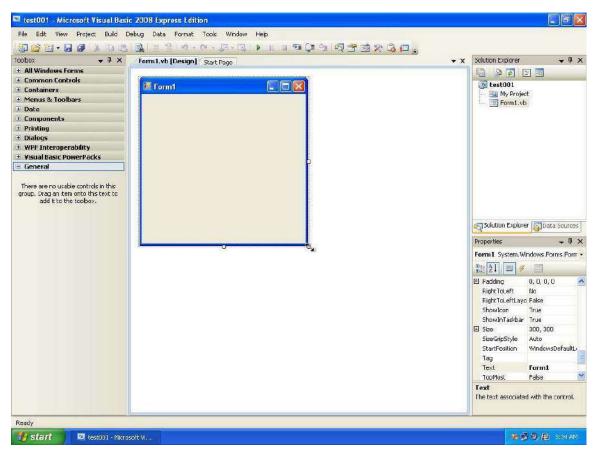
Select File->New Project

File Edit View Tode Window Help         Start Page         Start Page         View Tode Window	🗟 Start Page - Microsoft Visual Basic 2008 Express Edition			_ 7 🗙
Start Page       • x       jottom Expirer       • # x         Workert       Image: Start Page       • x       jottom Expirer       • # x         Recent Peoplets       Image: Perfect Peoplets       Image: Perfect Peoplets       Image: Perfect Peoplets       Image: Peoplets	File Edit View Tools Window Help			
Start Page       • x       jottom Expirer       • # x         Workert       Image: Start Page       • x       jottom Expirer       • # x         Recent Peoplets       Image: Perfect Peoplets       Image: Perfect Peoplets       Image: Perfect Peoplets       Image: Peoplets				
Image: Second Devices		* X	Solution Explorer	<b>▼</b> ₽ X
Image: Second Devices		•		
Image: Second Devices		nes F		
Percent Project     Visual Studio installed templates     Visual Studio Studio Studio Visual Studio Installed templates     Visual Studio Installed templates     Visual Studio Studio Studio Visual Studio Installed templates     Visual Studio Installed templates     Visual Studio Studio Visual Studio Installed templates     Visual Studio Studio Installed templates     Visual Studio Studio Studio Studio Visual Studio Installed Templates     Visual Studio Studio Studio Studio Studio Visual Studio S				
Center Supplicit         Wirdwos       Casa Likrary         Search Online       Tempidets         Search Online       Tempidets         Visua Posto       For creating an application with a Windows user interface (.NET framework 3.5)         What's New?       New:         Visual Basic Developer to       Or         Visual Basic Developer       Or         Visual Basic Developer       Or         Under Struke 2: Welcome to       Or         U				
Open: Promos A   Create: Promos A   Backling Started   Create:   Promos A   Create:   Promos A   Promos Pr	Recent Projects Visual Studio installed templates			
Open:       Pro         Create:       Pro         Search Online       Tempides         Create Your First Apple       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Learn VB       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Name:       test001         Downbad Addbinal Go       OK         Visual Basic Develope       OK         Visual Basic Develope       Image: Test001         Use Studio 2010       Test001         10-4 Episode 1: Working with the       Test001	Windows Gass Library WPF Browser Console			
Create:       Pr         Section Started       Search Online         Create from First Apple       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Visual Sector Lev       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Name:       Testion Comparison         Developer       Or Cance         Visual Basic Developer       Or Cance         Visual Basic Developer       Image: Cance	My Templates			
Create:       Pr         Section Started       Search Online         Create from First Apple       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Visual Sector Lev       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Name:       Testion Comparison         Developer       Or Cance         Visual Basic Developer       Or Cance         Visual Basic Developer       Image: Cance				
Getting started         Create Your First Apply         Leain VB         What's New?         Described Additional Co         Mame:         test001         OK         Cancel				
Getting Started         Create Your First Apply         Video Footure Tour         What's New?         Devribad Additional Co         Mre:         Create Your First Apply         Devribad Additional Co         Mission Footure         Visual Basic Developer         Visual Basic Developer         Visual Basic Developer         Visual Basic Control         Visual Studio 2010         Visual Studio 2010<				
Create //ur Fis: Apple Vides Feature Tour Learn VB Despiner Developer Lev Developer Lev Developer Visual Basic Developer Visual Basic Developer				
Video Footure Tour       A project for creating an application with a Windows user interface (.NET Framework 3.5)         Learn V8       What's New?         Deprimer Developer Len       Nme:         Developer       OK         Visual Basic Developer       OK         Developer       OK         Visual Basic Developer       OK         Visual Basic Developer       OK         Visual Basic Developer       OK         Developer       OK         Developer       OK         OK       Cancel         Visual Basic Developer       OK				
What's New?       Name:       test001         Download Additional       OK       Cancel         Visual Basic Developer       OK       Cancel <tr< th=""><th>Video Feature Tour A project for creating an application with a Windows user interface (.NET Framework 3.5)</th><th></th><th></th><th></th></tr<>	Video Feature Tour A project for creating an application with a Windows user interface (.NET Framework 3.5)			
Develoar Additional Comment       MSUB Partices       Visual Basic Developer         Visual Dasic Exploses It coullines       10-4 Episode 2: Welcome to       Visual Studio 2010       10-4 Episode 1: Working with the		-		
MSCN Fortine Visual Basic, Developer Visual Basic, Express ItCadlines 10-4 Episode 2: Welkome to Visual Studio 2010 10-4 Episode 1: Working with the		0		
Visual Basic Express Headines 10-4 Episode 2: Welcome to Visual Studio 2010 10-4 Episode 1: Working with the Visual Studio 2010 TO P VP	MSDN Forums OK	Cano	e	
	Visual Dasic Express Headlines 10-4 Episode 2: Welcome to Visual Studio 2010 10-4 Episode 1: Working with the Visual Studio 2010 CTP V/C			
		*		
🔣 start 🔤 start Page - Microsof 🛛 🥵 🖗 sico AM	🔐 start 🔤 start Page - Microsof		<b>*</b> \$0	👜 3:35 AM

Select Windows Forms, and give a name to your first application, then press OK.



Your application contains one window... you can modify it visually without doing any programming...



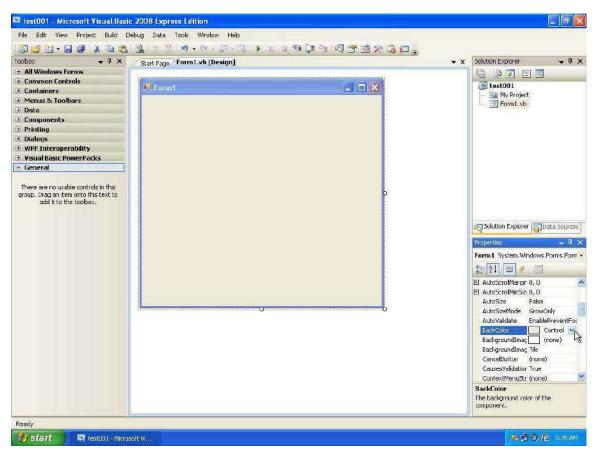
Put the mouse pointer over the small square on the corner, and drag the window's corner and see what happens ... you are resizing the window very easily...

E test001 (Running	g) - Microsoft Visual Ba	sic 2008 Express Edition						
		a Format Tools Window H						
Form1.vb [Design]		5 . G. B. C.		<b>₹</b> ⊴≈≈©∎, ∘∈	1 @ ₹ ▼ X	Solution Explorer		ŢХ
Form L.Vo [Design]					* *			* ^
🖶 Form1		Form1	3			Form1.vt	t	
						Solution Explor Properties Form1 System.W	-	φ×
						BERT E		i oni
						HelpButton	False MoControl False False (Default)	
Immediate Window					ŧΧ	Localizable	False	
						Location Locked MainMenuStrip MaxmizeBox	0, 0 False (none) True	
					2	Text The text associate		trol.
Ready				-L. []]	15, 15	] 고[개 43	15 × 409	
🎁 start 🔰 🔯	test001 (Running)	📰 Form1				*	<b>9 (p)</b> 3 3	H AM

To test this window, you hit the play button... the play button runs your application, to stop the application hit the stop button or close the application window...

🖳 test001 (Running) - Microsoft Visual Basic 2008 E	xpress Edition	- <b>a</b> 🗙
File Edit View Project Build Debug Data Format		
Form1.vb [Design]4		※ る ロ 、 ・ ス Solution Explorer ・ キス
Form L.vo [Design]a		
R Form1		ریک <b>test001</b> ها My Project Form1.vb
	Maximize       X     Close       Alt+F4	Solution Explorer Data sources
		Properties + II X
		Form1 System Windows Forms Form +
		B: 21 - 4
U		HelpSutton False
Immediate Window		Language (Default) Localizable False
		Control of the second of the
3		Text The text associated with the control.
Ready		🛄 15, 15 📰 435 × 409
🔠 start 🛛 💌 test001 (Running) 🗱 Form 1		🤕 🖉 🥹 🚇 3.54 AM

You can test the window and see it behaves like many XP forms or Vista forms...



Now stop the application and search for Back color property from the list on the right...

🖳 test001 - Microsoft Visual Bas	c 2008 Express Edition		_ 7 🗙
File Edit View Project Build I	Pebug Data Tools Window Help		
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	<u>A</u>  = 2 ■• • •	47320	
Tcobox - 7 x	Start Page Form1.vb [Design]	• X	Solution Explorer 🚽 🗣 🗙
+ All Windows Forms			
+ Common Controls	fair -		test001
E Containers	📅 Form1		- B My Project
Menus & Toolbars			Form1.vb
🗄 Data			
T Components			
• Printing			
🗄 Dialogs			
± WPF Interoperability			
Yisual Basic PowerPacks     General			
There are no usable controls in this group. Drag an term onto this text to add it to the toobox.			Surtician Web System
The second s			component.
Ready			
👸 start 🛛 🖾 testobi - Micro	soft V		🧶 🥵 🥹 👰 3.35 AM

Press the small arrow that appears, and select General as shown In the image above... then select a color that you like.

🖾 test001 - Microsoft Visual Basi	: 2008 Express Edition		- 6 🛛
File Edit View Project Buld D	ebug Data Tools Window Help		
1 🔄 😂 • 🖬 🖉 🕺 🖎 🖎	▲ = = = • • •		
Toobox - 7 x	Start Page Form1.vb [Design]*	• x	Solution Explorer 🛛 🖌 🐺 🗙
+ All Windows Forms	A sector age		
+ Common Controls			1 test001
- Containers	🔮 This is my first application 💦 📑 🗖 🔀		My Project
Menus & Toolbars			Form1.vb
🗄 Data			
Components			
1 Printing			
Dialogs			
WPF Interoperability     Visual Basic PowerPacks			
- General			
a deneror			
There are no usable controls in this group. Drag an tem onto this text to add it to the toobox.	P		
			Solution Explorer Data Sources
			Properties 🗢 🕂 🗙
			Form1 System. Windows Forms Form
			8:21 = # =
			ShowInTaskbar True A ShowInTaskbar True ShowInTaskbar Auto ShowInTaskbar Show
			TransparencyKe UseWaitCurson Felse WindowState Normal
			Text The text associated with the control.
Ready			
🥼 start 🛛 🛤 testool - Micros	soit V.,		🦔 💆 🧐 👰 SLOSIAM

Now search for the Text property and replace its value with anything you like ...

🖼 test001 (Runnin	ng) - Microsoft Visual Ba	sic 2008 Express Edition				_ 7 🗙
	sector sector subscription of the sector	ta Format Tools Window He		10		
		19.10.12.13.1	u 🛛 🕫 🕻			
Form1.vb [Design	Ja				× X Solution Explore	
6			S		test001	
SE This is my I	irst application	<b>_ D X</b>			My Pro	ject
					Form1	
	🔙 This i	is my first application	N			
			8			
					Solution Expl	orer 🛐 Data Sources
					Properties	≁ Ū ×
					Form1 System	Windows Forms Form +
					BE 21 💷	<b>#</b>
					E Padding	0, 0, 0, 0 🔥
					Right ToLeft	No
					Right ToLeftL: ShowIcon	True
					ShowInTaskb	
Immediate Window				 ¥ .	× E Size     SizeGripStyle	435, 409 Auto
					StartPosition	WindowsDefaultL
					Tag	
					Text TopMost	This is my first False
					Text	1 (13)
					The text association	ited with the control.
(\$.]				 	2	
Ready				 		
🎁 start 🔰	💌 text001 (Ronring)	醥 This is my first applica				🥵 🥘 👰 (3.36 AM

Run the application...

💌 test001 - Microsoft Visua	I Basic 2008 Ex	press Edition		X
File Edit View Project Bu	uld Debug Dat	a Format Tools Window Help		
and the second se	X Start Da	Contraction of the second se	-	<b>↓</b> ↓ ×
Tcobox + All Windows Forms	Start Pa	<sub>30</sub> ∕ Form1.vb [Design]	Solution Explorer	
🗄 Common Controls	- Cart	is is my first application	test001	
Containers     Menus & Toolbars			🗕 🔄 My Project	
E Data			Form1.vb	
🗄 Components				
Printing     Dialogs				
± WPF Interoperability				
🗄 ¥isual Basic PowerPacks				
🖻 General			2 🛛	
There are no usable controls in t	Save Project			
group. Drag an tem onto this tex add t to the toobox.		1		
	Location:	C:\Documents and Settings\mka\My Documents\Visual Studio 2006\Projects	e ]	
	Solution Name:	test001		Data Sources
				100
		Silve Can	rties	+ Q ×
1				dows Forms Form *
			8221 💷 🐔	<u>15</u>
				0, 0, 0, 0 🔥 📩
			RightToLeftLayo	
	(Province)			True
			ShowInTaskbar El Size	True 435, 409
			and the second se	433, 409 Auto
				WindowsDefaultL
			Tag Text	This is my first
				False
			Text	1.25
			The text associated	with the control.
Ready	- Microsoft Vi		a 1	<b>9</b> , 🖬 3.36 AM
	THEODEDIC W.T.			ALTER OFFICIAL

Very easy... still one last thing... you should save the application. Just Hit the Save all icon from the tool bar...and select save...

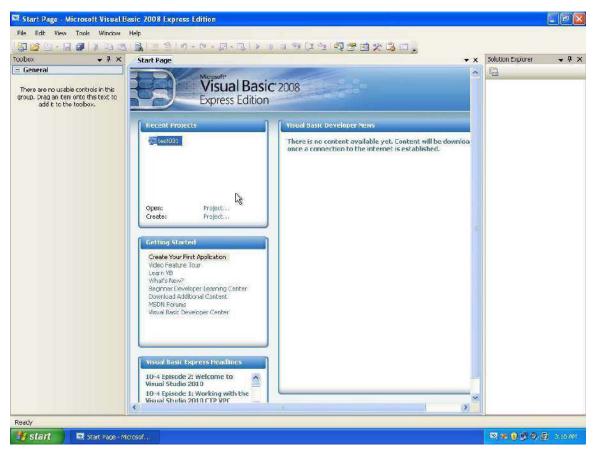
**Chapter 2: Understanding the IDE** 

## Understanding the IDE

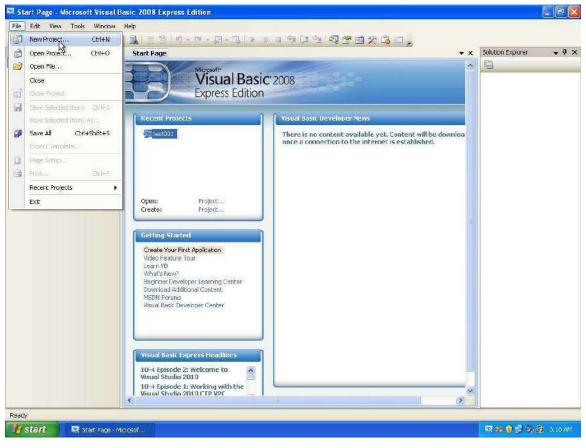
In this chapter, we will understand a little bit about the IDE, and what does it provide, so later on it will be easier for us to work on applications,



Open VB.NET



Now this is the IDE, in order to work with it, there should be an application, so we will create a new application in order to discover the IDE, and see what does it provide and how does it help us.



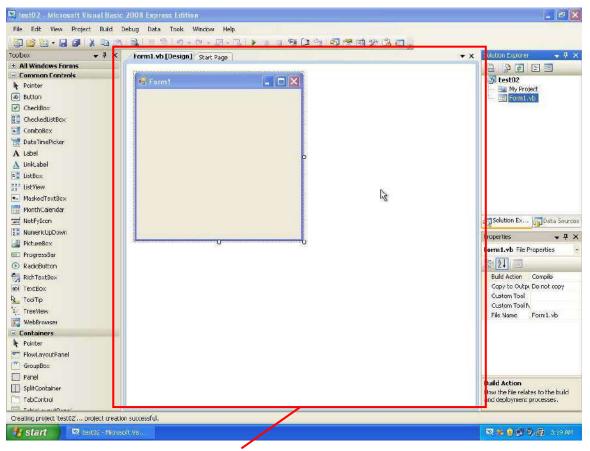
Open a new project

📧 Start Page - Microsoft Vis	val Basic 2008 Express Edition			×
File Edit View Tools Win				
	3 (1) 二 二 ヴ・ヴ・ワ・マ ト ロ 日 三 三 三 三 三 三 三 三 三 三 三 三 三 三 三 三 三 三	3.		
	Start Page	* X	Solution Explorer	• ₽ ×
🔄 General	Microsoft*	<u>^</u>	<u>_</u>	
There are no usable controls in t	New Project	?	3	
group. Drag an tem onto this tex add it to the toolbox.	Templates:			
	Visual Studio installed templates		7	
	Windows Application       Class Lbrary WFF       WFF       WPF Browser Application       Concole 			
	Name: WindowsApplication1		÷	
		DK Cantel	1	
	Visual Basic Express Headhines 10-4 Episode 2: Welcome to Visual Studio 2010 10-4 Episode 2: Working with the Visual Shufin 2010 CTP VPC	8		
😼 start 🔰 🖾 Start Pa	ge - Microsof		R # 0 3 0.	🔂 SIBAN

The new project dialog box appears, select windows forms application, and supply a name for your project

🖼 Start Page - Microsoft Vi	sual Basic 200	8 Express Edition				×
File Edit View Tools Wir	and the second second		A CONTRACT OF			
🔁 🧉 - 🔛 🗿  🗎		「ゆう・マ・ヨ・ジット」	= ? [] * ] * 🖓 🕾 🖄 🏹 🖧 🗂 🖕			
	₽ × Start	Page		* X	Solution Explorer	• 4 ×
🚍 General		Microsoft*		<u>^</u>	8	
There are no usable controls in t	New Project			?	3	
group. Drag an tem onto this tex add it to the toolbox.	Templates;					
	E CONTRACTOR OF THE OWNER	lio installed templates				
	Windows Forms A My Temple Search Onlin Templates	ites	wser Console lition Application		1	
	Name:	test02				
			- ale	Cancel	Ĵ	
	10- Vist	ual Basic Express Headlines 4 Episode 2: Welcome to al Studio 2010 4 Episode 1: Working with the ual Studio 2010 FTP VPF	And the ABE Transvert 3.5 has tons of new faature data-triven Web sites, working with SQL Server 2008 Windows applications, and much more. Microsoft Robotics Developer Studio 2006 has Fri, 12 Dec 2000 10:44:13 GMT - Read about d1 then features, improvements and tutorials available with th help you build coll robotics projects! Great benefits for registered Express users!	, building faste just release		
	Incard			100	91	
👘 start 🔰 🖾 Start Pa	age - Mcrosof				10 + 0 S 9.	🗊 3:16 AM
	annes ann an					Merican Constanting

Press OK



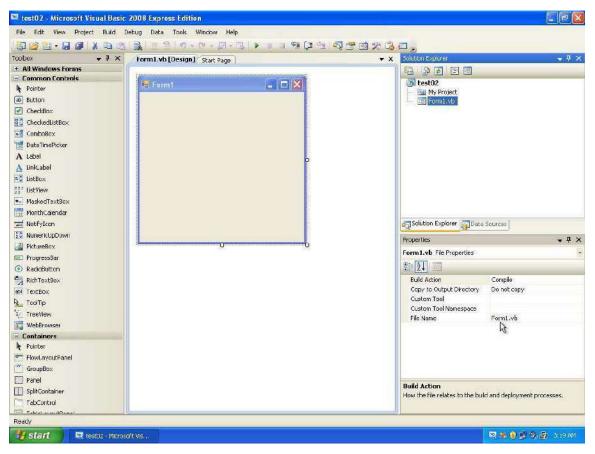
Now you see the IDE. The central part is your working area, there you write code, design the user interface, and do many other things. What you are seeing now is the how the user interface looks like.

File Edit View Project Build Debug Data Tools Window Help         Image: State Page         Toobox         + All Windows Forms         Common Controls         Pointer         Image: State Page         Image: State Page	and a state of the	<del>~</del> ₽ X
Toobox • + A × Form1.vb[Design] Start Page • • > + All Windows Forms - Common Controls • Pointer      Button     CheckBox	X Solution Exposes	★ # X
Toobox • 7 × + All Windows Forms - Common Controls • Pointer @ Button • CheckBox	X Solution Exposes	* # ×
+ All Windows Forms - Common Controls Pointer - Button CheckBox	test02	
Pointer     Button     CheckBox	test02	
Button     CheckBox	🖳 🔜 My Project	
CheckBox	Form1.vb	
		-
BC CheckedlistRox		
E ComboBox		
📷 DateTimePicker		
A Label		
A Link.abel		
📲 ListBox		
👯 ListYiew		
MaskodTextBox		
MonthCalendar		
Tari NotfyIccn	Solution Explorer Solution	Scuree
3 NumericupDown	Provide Hard	* <del>4</del> ×
PictureBox	Properties	• <del>+</del> ×
💷 ProgressBar	Form1.vb File Properties	1.12
RadicButton	8: 21	
nichTextBox	Bull Action	Compile
abi TextBox	Copy to Output Directory	Do not copy
Res ToolTip	Custom Tool	
Treeview	Custom Tool Namespace File Name	Form1.vb
WebBrowser	File Name	Pormi,vb
= Containers		
Pointer		
ThowLayoutPanel		
🚰 GroupBox		
Panel	Dull Lating	
III SplitContainer	Build Action How the file relates to the built	d and deployment processes
TabControl	and the reaction of the best	a nua subuchuran buca canada
The Tables and Danal		
Ready		
🛃 start 🔍 🛤 testoz - Microsoft Vs		🛤 🏶 🔮 🧶 🖗 i S 19 AM

Next part is the solution explorer. This part you can see the main files that your application is consisting of. You use it to quickly move from one part of the application to the other.

🖾 test02 - Microsoft Visual Basic	2008 Express Edition	
File Edit View Project Build De	abug Data Tools Window Help	
🔄 🗃 🖬 • 🖬 👰 🔥 🛍 🚳	▲目前は10・0・21・01 ● 11 目前目前に	ୟ <u>જ</u> ∞ × 3 ⊡ .
Toobox 🚽 🕈 🗙	Form 1.vb [Design] Start Page	👻 🗙 Solution Explorer - test02 🛛 👻 🗣 🗙
+ All Windows Forms		
E Common Controls	🛃 Form1 🔤 🗖 🗙	12 test02
h Pointer		My Project
ab Button		E Form1 vb
CheckBox		
🛐 CheckedListBox		
E ComboBox		
📷 DateTimePicker		
A Label	6	
A LinkLabel		
🗐 ListBox		
222 ListView		
#- MaskedTextBox		
🎬 MonthCalendar		
📷 NotfyIcon		Solution Explorer Sources
🔝 NumericUpDown		Properties 🚽 🖟 🗙
📓 PictureBox		
📖 ProgressBar		test02 Project Properties
📀 RadicButton		
📆 RichTextBox		Project File test02.vbprof-
abl TextBox		
🧏 TooTip		
ी TreeView		
🛐 WebBrowser		
- Containers		
Pointer		
FlowLayoutPanel		
(**) GroupBox		
Panel		Project File
III SplitContainer		The name of the file containing build, configuration, and other
TabControl		information about the project.
TEN TablelanenDasel		
Ready		
👘 start 🛛 📓 testüz - Microso	t vis	🗷 🤗 🔮 🔗 👜 3:19 AM

Next is the properties window. This window changes according to what you are doing right now. For example, whenever you select a file from the Solution Explorer window, it changes itself to show you only the properties related to that specific file. It works the same way with the graphical user interface in the central area (the GUI).



Select Form1.vb from within the solution explorer window by clicking it once and see how the properties change to view the relevant information. Try to select My project again by clicking it one and see how the properties changes again.

File Edit View Project Build D	ebug Data Tools	Window Help			
🗊 📻 🔛 • 🔙 🌒 认 📭 🖄		· (* · 四·马 ) • · · · · · · · · · · · · · · · · · ·		3 🖂 🚬	
obox 🗸 🕈 🗙		b [Design] Start Page	•)	Here and a second	<b>↓</b> ₽ ×
General	The second s				
	Application Comple Debug References Sectings Signing My Extoneions Securty Publish	Assembly name: Sector Application type: Windows Forms Application Scatup form: Form1 Assembly Information View UAC 5e Enable application framework Windows application framework Windows application framework properties Prable XP Visual styles Make single instance application Save My. Settings on Shutdown Authentication mode: Windows Shutdown mode: When startup form closes Solach screen:	Root name: test02 Icon: (Default Ico ttings	Image: Solution Explorer         Properties         Image: Solution Explorer	→ # ×
		(None)	~		
eady					

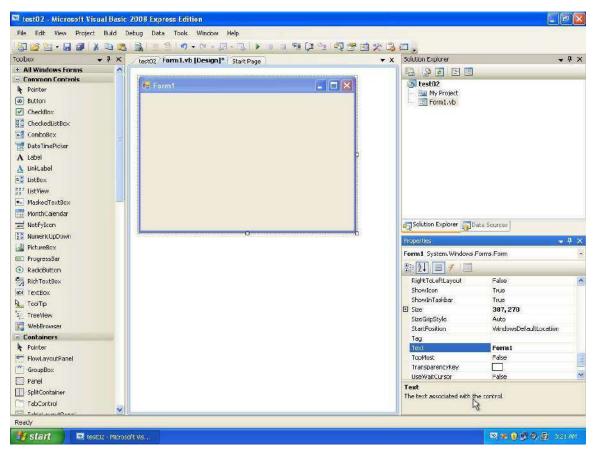
Now as I said before, the Solution explorer helps you navigate your application quickly. So now try double clicking My Project to see something similar to the above. Now you can modify your application. This is just an example, so don't worry about the details of all these options we will come to this later.

🖾 test02 - Microsoft Visual Basic 2008 Express Edition		
File Edit View Project Build Debug Data Format Tools Window Help		
	<b>N 🖷 🖄 🕫 🗋 🗖</b>	
Tcobex + 7 × testo2 Form1.vb [Design] Start Page		• ª ×
+ All Windows Forms		
	and the second statement of the second statements	
Pointer	Lest02	
(a) Button		
CheckBox	Forma.vb	
E CheckedListBex		
E ComboBox		
Data TimePicker		
A Label		
A LinkLabel		
Ta ListBox		
ar ListYiew		
- MaskodTextBox		
MonthCalendar		
w Notfylcon	Solution Explorer Data Sources	
20 NumericupDown		
PictureBox	Properties	• <del>4</del> ×
ProgressBar	Form1 System. Windows Forms Form	
RadieButton		
RichTextBox	Padding     O, 0, 0, 0	*
at rextBox	RightToLeft No	100
	Right ToLoft Layout False	
Res ToolTip	ShowIcon True	
3, TreeView	ShowInTaskbar True	
C WebBrowser	E Size 300, 300	
E Containers	SizeGripStyle Auto StartPosition WindowsDefaultLocation	
Revenue and Antonio and Antoni		
FlowLayou£Panel	Tag Text Form1	
C GroupBox	TooMost False	~
Denel	Text	
SplitContainer	The text associated with the control.	
TabControl		
Ready		_
👭 start 🔍 🔍 testiz - Microsoft Vis.,	1	20 AM

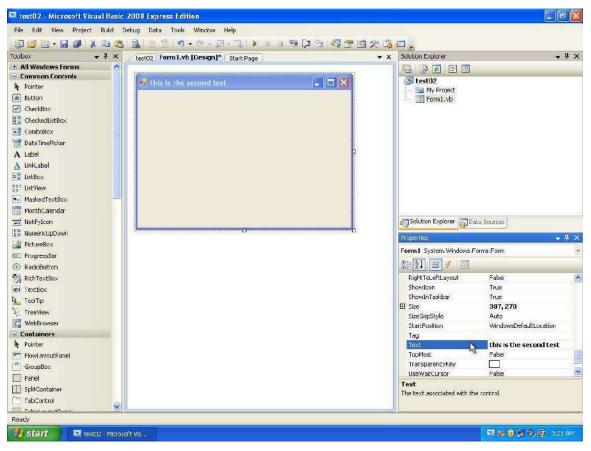
Now double click on Form1.vb/you see the GUI again. Now let us work a little with these toys. Right now, our application has one window. We want to say, change its size. To do so, you drag one of the white boxes.

Image: Second	T ⊠ X G L , ▼ X Solution Explorer		
All Windows Forms			
Common Controls			4 3
		<u>a</u>	
	test02		
Button	My Project		
CheckBox	<u>so</u> romit.vo		
CheckedListBox			
E ComboBox			
Data TimePicker			
A Label			
, Link abel			
ListBox			
² ListYiew			
MaskedTextBox			
MonthCalendar			
Notfylcon	Solution Explorer	Data Secure	
MumericupDown	Capacitan Colorer 1		
PictureBox	Properties		4
ProgressBar	Form1 System.Windo	ws Forms Form	
) RadicButton	2 AL I /	5	
RichTextBox	E Padding	0,0,0	
TextBox	RightToLeft	No	
TooTp	Right ToLeftLayout	False	
TreeView	ShowIcon	Truc	
WebBrowser	ShowInTaskbar	True	
Containers	1 Size	300, 300	
- Pointer	SizeGripStyle StartPosition	Auto WindowsDefaultLocation	
FlowLayoutPanel	Tag	Williabwsberdaitebcacion	
GroupBox	Text	Form1	
Panel	TopMost	False	
] SplitContainer	Text		
TabControl	The text associated w	th the control	

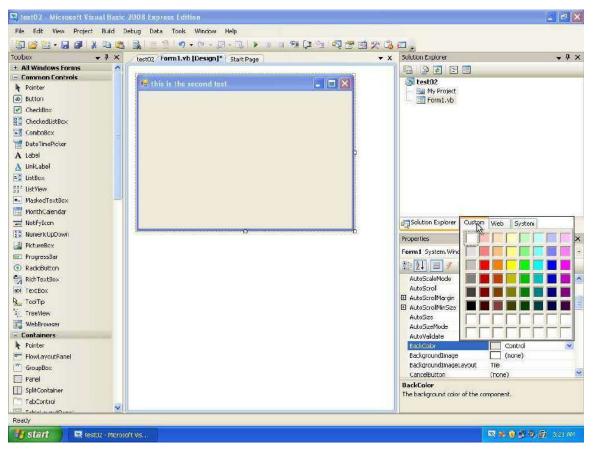
Notice also how does the properties window change to help you work on what you have just selected (here the window is selected so the properties window is showing its properties).



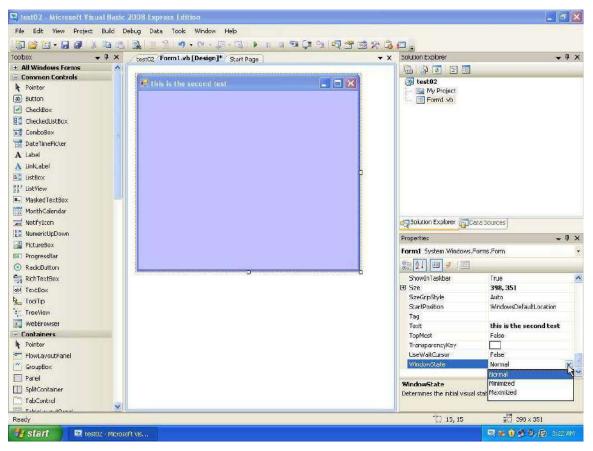
Now let us work with the properties window. Each object (window, or file or control) within the IDE has a number of properties that affects its behavior, and/or appearance. For example, if you search for a property called Text and change it, you can modify the title that appears on the window.



Try entering this is the second test and then press enter. You can see how does that affects the window you are designing.



Now let us work with another property, which is BackColor, try to choose a color and see how does that affects the window's color.



And so on, there are many other properties, we will learn just some of these that are commonly needed.

cobox + + × Start Page fact(2 Form1.vb [Design]*	▼ X	Solution Explorer	<b>~</b> 7
Stork Page         test02         Form1.vb [Design]*           * All Windows Forms         •			
Pointer         Button         Check Op:         Check Op:         Check Op:         Cont         Matrix Component:         Dato         Raises an event when the use clcis it.         ListBox         ListBox         MatwolfextBox         MonthCalendar         MothCalendar         NotFCircon         Numerit(pDown		Solution Explorer	Vata Scurcos
3 PictureBox ■ ProgressBar		orm1 System.Windows.	Forms Form
9 RadicButton		the second se	True
	b	ShowInTaskbar	TIME
RadcButton     RichTextBox	0	3 Size	398, 351
RadicButton	0	3 Size SizeGripStyle	398, 351 Auto
D RadcButton	0	3 Size SizeGripStyle StartPositian	398, 351
RadicButton     RichTextBox     IntrextBox	0	3 Size SizeGripStyle StartPosition Tog	398, 351 Auto WindowsDefaultLocation
RadicButton RichTextBox RichTextBox TextBox TooTp TrevNew	0	3 Size SizeGitpStyle StartPositien Tag Text	398, 351 Auto
RadicButton       RichTextBox       Di TextBox       Di TextBox       TooTTp       TraeVlew       WebBrowser	0	3 Size SizeGripStyle StartPosition Tog	398, 351 Auto WindowsDefaultLocation this is the second test
RadcButton     RadcButton     RichTextBox     TextBox     TodTp     TrevtRew     WebBrowser     Containers     Pointer	0	3 Size SizeGripStyle StartPosition Tog Text TopMost	398, 351 Auto WindowsDefaultLocation this is the second test False False
AcideButton       RichTextBox       TextBox       ToolTip       TreeView       WebBrowser       Containers       Fourier       FlowLayoutPanel	0	3 Size Size Size Size Size Size Size Size	398, 351 Auto WindowsDefaultLocation this is the second test False
RadcButton       RichTextBox       RichTextBox       TextBox       TooTip       TreeNew       WebFowser       Containers       Pointer       PlowLayoutPanel       "GroupBox		3 Size SizeGripStyle StartPosition Tog Text TopMost TransparencyKey UseWaltCursor Windowstate	398, 351 Auto WindowsDefaultLocation this is the second test False False
RadcButton     RedFaxBox     RichTextBox     TextBox     TooTp     Trevelwew     WebBrowser     Containers     Pointer     FowLayourPanel		<ul> <li>SizeGirpStyle</li> <li>SizeGirpStyle</li> <li>StartPosition</li> <li>Tog</li> <li>Text</li> <li>TopMost</li> <li>TransparencyKey</li> <li>UseWaitCursor</li> </ul>	398, 351 Auto WindowsDofaultLocation this is the second test False False Normal

Now we come to the left side which we forget to tell you about. This one helps you adding controls to your window. For now click on the command button, and then draw it on the window (the command button is used to trigger some kind of actions or processing, more details on that later).

	Debug Data Format Tools Window Help		
	ă  <b>3</b>  ≘≌ ¶•∀•₽•₽ ▶∥ ¤ %0		
rcobox 🗢 🕈 🗧	Start Page test02 Form1.vb [Design]*	▼ X Solution Explorer	<b>▲</b> ù
	N [		
- Common Controls	this is the second test	Test02	
Pointer		My Project	
ab Button		i 💼 Form1.vb	
CheckBoz			
CheckedListBax			
ComboBax			
📑 DateTimePicker			
A Label			
📐 LinkLabel			
🗧 ListBox		P	
ListView			
- MaskedTextBax			
MonthCalendar			
Nothylcon		Solution Explorer	Data Sources
討 NotfyIcon ③ NumericUcOown		Solution Explorer	
NumericupOown		Properties	• <b>4</b>
NumericUpDown PictureBox			• <b>4</b>
NumericUpOown PictureBox ProgressBar		Properties Form1 System.Wind	₹ ₽ cws.Forms.Form
NumericUpOown PictureBox PrograssBar RadicButton		Properties	• P ovs Forms Form
S NumericUpOown B PictureBox D ProgressBar D RadicButton RichTextBox		Properties Form 1 System Wind	₹ ₽ cws.Forms.Form
NumericUpOown PictureBox ProgressSar RadicButton RichTextBox PrexEox	U	Properties Form 1 System. Who ShowinTaskbar	v ₽ cv/s Forms Form True
MumerkUpOown UptureBox ProgressBar RadjessBar RadjeSbutton RichTextBox To TextBox To TextBox	J	Properties Form 1 System Wind ShowinTaskbar ShowinTaskbar SizeGirjGityle StarFosition	<ul> <li></li></ul>
MumerkUpOown  ProgressBar  RadioButton  RadioButton  RaihTextBox  TooTip  TextBox  TooTip		Properties Form1 System.Wind ShowinTaskbar SizeGsipStyle StartPosition Tog	↓ ↓ cws Forms Form Inue 398,351 Auto WindowsDefaultLocation
NumericlyDown  PictureBox  ProgressBor  RadicButton  RichTostBox  TostBox  TostBox  TostBox  VextBox  WebBrowser		Propertiles Form1 System.Wind ShowinTaskbar: ■ Size SizeSizpStyle StarFosition Tag Text	↓      ↓
NumerkUpOowin  PictureBox  ProgressBar  RadcButton  RichTextBox  TextBox  TooTip  TrevNew  WebBrowser  Containers	J	Properties Promi System Wind 2: ↓ ● ✓ ShowinTaskbar ⊡ Size SizeGiipStyle StarFosition Tag Text TagMost:	↓ ↓ cws Forms Form Inue 398,351 Auto WindowsDefaultLocation
NumerkUpDown  PictureBox  ProgressBar  RadicButton RichTextBox DirtextBox DirtextBox TooTip TooTip TreeView WebBrowser Containers Pointer Pointer	J	Properties Form1 System.Wind ShowInTaskbar SizeGsipStyle SizeGsipStyle Sizefrosition Tag Text TapMost TransparencyKey	
MumericupDown  ProgressBar  RadicButton  RadicButton  RichTextBox  ToolTp  ToolTp  ToolTp  Containers  Pointer  Pointer  Pointer  Pointer  Pointen		Properties Promi System Wind 2: ↓ ● ✓ ShowinTaskbar ⊡ Size SizeGiipStyle StarFosition Tag Text TagMost:	↓      ↓
MumericupDown  PictureBox  ProgressBar  RachTextBox  ToolTp  ToolTp  ToolTp  Containers  Pointer  Pointer Pointer  Pointer Poi		Propertiles Form1 System Wind ProvinTaskbar ShowinTaskbar SizeGipityle StorFosition Tag Text TapMost Transparencykey UseWaltCursor	
NumerkUpDowin  PictureBox  ProgressBar  RadcButton  RadcButton  RadcButton  TextBox  TooTp  TreeView  WebBrowcer  Containers  Pointer  FlowLayoutPanel  Panel		Propertiles Form1 System Wind ProvinTaskbar ShowinTaskbar SizeGipityle StorFosition Tag Text TapMost Transparencykey UseWaltCursor	
NumerkUpDowin  PictureBox  ProgressBar  Radcblutton  RichTextBox  TooTp  TooTp  TraeView  WebBrowcer  Containers  FlowLayouPanel  Solitor  FlowLayouPanel  SolitContainer		Properties Form1 System Wind ShowInTaddar SizeGitjGityle SizeGitjGityle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGittyle Siz	
NumericupDowin       PichureBox       ProgressBar       RadieButton       RichTextBox       TooTp       TooTp       TreeView       WebBrowser       Containers       FlowLayoutPanel       GroupBox       Pail       SplitContainer       Zabintorial	Γ	Properties Form1 System Wind ShowInTaddar SizeGitjGityle SizeGitjGityle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGitGittyle SizeGittyle Siz	✓      ✓

By pressing the mouse button continuously, you can specify the dimensions of that control. After that release the mouse button.

the second	Debug Data Format Tools Window Help		
🔄 🚅 · 🔒 🕔 🔥 🖬 🖄	. <u>}</u> = = <b>! • • • • </b>	<b>4</b> 2 2 2 2 5 0 .	
Toobox 🔹 🕈 🗙	Start Page test02 Form1.vb [Design]*	▼ X Solution Explorer	<b>•</b> 1
🛨 All Windows Forms 🛛 🔥			
- Common Controls	🔛 this is the second test	test02	
N Pointer		📃 🔛 🔤 My Project	
ab Button	9	🦾 📰 Form1.vb	
🗹 CheckBoz	Buttonl		
😭 CheckedListBox	<u>¢</u> ¢		
📑 ComboBax			
📑 DateTimePicker			
A Label			
📐 LinkLabəl			
😋 ListBox			
🚦 ListYiew			
🖣 MaskedTextBax 📒			
📅 MonthCalendar			
📷 NotfyIcon		Salution Explorer	Data Sources
😂 NumericUpDown		Properties	• 1
📓 PictureBox		a construction of the	
📰 ProgressBar		Button1 System.Wind	
RadicButton			
RichTextBox		🗵 Maximum5ize	0,0
IN TextBox		E MinimumSiaa	0,0
🖕 ToolTip		Modfiers	Friend
TreeView		Padding RightTaLeft	0, 0, 0, 0 No
WebBrowser		E Size	117, 31
Containers		TabIndex	0
Pointer		TabStop	True
FlowLayoutPanel		Тад	
SroupBox		Text	Button 1
		TextAlign	MiddleCenter
Panel		Text The text associated with	h the control
Panel SplitContainer			
Contraction of the second s		The text associated with	

You can see how does your window look like. You can move the button by dragging it, or you can resize it using the white boxes. Also check out the properties window that shows you the properties of this button.

Image: Start Page     Image: Start Page       Image: Start Page     Start Page	▼ X Solution Explorer	•
Il Windows Forms	test02	3
Pointer Button CheckBox Button Button Button	test02	
Button CheckBoz ButtonL	🚽 🖳 🔜 My Project	
CheckBox Buttond		
DALONA	🦾 📰 Form1.vb	
ComboBox		
DateTimePicker		
Label		
LinkLabel		
ListBox	P	
ListYiew		
MaskedTextBox		
MonthCalendar		
Notrylcon	Solution Explorer	Data Services
NumericUpDown	Commenter of	
PictureBox	Properties	• ·
ProgressBan	Form1 System, Windo	As Forms Form
RadicButton		
RichTextBox	E Padding	0,0,0
TextBox	RightToLeft	No
ToolTip	RightToLeftLayout	Falso
TreeView	ShowIcon	Truc
	ShowInTaskbar	True
WebBrowser	1 Size	390, 351
ontainers	SizeGripStyle	Auto
Pointer	StartPosition	WindowsDefaultLocation
FlowLayoutPanel	Tag Text	this is the second test
GroupBox	TooMost	False
Panel	Test	
SplitContainer	The text associated wi	h the control
TabControl		

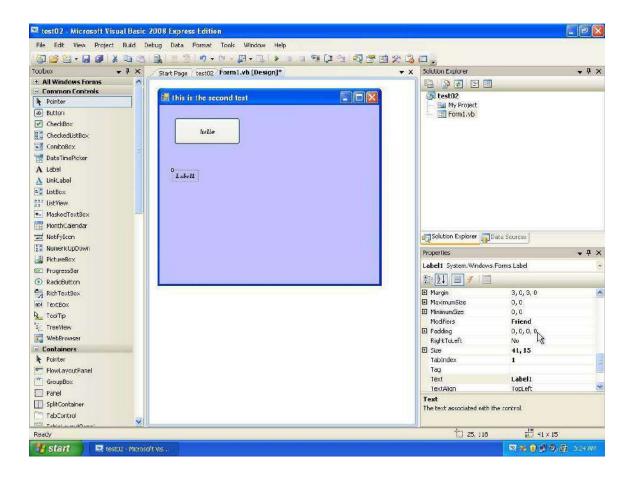
Now click on any empty space on your form to select it, and see how the properties window shows details about the selected object.

3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		<b>- - - -</b>
+ All Windows Forms		
- Common Controls		
Pointer	Extension Contension Contensi Contension Contension Contension Contension Contension Con	
a) Button	Form1.vb	
CheckBox		
CheckedListBox		
ComboBox		
DataTimePicker		
A Label		
A LinkLabel		
 =0 ListBox		
1 ListYiew		
MonthCalendar		
NotfyIccn	Solution Explorer	
20 NumericupOpwin		
A PictureBox	PAPEAR AND	<b>4</b>
🕮 ProgressBar	Button 1 System, Windows, Forms, Button	
RadicButton		
RichTextBox	E MaximumSize 0, 0	1
abi TextBox	🕒 MinimumSiaa 0, 0	
🐛 ToolTip	Madfiers Friend	
TreeView	E Fadding 0, 0, 0, 0	
WebBrowser	RightTaLeft No	
Containers	Tabindex 0	
🔭 Painter	TabStop True	
FlowLayoutPanel	Тад	
GroupBox	text hello	Y
Panel		
SplitContainer	Text The text associated with the control	
TabControl	The text associated with the control	

Now select the button again. You remember that when we wanted to change window title we modified the Text value for our window. The same is true for the button, and many other controls, so change the button's Text property to hello and see how does the GUI changes as well.

ceko 🗢 🗸		✓ X Solution Explorer		• 4 3
	Start Page test02 / Form1.vb [Design]*		-	• • •
- Common Controls				
Pointer	🖶 this is the second test	. 🗆 🔀 🥵 test02		
ab Button		My Project		
CheckBoz	9 <del>00</del>			
	c lutio -			
= ComboBax				
DateTimePicker				
A Label				
	+ <b>A</b>			
ListBox	<b>A</b>			
ListView				
- MaskedTextBox				
Education day				
MonthCalendar		The Solution Fundament	The second second	
NotfyIcon			Data Scurces	
NotfyIcon		Properties	Data Sources	• 4
<ul> <li>NotfyIcon</li> <li>NumericUpDown</li> <li>PictureBox</li> </ul>		A		• •
NotfyIcon NumericUpDown PictureBox ProgressBar		Properties Button1 System, Wind	lows.Forms.Button	• ¢
Notfylcon NumericUpDowin PictureBox ProgressBar RadicButton		Properties Button1 System.Who State 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	lows Forms Button	• 4
Motfylcon NumericupDowin PictureBox ProgressBar RadioButton RadioButton		Properties Button 1 System. Wind State 1 A Constraints of the State 1 Maximum Size	lows Forms Button	<b>→ 4</b>
w Notfylcon NumerklyDown ProgressBar RadcButton RadcButton RichToxtBox Di TextBox		Properties Button1 System.Who State 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	lows Forms Button	• 4
MotfyIcen Propression Rectures and Rectures and Rectur		Properties Button1 System.Wind 2015 24 11 21 21 21 21 21 21 21 21 21 21 21 21	lows Forms Button	• 4
Motfylcen MumerkupCown PictureBox ProgressBar RadcButton RadcButton RichTextBox TextBox TextBox Control TextBox Control TextBox		Properties Button1 System.Wind Star [] [] [] [] [] [] [] MaxmunSize [] MinimunSize [] MinimunSize Modifiers	oves Forms Button 0, 0 0, 0 Friend	• 4
Motfylcen MumerkupOown ProgressBar ProgressBar RachButton RachButton RichTextBox Di TextBox CoTp TeavIew WebBrowser		Properties Button1 System.Wind State MainumSize MinimumSize Medifies Pradding RightTaLoft Size	oves Forms Button 0, 0 0, 0 Friend 0, 0, 0 No 117, 47	• 7
Motfylcon NumerkUpDown PichueBox ProgressBar RadicButton RadicButton RichTextBox TeoTp TracVlew WebBrowcear Containers		Properties Button 1 System Wind Star [] [] [] [] [] [] [] [] [] MaxmunSize [] MaxmunSize [] MininumSize [] Madfiers [] Padding RightToLeft [] Size Tabindex	0xxs Forms Dutton 0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0	• 7
Motfylcon Notfylcon Progresslar RadcButon RadcButon RichToxbox TooTp ToreView WebBrowser Containers Pointer		Properties Button1 System Wind MastrumSize MinimumSize Modifiers Produce RightToLeft Size Tab2hdex Tab2hop	oves Forms Button 0, 0 0, 0 Friend 0, 0, 0 No 117, 47	• 4
MotfyIcen NumerkUpDown ProgressBar ProgressBar RadcButten RadcButten RichTostBox RichTostBox RichTostBox Containers Pointer Pointer PowtayoutPanel		Properties Button1 System.Wind Bit A B F F MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize Size Tabindex Tabitop Tag	oves Forms Dutton 0, 0 0, 0 Friend 0, 0, 0 No 117, 47 0 True	• 7
MotfyIcen NumerkUpDown ProgressBar ProgressBar RadcButton RadcButton RichToxtBox RichToxtBox TooTTp TextBox WebBrowcear Containers Painter FlowtLayoutPanel GroupDox:		Properties Button1 System.Wind Bill NaxmunSize Madfiers Prodding RightToLoft Gize Tabindex Tabindex Tabindex Tag Text	oves Forms Button 0, 0 0, 0 Friend 0, 0, 0 No 117, 47 0 True hello	• 7
MotfyIcen MumerkUpCown ProgressBar ProgressBar RadeButten RadeButten RadeButten TextBox RichTextBox Continees PrevNew Pointer Pointer PownaroucPanel Panel Panel Panel		Properties Button 1 System Wind State (A) (B) (A) MaxmunSize MinimumSize Minim	oves Forms Dutton 0, 0 0, 0 Friend 0, 0, 0 No 117, 47 0 True	• 7
MotfyIcen NumerkUpDown ProgressBar ProgressBar RadcButton RadcButton RichToxtBox RichToxtBox TooTTp TextBox WebBrowcear Containers Painter FlowtLayoutPanel GroupDox:		Properties Button1 System.Wind Bill NaxmunSize Madfiers Prodding RightToLoft Gize Tabindex Tabindex Tabindex Tag Text	lows Forms Button 0, 0 0, 0 Friend 0, 0, 0 No 117, 47 0 True hello MiddleCenter	• 7

Next place a LABEL on the form. The Labels are used to display text information



obbx     - 7 ×     Start Page     Form1.vb [Design]*       All Windows Forms     -     -     -       Common Controls     -     -     -       Depter     -     -     -	▼ X Solution Explorer		
Common Controls			• 4 >
Lommon Lontrols		<u>e</u>	
THIS E THE SECOND LEFT	test02	1	
	- 🔜 My Project		
Button     CheckBox	🦾 📰 Form1.vb		
CheckodustBox			
ComboBox			
🖞 DateTimePicker			
A Label D			
Lakett			
🖞 ListBax			
f ListYiew			
- MaskodTextBox			
MonthCalendar			
🖬 NotfyIcon	Solution Explorer	Data Sources	
C NumericucOown	Contraction of the Contraction o		
MumerkUpDown     PictureBox	Properties		• Ф
📓 PictureBox	Properties L <b>abel1</b> System, Window	vs.Forms.Label	<b>•</b> ₽
in ProgressBar	Label1 System.Window		* <b>4</b>
I ProgressBar D ProgressBar D RadicButton	Label1 System.Window	1	• 4
ProgressBar     PragressBar     RadcButton     RichTextBox	Label1 System.Window	3, 0, 3, 0	• 4
Image: ProgressBar       D       RadcButton       Image: RichTextBox       Image: Pottbox	Lebel1 System.Window State of the system of the system I Margin	1	• 4
PirtureBox       ProgressBar       RadcButton       RichTextBox       Di TextBox       La TeoTTp	Label1 System, Window State And State And Sta	3, 0, 3, 0 0, 0 0, 0 Friend	• <b>ग</b>
ProgressBar  ProgressBar  RachColution  RichTextBox  TextBox  TextBox  TextBox  TextBox  TextBox	Lebel1 System, Window Margin Margin MaximumSize MinimumSize Medifiers Podding	3, 0, 3, 0 0, 0 0, 0 <b>Friend</b> 0, 0, 0, 0	• <b></b>
PictureBox     Image: State	Lebeli System, Window Margin MaximumSize MinimumSize Modifies Podding RightToLeft	3, 0, 3, 0 0, 0 0, 0 Friend 0, 0, 0, 0 No	- <b>4</b>
PictureBox       ProgressBar       RadicButton       RichToxtbox       Di TextBox       TaolTip       Travitew       WebBrowser       Containers	Label1 System, Window Margin Margin MaximumSize MinimumSize Modifiers Padding RightToLoft Size	3, 0, 3, 0 0, 0 0, 0 Friend 0, 0, 0, 0 No 41, 15	<del>•</del> ₽
PictureBox ProgressBar RadicButton RadicButton RichTextBox TaolTp TaolTp TaolTp Containers Pointer Pointer	Lebeli System, Wndor Margin MaximumSize Modificis Padding RightColeft Size Tabindex	3, 0, 3, 0 0, 0 0, 0 Friend 0, 0, 0, 0 No	÷ 7
PictureBox ProgressBar Radcbutton Radcbutton RichTextDox RichTextDox TextDox TextDox TextNew WebBrowser Containers Painter PlowLavoutPanel	Lebeli System, Windor Wargin MaximumSize MinimumSize MinimumSize Modifiers Podding RightToLeft Size TabIndex Tag	3, 0, 3, 0 0, 0 0, 0 Filend 0, 0, 0, 0 No 41, 15 1	* 1
PictureBox ProgressBar Radicbutton RichTextBox TextBox TextBox TextBox TextBox Forture For	Lebeli System, Wndor Margin MaximumSize Modificis Padding RightColeft Size TabIndex	3, 0, 3, 0 0, 0 0, 0 Friend 0, 0, 0, 0 No 41, 15 1 Label1	÷ 4
PictureBox       ProgressBar       RacKebutton       RacKebutton       RichestBox       It TextBox       It TextBox       TooTp       TreeView       WebBrowser       Continees       Pointer       FlowLayoutPanel       GroupEox       Panel	Lebeli System, Window Margin MaximumSize ModimumSize Modifies Padding Right ToLeft Size Tabindex Tag Text	3, 0, 3, 0 0, 0 0, 0 Filend 0, 0, 0, 0 No 41, 15 1	• 4
PictureBox ProgressBar Radicbutton RichTextBox TextBox TextBox TextBox TextBox Forture For	Lebeli System, Wndor Margin Margin Margin Modifiers Podding Right ToLeft Size Tabindex Tag Text TextAlgn	3, 0, 3, 0 0, 0 0, 0 Friend 0, 0, 0, 0 No 41, 15 1 Label1 TocLeft	• 4

Now change the label to view the message: press the button

	Debug Data Format Tools Window Help		
		· · · · · · · · · · · · · · · · · · ·	
obox 🗸 🗘 🗸	Start Page test02 Form1.vb [Design]*	★ X Solution Explorer	<b>↓</b> ₽
All Windows Forms			
Common Controls	🖁 this is the second test	E K test02	2
Pointer		My Project	
ab Button		🔚 Form1.vb	
CheckBox			
🗘 CheckedListBox			
📱 ComboBox			
🕈 DateTimeFicker	00		
A Label			
LinkLabel	Press the button hefe		
ListBox	d <del>6</del>		
<sup>3</sup> ListView			
MaskedTextBox			
MonthCalendar			
NotfyIcan		Solution Explorer	lata Spurces
NumericUpDown		the second se	
PictureBox		Properties	÷ û
		Properties Button1 System Window	
PittureBox		Button1 System Window	
- ProgressBar ) RadicButton		Button1 System Window	s.Forms.Button
ProgressBar ) RadicDutton ; RichTextBox		Button1 System Window Siz 2.1 III / III El MaximumSize	s.Forms.Button 0, 0
ProgressBar ) RadicButton § RichTextBox ) TextBox		Button1 System Window	s.Forms.Button
ProgressBar ) RadicDutton ] RichTextBox ] ToxtBox _ TooTip		Button1 System Window இ≣ Q1 III ≠ III II MaximumSize II MinimumSize	s Forms Button 0, 0 0, 0
ProgressBar RadicButton RichTextBox ToxtBox TooTip TreeView		Button1 System Window 응발 Q1 III 중 III 면 MaximumSize 면 MirimumSize Modifiers	s.Forms.Button 0, 0 0, 0 Friend
ProgressBar Production RichTextBox ToxtBox ToxTop ToroVip Webbrowser		Button1 System Window ﷺ ▲ ↓ Ⅲ 参 ↓ Ⅲ Ⅲ MaximumSize ④ MirimumSize Modifiers ④ Padding PightToLeft ④ Size	s,Forms,Button 0,0 0,0 Friend 0,0,0,0 No 117,47
ProgressBar ) RadicDutton RichTextBox ) ToxEox , ToxTip TreeView Webbrowser Containers		Button1 System Window S∷ 21 III ≠ III MaximumSize Modifiers RightToLeft Size TabIndex	s.Forms.Button 0,0 0,0 Friend 0,0,0,0 No 117,47 0
ProgressBar ) RadieDutton 4 RichTextBox 1 ToxtBox 1 ToxtBox 1 ToxtBox 1 ToeView WebBrowser Containers Pointer		Button1 System Window Bi (2) III # III # III III MaximumSize Modifiers II Padding RightToLeft III Size TabIndox TabStop	s,Forms,Button 0,0 0,0 Friend 0,0,0,0 No 117,47
ProgressBar ProgressBar RichTextBox ToxtBox ToxtBox ToxtBox TooView Webbrowser <b>Containers</b> Pointer FlowLayoutPanel		Buttoni System Window Bi MaximumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize MinimumSize Modifiers Bi Padding RightToLeft Size TabIndex TabStop Tag	s, Forms, Button 0, 0 5, 0 <b>Friend</b> 0, 0, 0, 0 No 117, 47 0 True
ProgressBar ) RadcDutton 3 RchTextBox 1 TextBox 1 TextBox 1 TextBox 1 TereView Webbrowser Containers Pointor Pointor Pointor 1 HowLayoutPanel GroupEox		Button1 System Window ﷺ 실↓ Ⅲ 参 /Ⅲ ℍ MaximumSize ℍ MinimumSize ℍ MaximumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MaximumSize ℍ MinimumSize ℍ MinimumSize	s Forms Button 0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 True here
ProgressBar ) RadicDutton 3 RichTextBox 1 TostBox 1 TostBox 1 TostBox 1 TostBox 2 ToolTip Treaview Webbrowser Containers Pointor FlowLayoUtPanel 3 GroupBox 2 Panel		Button1 System Window Button1 System Window MainumSize MinimumSize Modifiers Modifi	s, Forms, Button 0, 0 5, 0 <b>Friend</b> 0, 0, 0, 0 No 117, 47 0 True
ProgressBar       ProgressBar       RchTextBox       TooTip       TooTip       TooViswi       Webbrowser       Containers       Pointer       HowLayoutPanel       GroupBox       Panel       SplitContainer		Button1 System Window ﷺ 실↓ Ⅲ 参 /Ⅲ ℍ MaximumSize ℍ MinimumSize ℍ MaximumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MinimumSize ℍ MaximumSize ℍ MinimumSize ℍ MinimumSize	s, Forms, Button 0, 0 <b>Friend</b> 0, 0, 0, 0 No 117, 47 0 True here MiddleCenter
ProgressBar RadcDutton RichTextBox TextBox TooTip TreoView Webbrowser Containers Pointer HowLayoutPanel GroupBox Panel		Buttoni System Window Buttoni System Window Bi MaximumSize Mini	s, Forms, Button 0, 0 <b>Friend</b> 0, 0, 0, 0 No 117, 47 0 True here MiddleCenter

Now try to move the control and see how the IDE will help you place it relative to the label by showing you imaginary lines for placing the control on the form

				To Carto
cobox 🗸 🕈 🗶	Start Page test02 Form1.vb [Design]*	▼ X Solution Explorer		• 4 >
+ All Windows Forms 🔷			8	
Painter	🔛 this is the second test	test02		
ab Button		My Project		
CheckBoz				
CheckedListBox				
SomboBox				
DateTimePicker				
A Label	here			
A LinkLabel	Press the button:			
ListBox				
ListYiew				
*- MaskedTextBax				
MonthCalendar				
NotfyIcon		Solution Explorer	Data Sourcas	
10 NumericupOown			(Promotion)	
J PictureBox		Properties		• 4 >
		Button1 System.Wr	ndaws Forms Button	
ProgressBar     RacicButton		82 A I I /	12	
📼 ProgressBar 🕄 RadicButton		E MaximumSize	0,0	1
ProgressBar ③ RadicButton 剂 RichTextBox		Contraction of the second seco		
ProgressBar RadicButton RichTextBox Di TextBox		MaximumSize     MinimumSize     Modfiers	0, 0 0, 0 Friend	
ProgressBar RadieButten RichTextBox FitCextBox FitCext		E MaximumSize E MinimumSiza Modifiers E Padding	0, 0 0, 0 Friend 0, 0, 0, 0	
Progress8ar RadicButton RichTextBox TextBox TextBox TodTip TrevNiew		E MaximumSize E MinimumSize Modfiors E Padding RightToLoft	0,0 0,0 Fiiend 0,0,0,0 No	
ProgressBar RadicButton RichTextBox ToxTp ToxTp ToxTp TrevNew WebBrowser		E MaximumSize E MinimumSiza Modifiers E Padding	0, 0 0, 0 Friend 0, 0, 0, 0	
🖭 ProgressDar		E MaximumSize E MinimumSize Madfiors E Padding RightTaLoft E Size	0,0 0,0 Friend 0,0,0,0 No 117,47	
ProgressBar RadicButton RadicButton RichTextBox TextBox TooTip TrevUlew WebBrowser Containers Painter		E MaximumSize MinimumSize Modifiens Padding RightToLoft G Size Tabindex	0,0 0,0 Friend 0,0,0,0 No 117,47 0 True	
ProgressBar RadicButton RadicButton RichTextBox FooTp TextBox TooTp WebBrowser Containers Pounter FlowLayoutPanel		MaximumSize     MinimumSize     MinimumSize     Modifiers     Podding     RightToLoft     Size     TabIndex     TabStop     Tag     Text	0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 True here	
ProgressBar RadicButton RichTextBox RichTextBox TextBox TextBox TotoTip TreeView TreeView WebBrowser Containers		MaximumSize     MinimumSize     MinimumSize     Modifiers     Padding     RightToLoft     Size     TabIndex     TabIndex     TabIstop     Tag     Text     TextAlign	0,0 0,0 Friend 0,0,0,0 No 117,47 0 True	
		MasimumSize     MinimumSize     MinimumSize     Modifiers     Padding     RightToLoft     Size     TabIndex     TabIndex     TabIndex     TabIndex     Tag     Text     TextAlign     Text	0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 Trus here MiddleCenter	
ProgressBar RackButton RichTextBox TooTp TooTp TooTp WebBrowser Containers Pointer FlowLayoutPanel GroupBox Panel		MaximumSize     MinimumSize     MinimumSize     Modifiers     Padding     RightToLoft     Size     TabIndex     TabIndex     TabIstop     Tag     Text     TextAlign	0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 Trus here MiddleCenter	

	3 3 = =   • • • • - ₽ • 3   ▶ □ □ 9 (₽			
cobox 🗸 🗘 🗸		★ X Solution Explorer		• 4 >
+ All Windows Forms	SIL			
- Comman Cantrols	this is the second test	test02	dis.	
N Painter	Em this is the second test	My Project		
ab Button		Form1.vb		
CheckBoz				
😭 CheckedListBox				
📲 ComboBax				
📝 DateTimePicker				
A Label				
🔥 LinkLabel	Press the button			
ListBox	luno			
ListView				
MaskedTextBax				
MonthCalendar				
NotfyIcon		Solution Explorer	Data Course	
NumericupDown		Paradon capata 4		
📓 PictureBax		Properties		• 4
📓 ProgressBar		Button1 System.Wind	ovis Forms Button	
			2	
		95 A		
RadicButton				-
RadicButton RichTextBox		E MaximumSize	0,0	
RadicButton RichTextBox Di TextBox		E MaximumSize		
RadicButton RichTextBox Di TextBox ToolTip		MaxmumSize     MinimumSize     Modfiers		
RadcButton     RichTextBox     TextBox     TooTip     TooVip		E MaximumSize		
AddcButton  RichTextBox  TextBox  TooTip  TreeView  WebBrowser		MaximumSize     MinimumSize     Modifiers     Padding	- 0,0 7,0 Friend 0,0,0,0 No 117,47	
RadicButton     RichTextBox     TextBox     TooTip     TreeView     WebBrowser     Containers		I MaximumSize MinimumSiza Modifiers Podding RightToLoft I Size TabIndex	0,0 0,0 Friend 0,0,0,0 No 117,47 0	
RadicButton     ReichTextBox     TextBox     TooTip     TreVlew     WebBrowcer     Containers     Pointer		MasmumSize     MinimumSize     Modifiers     Produing     RightToLoft     Size     TabUndex     TabStop	- 0,0 7,0 Friend 0,0,0,0 No 117,47	
		MaximumSize     MinimumSize     Modifiers     Podding     flightToLeft     Size     TabIndex     TabStop     Tag	0,0 0,0 Friend 0,0,0,0 No 117,47 0 True	
		MaximumSize     MolimumSize     Moldines     Podding     RightToLeft     Size     Tabindex     TabStop     Tag     Text	0,0 0,0 Friend 0,0,0,0 No 117,47 0 True here	
		MasmumSize     MinimumSize     Modifiers     Fadding     RightToLoft     Size     TabUndex     TabStop     Tag     Text     TextAlign	0,0 0,0 Friend 0,0,0,0 No 117,47 0 True	
		MaximumSize     MinimumSize     Modifiers     Podding     Fodding     FightToLeft     Size     TabIndex     TabStop     Tag     Text     Text	0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 True here MiddleCenter	
RadicButton     RadicButton     RichTextBox     TextBox     TooTip     TreeVlew     WebBrowcer     Containers     Pointer     FlowLayOLEPanel     SpiRContainer     SpiRContainer     TadControl		MasmumSize     MinimumSize     Modifiers     Fadding     RightToLoft     Size     TabUndex     TabStop     Tag     Text     TextAlign	0, 0 0, 0 Friend 0, 0, 0, 0 No 117, 47 0 True here MiddleCenter	

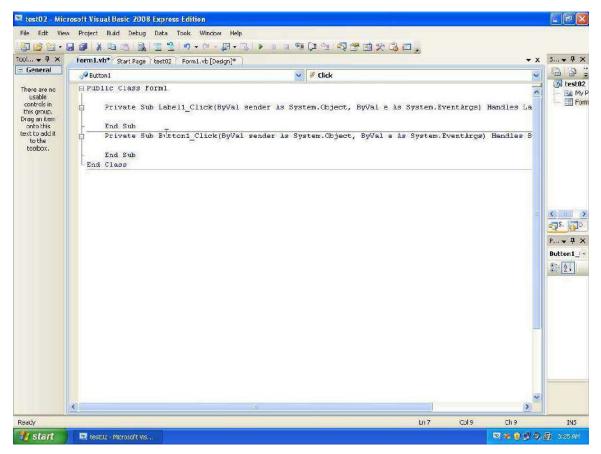
	ebug Data Format Tools Window Help	in the second		
	▲ 三 当 9 • 0 • 四 • □ 1 ● □ ● □ ● □ ● □ ● □ ● □ ● □ ● □ ● □			
rcobox 👻 🕈 🗙	Start Page test02 Form1.vb [Design]*	▼ X Solution Explorer		• <del>4</del> >
+ All Windows Forms			2	
- Common Controls	🛃 this is the second test	🗖 🔯 🕼 test02		
Pointer		My Project		
ao Button ▼ CheckBox		🦾 📰 Form1.vb		
And a second				
CheckedListBox				
ComboBox				
📷 DateTimePicker				
A Label	Press the button: here			
🛓 LinkLabel				
📲 ListBox				
🚦 ListView				
🗝 MaskedTextBax 🔤				
MonthCalendar				
📰 NotfyIcon		Solution Explorer	Data Sources	
1. NumericupOown		Properties		🗸 🕂 🤉
🔏 PictureBox		a construction of the second		10 THE
🖭 ProgressBar		Button1 System.Wr		
RadicButton		2 2 2 V III / III / III		
RichTextBox		E Maximum5ize	0,0	
IDI TEXTBOX		E MinimumSiaa	0,0	
		Modifiers	Friend	
🐛 ΤασίΤιρ		Padding     RightTaLeft	0,0,0,0	
			No	
TreeView				
G TreeView		E Size Tabindex	117,47 0	
kan TeolTip ir TreeVlew Su WebBrowser ■ Containers Pointer		E Size		
TreeView WebBrowser Containers		E Size Tablndex	0 True	
TreeView WebBrowser Containers Pointer FlowLayouCPanel		E Size Tebindex Tabitup Tag Text	0 True here	
S- TreeView C WebBrowser Containers		G Size Tabondex Tabotop Tag Text TextAlign	0 True	
TreeView WebBrowcer Containers Pointer FlowLaYou£Panel ™ GroupBox		E Size Tebindex TebStop Teg Text TextAlign TextAlign	0 True here MiddleCenter	
TreeView WebBrowcer Containers Pointer FlowLayout/Panel GroupBox Panel SplitContainer TabContainer TabContainer		G Size Tabondex Tabotop Tag Text TextAlign	0 True here MiddleCenter	
TreeView WebBrowser Containers Pointer Flowtayou:Panel GroupDox Panel SplitContainer		E Size Tebindex TebStop Teg Text TextAlign TextAlign	0 True here MiddleCenter	

📧 test02 (Runn	iing) - Microsoft Visual Bas	ic 2008 Express Edition				2	I X
21.22	Project Build Debug Da	201 211 201 201 201 201 201 201 201 201	Help				A
III 🖀 🖂 - I		空 9・0・夏・3	n a se (e		圆。		
test02 Form1					50 Solution Explorer		• # ×
1 mm						2	
Ber this is	this is the second test This is the battene	kan 📐			StestO2	iect vb	
					Solution Explo		Sources ▼ ₽ × +
					₩ <b>2</b> ↓ ■	10	
					Locked Margin MaximumSize MinimumSize Modifiers Pedding	False C, D C, D Friend C, D, C, O	*
Immediate Window				<b>→</b> 中 >	RightToLeft	0, 0, 0, 0 No	
					E Size Tag Text TextAlign		
				28	Text The text associa		ntrol.
Ready			_	181,	123 <u><u><u></u></u></u>	117 × 47	
🐮 start	🖳 testoz (Running) - Mi .	🔛 this is the second test			💌 🔅 🕄	1 9 🚇 S	25 AM

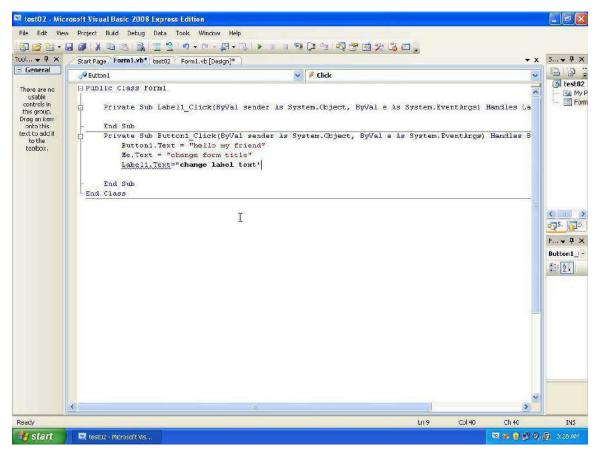
Now run the application by pressing the Play button. Now pressing the button does not do anything at all. That is because you haven't tell the computer what to do when you press the button.

🖾 test02 - Mici	osoft Visual Basic	2008 Express Edition	)				
File Edit View	Project Build I	Debug Data Tools W	indow Help				
🔄 🗃 🔛 • 🕻		A = 2 9 - (	×-Ø•□ * □ * □ * (1 * 1	2 2 2 2 X 3	<b>.</b>		
Toobox	* ‡ ×	Form1.vb* Start Pag	ge   test02   Form1.vb [Design]	▼ X	Solution Explore	t	<b>~</b> ₽×
group. Drag an te	ble controls in this m orta this text to he toolbox.	Public Class Private F End Sub	Empty Form1 Sub Label1_Click(ByVal sender	As System.O	test02	ject	
		Private End Sub End Class	Sub Button1_Click(ByVal sende	r ås System.			
					Solution Expl	orer 🙀 Data Sources	
					Properties		• 4 ×
					81 <u>21</u> 🖂		
		<		×			
Ready					ហ 7	Col 9 Ch 9	INS
🐉 start	📉 🔤 testüz - Micros	oft Vis				🗠 🌸 🜒 🖉	🧐 😰 3:25 AM

Now stop the running application by closing the window, then double click on the button, you should see something like this.



This is the code editor which helps you telling application what should it does in a specific event. For our example telling it what should it does when you press the button.



Write the lines above exactly as you see, the code means the following:

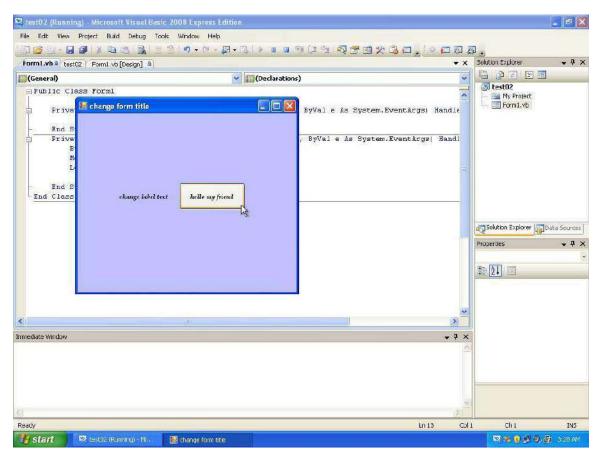
Button1.Text = "hello my friend" means that go to the button on the form those name is Button1, and modify its Text property to hello my friend. (more on that in the next tutorial).

Me.Text ="change form title" changes the title of the form by changing its Text property, and the same for the last line which changes it for the label.

It is important to understand that the Text property here is the same one that you changed in the properties window. The properties window changes the properties while you are designing the window, hence the first change you did is a design time change. However the code you just added will not be executed until you press the button while the application is running (i.e. run time), so such changes are not visible (yet). Now press F5 to see the application running

Form1.vb a test02 Form1.vb [Design] a		• X	Solution Explorer 🛛 👻
Ceneral  Public Class Form  Prive  Find S  Find S  End S  End S  Press the battore	(Declarations)       Image: Constraint of the second s		TestD2 My Project Form1.vb
¢j		× 0 ×	Properties
mediade window:		• • • •	

Now press the button and see what happens



Don't worry too much about the code. The idea here is to know that there are some properties that can be modified later on, and make you familiar with the GUI.

The next tutorial will explain about controls, their names, and their events. So for now you may save your project by pressing save all

	and the second sec	Basic 2008 Exp		- 0
	and the state of t	and the second second	a Took Window Help 월 97 - 연 - 월 - 3, ) > 11 대 역 대 대 전 20 월 20 왕 13 교 ,	
… ▼ ₽ ×	Stall Page	Form1.vb* test	02 Form1.vb [Design]*	• X 5 • 4
enera	🛷 Button 1		S Click	
There are no usable controls in this group. Drag an item onto this text to add it to the todbox.	Pr Pr Pr End C	d Sub ivate Sub Eu Buttonl.Te Me.Text = Labell.Tex Save Project Name:	abell_Click(ByVal sender is System.Object, ByVal e is System.Eventings) Handles L attoni_Click(ByVal sender is System.Object, ByVal e is System.Eventings) Handles ext = "hello my friend" "change form title" t = "change label text"	a B
		Location: Solution Name:	C \Documents and Settings/mka/My Documents\Visual Studio 2008\Projects et al.	S. (
			Save Cencel	Button
<i>y</i>	*		//	×
8				

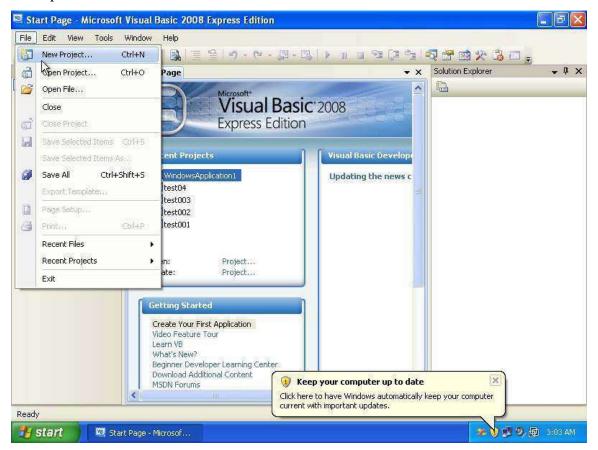
This concludes this chapter.

**Chapter 3: Understanding Buttons, and Textboxes** 

## **Understanding Buttons, and Textboxes**

This chapter deals with buttons and textboxes. We are going to see how to develop a simple calculator application using VB.NET, and examine the controls and their properties.

First open VB.NET



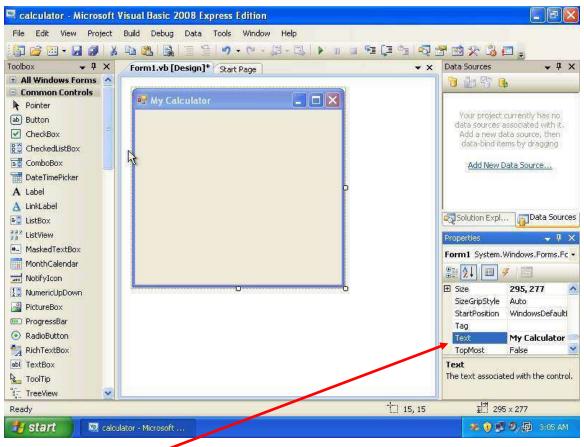
Create a new project

	soft Visual Basic 2008 Express Edition	
Name Brooks	le marilent melle	
New Projec		
× Templates:		
Visual Str	udio installed templates	
here htrol: ag an xt to t Forms A.	vs Class Library WPF WPF Browser Console	
Martin		
my temp	plates	
	]	
Search Or Templates	nline	
Search Or	nline	
Search Or Template:	nline	
Search Or Template:	nline Is	
Search Or Templates	nline s r creating an application with a Windows user interface (.NET Framework 3,5)	OK Cancel
Search Or Templates	nline s r creating an application with a Windows user interface (.NET Framework 3,5)	OK Cancel

Set its type to windows forms application and set its name to calculator. Press OK

🖾 calculator - Microsoft	Visual Basic 2008 Express Edition	
File Edit View Project	Build Debug Data Format Tools Window Help	
1 🚰 🗃 • 🖬 🗿 🛛 🐰		🗟 🚰 🥶 🏷 💪 🖾 🖕
Toolbox 👻 🗣 🗙	Form1.vb [Design] Start Page	Data Sources 🗸 🗸 🗙
🗄 All Windows Forms 🔥		1 🛃 🖓 🕒
🖃 Common Controls	🖪 Form1 💽 🗖 🔀	
R Pointer		
ab Button		
CheckBox		
CheckedListBox		
📑 ComboBox		
TateTimePicker		
A Label	Ь	
A LinkLabel		
E ListBox		Solution Explorer
222 ListView		Properties 🗸 🗸 🗙
*- MaskedTextBox		
MonthCalendar		0=- AT 1 ===
📷 NotifyIcon		
10 NumericUpDown		
📓 PictureBox		
💷 ProgressBar		N
RadioButton		R
🛃 RichTextBox		
abl TextBox		
🖢 ToolTip		
👬 TreeView 💽		
Creating project 'calculator' p	project creation successful.	
🤧 start 🛛 🔍 calcu	ulator - Microsoft	🐲 👽 😴 🧶 👰 - 3:04 AM

You should see the main form on the workspace



Change the form text property to **My Calculator**, you don't want your application to have the title **Form1** when it starts.

🖾 calculator - Microsoft V	isual Basic 2008 Express Edition		
File Edit View Project	Build Debug Data Format Tools Window Help		
👘 🗃 🖽 • 🖬 🜒 🐰	BBBBE 2 9 • 0 • 2 • B • 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	🚰 🐋 🋠 😘	🖾 📮
Toolbox 👻 🗣 🗙	Form1.vb [Design]* Start Page	1	• ‡ ×
📑 PrintPreviewDialog		1 1 La P	<b>1</b>
Process	🔜 My Calculator	a t	
ProgressBar		Your project	t currently has no
🚰 PropertyGrid	¢		associated with it. data source, then
RadioButton	N N		tems by dragging
🛃 RichTextBox	à		
🛃 SaveFileDialog		Add New	Data Source
🔎 SerialPort			
📲 ServiceController			
SplitContainer		Solution Exp	I Table Sources
+ + Splitter			
📥 StatusStrip		Properties	≁ ¢ ×
🛅 TabControl		TextBox1 Sys	item. Windows . Form: 💌
TableLayoutPanel		81 2↓ □	9 🖾
abl TextBox		1 Size	233, 20 🔥
🖄 Timer		TabIndex	0
💷 ToolStrip		TabStop	True
🛄 ToolStripContainer 📃		Tag Text	
🖢 ToolTip		TextAlign	Left 🖌
🗁 TrackBar		Text	
ैंद्रे_ TreeView		1.57257	ated with the control.
🍯 VScrollBar 🧹			
Ready	亡 19, 15	<b>∓</b> ⊠ 2	33 x 20
🛃 start 🌔 🖳 calcula	stor - Mich soft	æ 💓 :	🔋 🥝, 👘 3:05 AM
N			

From the **ToolBox**, drop a **TextBox** control onto the form, and resize it.

🖳 calculator - Microsoft V	/isual Basic 2008 Express Edition	
File Edit View Project	Build Debug Data Tools Window Help	
1 🚰 🗃 • 🖬 🥥 🐰	1) 1) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	🖻 📷 🋠 💪 🗖 🖕
Toolbox 🗸 🗣 🗙	Form1.vb [Design]* Start Page	
PrintPreviewDialog		
Process	🖷 My Calculator	1
ProgressBar		Your project currently has no
PropertyGrid	QQ	data sources associated with it. Add a new data source, then
RadioButton		data-bind items by dragging
🛃 RichTextBox		
🛃 SaveFileDialog		Add New Data Source
🔎 SerialPort		
ServiceController		
SplitContainer		Solution Expl
+ + Splitter		Properties - 4 X
- StatusStrip		TextBox1 System.Windows.Form: -
TabControl		
TableLayoutPanel		<u>≋</u> ⊉↓ <b>□</b> ≠ □
abl TextBox		(ApplicationSet
🖄 Timer		(DataBindings)
ToolStrip		AcceptsReturn False
ToolStripContainer		AcceptsTab False
TrackBar		AccessibleDesc 🛛
		(Name)
UCanallDay		Indicates the name used in code to identify the object.
Ready		0
start 🛛 🖳 calcul	lator - Microsoft	🗶 👽 🕏 🗐 🖗 3:06 AM

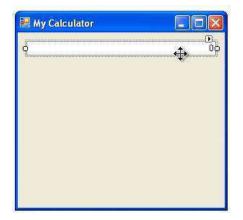
After setting the text box location, we will change its **Name** property. This property allows us to modify and access this control at runtime and change its behavior, or perform something for us. So, set it to **LCD** since it will be used to display the numbers. You can leave the name **TextBox1** without change, or select any other name. Keep in mind that the name you choose will be used instead of **LCD** in the code. Note that the **Name** property does not have any visible effect, you can't see the result of its change, because it will be used internally only to reference this control.

🖾 calculator - Microsoft V	Visual Basic 2008 Express Edition	- F 🔀
File Edit View Project	Build Debug Data Tools Window Help	
1 🚰 🗃 • 🖬 🗿  🐰	B B B = 2 9 · 0 · 2 · B > 1 = 9 Q 2 1	2 🕾 🖄 🛠 🖄 📼 🖕
Toolbox 🗸 🕈 🗙	Form1.vb [Design]* Start Page	
PrintPreviewDialog		T 🔁 💩 🖓 🕵
Process	💀 My Calculator	
ProgressBar	P	Your project currently has no
🚰 PropertyGrid	<b>0</b> 0	data sources associated with it. Add a new data source, then
RadioButton		data-bind items by dragging
🛃 RichTextBox		
🛃 SaveFileDialog		Add New Data Source
🔎 SerialPort		
📲 ServiceController		
SplitContainer		Solution Expl
+ + Splitter		
는 StatusStrip		Properties 🗸 🗣 🗙
TabControl		TextBox1 System.Windows.Form: -
TableLayoutPanel		82 2↓ □ 🖉 🖾
abl TextBox		TabStop True 🔦
🖄 Timer		Tag
ToolStrip		Text 0
🛄 ToolStripContainer 🔜	N	TextAlign Left UseSystemPas: False
🖢 ToolTip	R.	UseWaitCursor False
0— TrackBar		Text
ting TreeView		The text associated with the control,
🚆 VScrollBar 🛛 🐱		
Ready		
🛃 start 🛛 🖾 calcu	lator - Microsoft	🛷 😧 🕏 🧶 📵 3:06 AM
Stant Start		

Now, we start working with the display, usually when the calculator starts it should display the number 0. In out case it does not. So we modify the Text Property and write 0 in that. Make sure you don't add any spaces before or after the 0.

🖷 calculator - Microsoft V	/isual Basic 2008 Express Edition	<b>_</b> 2 X
File Edit View Project	Build Debug Data Tools Window Help	
1 🗗 🚅 🖽 • 🖬 🗶 🐰	🛍 📇 🔄 🖉 • 🖉 • 🖉 • 🖳 🕨 🗉 🖼 📮 🛃	🚰 🐋 🛠 🗳 🖾 🖕
Toolbox 👻 🕈 🗙	Form1.vb [Design]* Start Page	Data Sources 🗸 🗸 🗸 🗙
PrintPreviewDialog		
Process	💀 My Calculator	
ProgressBar		Your project currently has no
The property Grid Property Grid	¢0	data sources associated with it. Add a new data source, then
RadioButton		data-bind items by dragging
🛃 RichTextBox		
🛃 SaveFileDialog		Add New Data Source
🔎 SerialPort		
ServiceController		
SplitContainer		Solution Expl
+ + Splitter		
📙 StatusStrip		Properties 🗸 🗘 🗙
TabControl		TextBox1 System.Windows.Form: -
TableLayoutPanel		31 2↓ □ 🖉 🖾
abl TextBox		TabStop True 📩
🖄 Timer		Tag
ToolStrip		Text 0 TextAlian Left V
ToolStripContainer		UseSystemPast eft
🛓 ToolTip		UseWaitCursor Right
♥─ TrackBar		TextAlign Center
treeView		Indicates how the text should be
🚆 VScrollBar 🛛 😼		aligned for edit controls.
Ready		
🐴 start 🛛 🖾 calcul	ator - Microsoft	🐲 💽 🛃 🧶 🔂 3:06 AM
S orant		

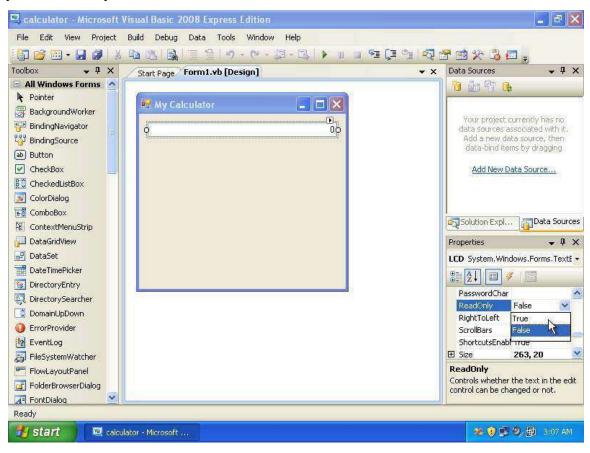
The second thing to note is that the number is aligned to the left, while calculators align the number to the right. Search for the **TextAlign** property and change it to Right.



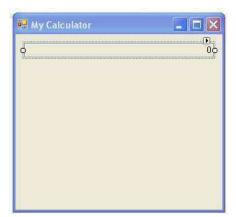
This what the window will look like.

Name:	calculator	
Location:	C:\Documents and Settings\mka\My Documents\Visual Studio 2008\Projects.	
Solution Name:	calculator Create directory for solution	

By the way, we forgot to save the project, so, press save all. It is good practice to save your work every few minutes.



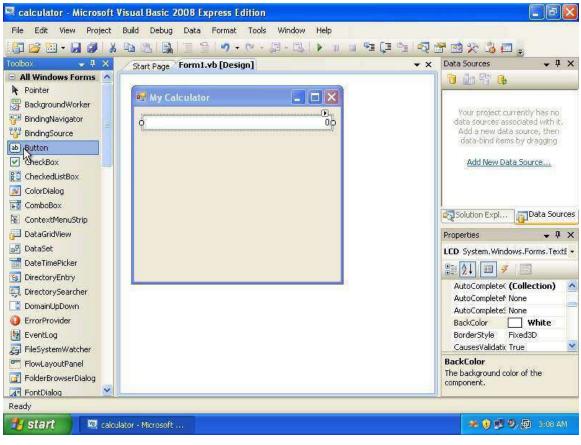
Now if you run the application, you will notice that you can access the text box via the cursor, and write some text, or remove the text. You don't want that to happen. Simply change the **ReadOnly** property to **True**. Notice that once the **ReadOnly** property is ture, the text box will look like this:



The solution to this problem is simple. Go to the **BackColor** property of the text box and select the white color for it.



The window should look fine, next we will add the buttons to the form.



From the tool box window, drag and drop a **Button** onto the form.

🕮 calculator - Microsoft	t Visual Basic 2008 Express Edition	E 🖬 🔀
File Edit View Project	Build Debug Data Tools Window Help	
🚺 😂 🖽 • 🖬 🌒 👌	🖌 🛍 🕵 🔄 일 🤊 - 언 - 및 - 및 🕨 🖬 🖉 👰 🛃	🚰 过 🛠 👶 🖂 🖕
Toolbox 👻 🕂 🗙	Start Page Form1.vb [Design]*	Data Sources 🛛 👻 🕂 🗙
😑 All Windows Forms 🔥		
Pointer	🖶 My Calculator	
BackgroundWorker		Your project currently has no
🚏 BindingNavigator	0	data sources associated with it.
🚏 BindingSource		Add a new data source, then data-bind items by dragging
ab Button	d Butt on P	data bina items by dragging
CheckBox	, damoninio	Add New Data Source
CheckedListBox		
🗾 ColorDialog		
ComboBox		
ContextMenuStrip		Solution Expl
🚰 DataGridView		Properties 🗢 🗸 🛠
🛃 DataSet		Button1 System.Windows.Forms.I -
🔡 DateTimePicker		3 A I 🖉 🖉
🛐 DirectoryEntry		Linear Contractor Internet in the Allowed Contractor
👯 DirectorySearcher		
DomainUpDown		(Name) n1
ErrorProvider		AccessibleDesc
EventLog		AccessibleNam
👼 FileSystemWatcher		AccessibleRole Default
FlowLayoutPanel		(Name)
FolderBrowserDialog		Indicates the name used in code to identify the object.
🕼 FontDialog 🛛 💆		
Ready		
🛃 start 🛛 🖳 cale	culator - Microsoft	👒 😧 🛒 🔍 📵 - 3:06 AM

Change its **Name** property to **n1**. This will help us identify which number was pressed.

🖳 calculator - Microsoft	Visual Basic 2008 Express Edition	<b>_</b> 2 ×
File Edit View Project	Build Debug Data Tools Window Help	
🚺 💕 记 • 🖬 🗿 🐰	) 🛍 🚨 📓 🗏 🗐 • • • • 💷 • 🖳 🕨 🗉 🖼 📮 🖏	🚰 过 🛠 💪 🖂 🖕
Toolbox 👻 🗣 🗙	Start Page Form1.vb [Design]*	Data Sources 🛛 👻 🕂 🗙
😑 All Windows Forms 🔥		
Pointer	🖶 My Calculator	
BackgroundWorker		Your project currently has no
🚰 BindingNavigator	0	data sources associated with it.
🚏 BindingSource		Add a new data source, then data-bind items by dragging
ab Button	9 Butt 9	data bina items by dragging
CheckBox	d-d-d	stom Web System
CheckedListBox		ActiveBorder
🗾 ColorDialog		ActiveCaption
E ComboBox		ActiveCaptionText
📓 ContextMenuStrip		ButtonFace
🚰 DataGridView		ButtonHighlight ButtonShadow
🖉 DataSet		🖬 Control
📷 DateTimePicker		ControlDark
🛐 DirectoryEntry		
👯 DirectorySearcher		ControlLightLight
DomainUpDown		ControlText
ErrorProvider		BackColor Control 🔨
🔄 EventLog		BackgroundIma (none)
👼 FileSystemWatcher		BackgroundIma Tile
FlowLayoutPanel		BackColor
🔄 FolderBrowserDialog		The background color of the component.
🔄 FontDialog 🛛 💆		
Ready		
🛃 start 🔰 🖾 calc	ulator - Microsoft	🐲 👔 🕏 🧶 🗐 - 3:08 AM

Change the **backcolor** property for the button, usually you can select from **Custom** a different color other than the ones the system provide.

🖾 calculator - Microsoft	Visual Basic 2008 Express Edition	
File Edit View Project	Build Debug Data Tools Window Help	
1 🚰 🗃 • 🖬 🜒 🐰	🗈 🚨 📓 🗏 🗐 • (* • 💭 • 🖳 🕨 🖬 📮 📮 😭	
Toolbox 🗸 🗣 🗙	Start Page Form1.vb [Design]*	👻 🗙 Data Sources 🔍 🗸 🖡 🗙
😑 All Windows Forms 🔥		1 1 1 Pr 🔒
Pointer BackgroundWorker	My Calculator	Your project currently has no
BindingSource	0	data sources associated with it. Add a new data source, then
ab Button		data-bind items by dragging
CheckBox		Add New Data Source
ColorDialog		
ComboBox		
📓 ContextMenuStrip		Solution Expl
📁 DataGridView		Properties 🔷 🗸 🗶
🗗 DataSet		Button1 System, Windows, Forms, I -
DateTimePicker		2↓ ■ 🖌 🔳
🛐 DirectoryEntry		TabStop True
👯 DirectorySearcher		
DomainUpDown		Text 1
ErrorProvider		TextAlign MiddleCentre
EventLog		TextImageRela Overlay
FileSystemWatcher		UseCompatible False
FlowLayoutPanel		Text The text associated with the control.
FolderBrowserDialog		The text associated with the control,
🔄 FontDialog 💆	L	
Ready		
🐉 start 🔰 🖳 calcu	ilator - Microsoft	🦀 💓 🐯 🧐 🌆 3:09 AM

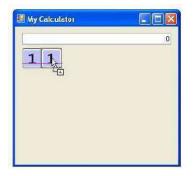
After the color changes, we will modify the text that the button is displaying, so change the text property into **1**.

💌 calculator - Microsoft V	'isual Basic 2008 Express Edi	ition				X
File Edit View Project	Build Debug Data Tools W	Vindow Help				
1 🚰 🗃 • 🖬 🜒 🐰	B B B E 9 .	(* - II - II - *	11 🖬 🖼 🕻 🖬 🖄	2	1 2 3	· ·
Toolbox 👻 🕈 🗙	Font			?	ata Sources	• 4 ×
😑 All Windows Forms 🔥					0 1 9 6	4
Pointer	Font	Font style:	Size:			
BackgroundWorker	Microsoft Sans Serif	Regular		К	Vour project	currently has no
🚏 BindingNavigator	Microsoft Sans Serif	Regular Italic	g Car	ncel	data sources a	associated with it.
🚏 BindingSource	🚺 🕖 Palatino Linotype	Bold	10			ata source, then ems by dragging
ab Button	O Raavi O Shruti	Bold Italic	11 12		Gales and R	1111151 (19338119)
CheckBox	O Sylfaen		14		Add New [	Data Source
CheckedListBox	🛛 O Symbol 🔛		16 💌			
🗾 ColorDialog	Effects	Sample				
E ComboBox	Strikeout					
ContextMenuStrip		AaBbYyZ	2z		Solution Expl.	Data Sources
🚰 DataGridView		1			roperties	<b>↓</b> ₽ ×
🗗 DataSet		Script:			utton1 System	n.Windows.Forms.I 👻
T DateTimePicker		Western	~			ý I 🖾
🛐 DirectoryEntry					distant and a second	a second s
👯 DirectorySearcher					Enabled FlatAppearan	True 🔬
DomainUpDown					FlatStyle	Standard
ErrorProvider				E	E Font	Microsoft Sans &
🔄 EventLog					ForeColor	
👼 FileSystemWatcher					GenerateMem	t True 👱
FlowLayoutPanel				1.5	Font	a a a assa
📑 FolderBrowserDialog					The font used to control.	display text in the
🔄 FontDialog 🛛 💌						
Ready						
🛃 start 🛛 🔍 calcula	ator - Microsoft				<b>* 0</b> 13	9. 😥 3:09 AM

Now the number is very small, we can increase its size a little bit, so go to the **font** property, and set its font to courier new, and size to 20 for example.

🖾 calculator - Microsoft	Visual Basic 2008 Express Edition		
File Edit View Project	Build Debug Data Tools Window Help		
🔚 🗃 记 • 🖬 🗿 🐰	- 🗈 🚨 🗟 🔄 😫 🔊 • 🕫 • 📮 • 🖳 🕨 🗉	SI (I SI 🖏 )	P 🖻 🛠 👶 🗖 🖕
Toolbox 👻 🗣 🗙	Start Page Form 1.vb [Design]*	• ×	Data Sources 🛛 👻 🕂 🗙
😑 All Windows Forms 🔥			
Pointer	💀 My Calculator		
BackgroundWorker		R	Your project currently has no
🚏 BindingNavigator	0	10	data sources associated with it.
🚏 BindingSource	<b>9</b>		Add a new data source, then data-bind items by dragging
ab Button	9 <b>1</b> P		data-bind rems by bragging
CheckBox	0-0-0		Add New Data Source
CheckedListBox			
🗾 ColorDialog			
📷 ComboBox			
ContextMenuStrip			Solution Expl
进 DataGridView			Properties 🚽 🗸 🗙
🛃 DataSet			Button1 System.Windows.Forms.I -
DateTimePicker			8 21 0 7 0
🛐 DirectoryEntry			and the second descent of the second descent of the second descent descent descent descent descent descent des
👯 DirectorySearcher			Enabled True A
DomainUpDown			FlatStyle Standard
ErrorProvider			E Font Courier Ne ()
🔄 EventLog			ForeColor ControlTex
👼 FileSystemWatcher			GenerateMemt True
FlowLayoutPanel			Font
📑 FolderBrowserDialog			The font used to display text in the control.
🔄 FontDialog 🛛 💆			
Ready			
🛃 start 🛛 🖾 calcu	ulator - Microsoft		🛷 😧 🛃 🧶 📵 3:09 AM

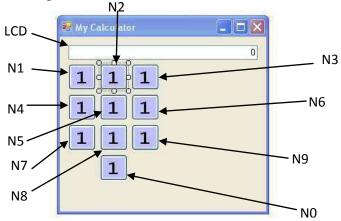
Now we can repeat the same operation to all the other nine buttons, or we can just copy this button and get the same result quickly. Just hold the **ctrl** key and drag the control.



Repeat the operation again for all the numbers

👽 My Calculator	
	0

Now we change the names of the buttons (or you can leave them as they are and skip this part). The names will continue to be n2,n3,n4,n5,n6,n7,n8,n9, and n0.



Next we change the text property for each one to display the numbers from 1 to 9, and 0 finally.

🖶 My Calculator	
123 456 789	0

Now if you run the application, you won't be able to do anything, i.e. the form appears, and pressing the buttons will have no effect. This is because we haven't tell the application what to do when you click any of the buttons. So, double click the button N1 to go to its event

File Edit View Project	Build Debug Data Tools Window Help	
🗊 🧉 • 🖬 🔊 🕷	Balaa = 2   9 · C · 2 · B   ▶    = 9 (2 ≤ 10 d	r 🖻 🛠 🕉 🗖 🖕
oolbox 🗸 🗣 🗙	Form1.vb* Start Page Form1.vb [Design]*	Data Sources 🚽 🗸 🗸
🗄 General	👻 Empty	🐌 🗄 🖓 📭
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.	Public Class Form1 Private Sub n1_Click(ByVal sender & System.( I End Sub End Class	Your project currently has no data sources associated with it. Add a new data source, then data-bind items by dragging Add New Data Source
	E	Solution Expl Properties + 4 >
leady	Ln 4 Col 9	Ch 9 INS

What you see here is a procedure or subroutine. This is a small block of code that you write to tell the application what to do when a button is clicked. The **n1\_Click** is the name of this procedure. It tells you that it get executed when the button whose name **n1** is clicked by the user. Write down the code to have it like this:

Note:I made the text small so that it fits one line. In VB.NET the new line is a sentence terminator.

The code means the following:

**Private** Sub part defines the subroutine. It is part of the language (or keywords as they call it).

n1\_click is the name of that subroutine. You can use the name to call the subroutine whenever you need. The name can be anything you choose. More on that in later tutorials.

(ByVal sender As System.Object, ByVal e As System.EventArgs) : these are called the arguments. The allow the subroutine to be flexible by checking some inputs, to process things differently. More on that later.

**Handles** n1.Click : this one means that this subroutine will be automatically called whenever the button n1 is clicked by the end user.

LCD.Text = LCD.Text & "1": this is the processing we are performing when we press the button 1. Its meaning is add to the current text on the display (which we call LCD) the number 1. Note that we used the Text property which we modified previously using the properties window.

**End Sub** : signals the end of subroutine.

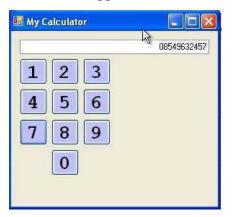
You should repeat the code for buttons n2,n3,n4,n5,n6,n7,n8,n9, and n0 to add the numbers 2,3,4,5,6,7,8,9,0 respectively. So the code should look like this:

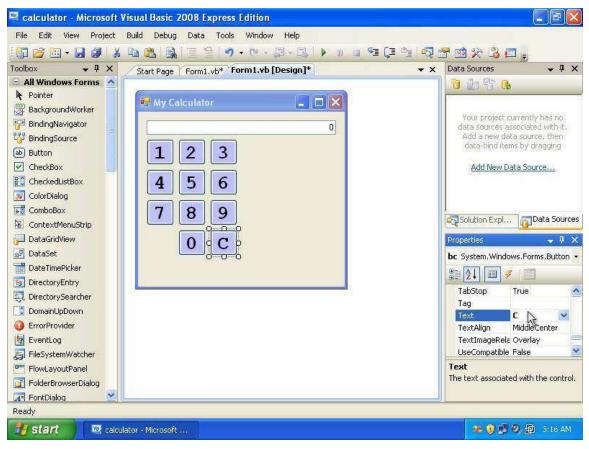
```
Private Sub n1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n1.Click
    LCD.Text = LCD.Text & "1"
End Sub
Private Sub n2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n2.Click
    LCD.Text = LCD.Text & "2"
End Sub
Private Sub n3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n3.Click
    LCD.Text = LCD.Text & "3"
End Sub
Private Sub n4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n4.Click
    LCD.Text = LCD.Text & "4"
```

End Sub

```
Private Sub n5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n5.Click
           LCD.Text = LCD.Text & "5"
    End Sub
    Private Sub n6 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n6.Click
            LCD.Text = LCD.Text & "6"
   End Sub
    Private Sub n7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n7.Click
           LCD.Text = LCD.Text & "7"
   End Sub
    Private Sub n8 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n8.Click
           LCD.Text = LCD.Text & "8"
    End Sub
    Private Sub n9 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n9.Click
            LCD.Text = LCD.Text & "9"
   End Sub
    Private Sub nO Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n0.Click
            LCD.Text = LCD.Text & "0"
   End Sub
```

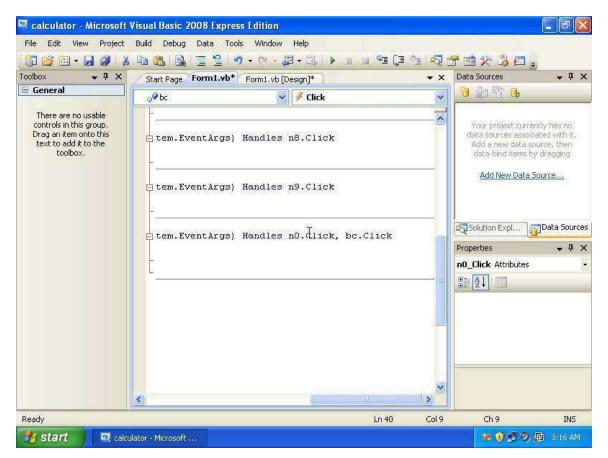
#### Now run the application and click few buttons:





Next we create the clear button, copy the 0 button, and place the copy next to the 0.

Change its name to **bc**, and text property to **C**.



Now if you double click the C button, you will go to the code handler of 0. Check the code and you will see that this subroutine will handle the events of **n0.click** and **bc.click**. Actually this will make both buttons behave the same way, so you should create a separate handler for **bc**. The reason why VB.NET linked them together into the same subroutine, is that you copied n0, so VB assumes that the copies should behave the same way as the original. What you should do now is remove the **, bc.Click** from the procedure and then create the handler for clear button.

File Edit View Project			ols Window He			- 12 er - er e		
index 🚽 🖬 🖌						1	2 🛠 🚨 🗖 🚽	2 1. H
olbox 👻 🕂 🗙 General	Start P	age / Form1.vb*	Form1.vb [Desig	]*			Data Sources	• 4 ×
General	(Gene	eral)	× 1	(Declaration	s)	~	1 b 5 6	
controls in this group. Drag an item onto this text to add it to the toolbox.		LCD.Te End Sub Private Su	ext = LCD.Te	xt & "O"	ler As System. ler As System. ]	- .«	Your project curre data sources associ Add a new data so data-bind items b Add New Data 3	ated with it. urce, then / dragging ource
	<	- 101			0		Properties	• ₽ ×
ror List					÷ 1	φ×		
🕽 0 Errors 🛛 <u> A</u> 0 Warnings	(i) 0 Mess	ages						
Description		File	Line	Column	Project			

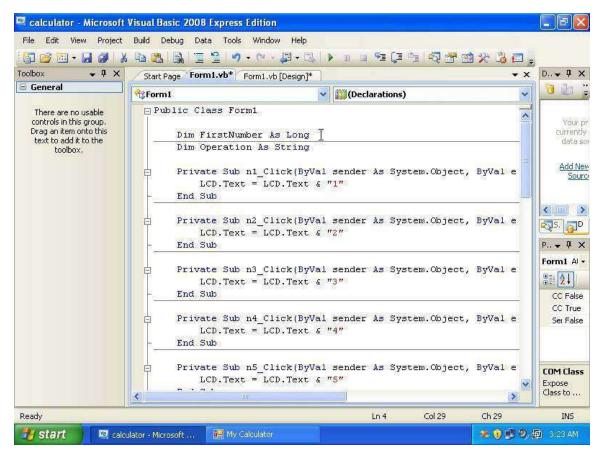
Just go back to design window, then double click the C button and add the text shown above. Try to run the program, clicking some numbers and then pressing the C button.

🖾 calculator - Microsoft	Visual Basic 2	008 Express Edition			
File Edit View Project	Build Debug	Data Format Tools	<b>-</b> -	•	Your project currently has no
<ul> <li>BindingSource</li> <li>Button</li> <li>CheckBox</li> <li>CheckedListBox</li> <li>ColorDialog</li> <li>ComboBox</li> <li>ContextMenuStrip</li> <li>DataGridView</li> <li>DataSet</li> </ul>	4	56- 89* 0C/			data sources associated with it.         Add a new data source, then         data-bind items by dragging         Add New Data Source         Solution Expl         Properties
Error List	(i) 0 Messages			<b>↓</b> ‡	× :: 21 🗉 🥖 🖾
Description	File	e Line	Column	Project	Image: Constraint of the second se
Ready					(Name) Indicates the name used in code to identify the object.
	ılator - Microsoft .				🐠 😯 🛃 🧐 🔂 3:20 AM

Now assuming you know now how to copy a control and change its text property, now, add the operations as shown above. Next name them as follows:

- + name is **bad**
- name is **bsub**
- \* name is **bmult**
- / name is **bdiv**
- = name is **bequal**

And then remove their handlers from the subroutines because we want to write new event handlers for them.



Double click the + button to go to the code window. Now add the code :

```
Dim FirstNumber As Long
Dim Operation As String
```

FirstNumber is a variable that helps the application remember integer numbers. The reason we use it is to keep track of the last number we entered into the calculator. When we press any operation the display is going to be cleared and so the number is lost, so this variable will store the number before it is removed from the display. Operation is a variable used to remember the last operation that is being pressed. It will store +,-,\*,/.

Next add the following code event handles into the subroutines of +,-,\*,/ respectively:

```
FirstNumber = LCD.Text
LCD.Text = "0"
Operation = "+"
```

### The - event hanlder

```
FirstNumber = LCD.Text
LCD.Text = "0"
Operation = "-"
```

# The \* event handler

FirstNumber = LCD.Text
LCD.Text = "0"
Operation = "\*"

## The / event handler

FirstNumber = LCD.Text
LCD.Text = "0"
Operation = "/"

## So the code should look like

```
Private Sub badd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles badd.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "+"
   End Sub
    Private Sub bsub_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bsub.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "-"
   End Sub
    Private Sub bmult Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bmult.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "*"
   End Sub
```

```
Private Sub bdiv_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bdiv.Click
    FirstNumber = LCD.Text
    LCD.Text = "0"
    Operation = "/"
End Sub
```

The last thing to do is the = handler where the operation should be executed

```
Private Sub bequal Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bequal.Click
        Dim SecondNumber As Long
       Dim Result As Long
       SecondNumber = LCD.Text
        If Operation = "+" Then
            Result = FirstNumber + SecondNumber
       ElseIf Operation = "-" Then
           Result = FirstNumber - SecondNumber
       ElseIf Operation = "*" Then
           Result = FirstNumber * SecondNumber
       ElseIf Operation = "/" Then
            Result = FirstNumber / SecondNumber
       End If
        FirstNumber = Result
       LCD.Text = Result
   End Sub
```

The code first gets the value from the display, then it check the last operation, if it is addition, then the result will be the sum of the number in memory and the number just we get from the display. If subtraction then subtract one from the other and so on. Finally store the result into memory for further operations on it (you can neglect that, it is not that you have to do it), and then display the result by updating the text property of LCD.

Now run the application and try adding two numbers like 30 and 5:

My Calculator	8	
	₽.	35
1 2 3	+	
4 5 6	-	
7 8 9	*	
= 0 C	: 7	

Next you will notice that when you enter any number you always gets a zero before that number. This is meaningless and should not happen in a calculator. So we will update our code to get rid of the zero in case we clicked on any number (0,1,2,3,4,5,6,7,8,9) and there is a 0 in the display. So update the code to be:

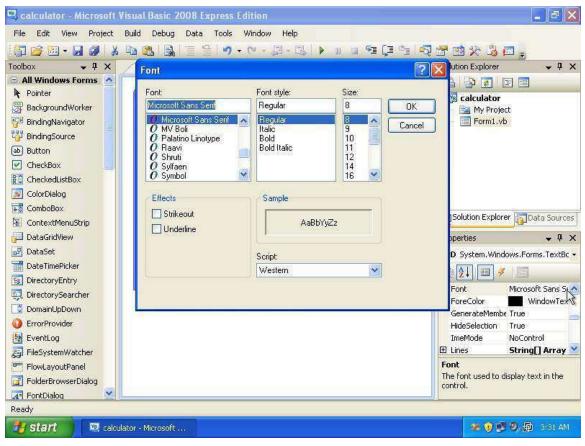
```
Private Sub n1 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n1.Click
       If LCD.Text = "0" Then
           LCD.Text = "1"
       Else
           LCD.Text = LCD.Text & "1"
       End If
   End Sub
    Private Sub n2 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n2.Click
       If LCD.Text = "0" Then
           LCD.Text = "2"
       Else
           LCD.Text = LCD.Text & "2"
       End If
   End Sub
    Private Sub n3 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n3.Click
       If LCD.Text = "0" Then
           LCD.Text = "3"
       Else
           LCD.Text = LCD.Text & "3"
       End If
   End Sub
    Private Sub n4 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n4.Click
       If LCD.Text = "0" Then
```

```
LCD.Text = "4"
       Else
           LCD.Text = LCD.Text & "4"
       End If
   End Sub
    Private Sub n5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n5.Click
       If LCD.Text = "0" Then
           LCD.Text = "5"
       Else
           LCD.Text = LCD.Text & "5"
       End If
   End Sub
    Private Sub n6 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n6.Click
       If LCD.Text = "0" Then
           LCD.Text = "6"
       Else
           LCD.Text = LCD.Text & "6"
       End If
   End Sub
    Private Sub n7 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n7.Click
       If LCD.Text = "0" Then
           LCD.Text = "7"
       Else
           LCD.Text = LCD.Text & "7"
       End If
   End Sub
    Private Sub n8 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n8.Click
       If LCD.Text = "0" Then
           LCD.Text = "8"
       Else
           LCD.Text = LCD.Text & "8"
       End If
   End Sub
```

```
Private Sub n9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n9.Click
       If LCD.Text = "0" Then
           LCD.Text = "9"
       Else
           LCD.Text = LCD.Text & "9"
       End If
    End Sub
   Private Sub n0_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n0.Click
       If LCD.Text = "0" Then
           LCD.Text = "0"
       Else
           LCD.Text = LCD.Text & "0"
       End If
   End Sub
```

This is not the best way to write the code, but it is just to practice working in the environment. Later on we will get into the language itself and understand how it works after understanding most of the controls.

Now run the application and see that the zero disappears when you click any number. After checking that the calculator is working fine, we will make some changes into it. First we change the size of the font in the LCD to be a little bit bigger so select the LCD control (the textbox whose name is LCD), and change its Font property



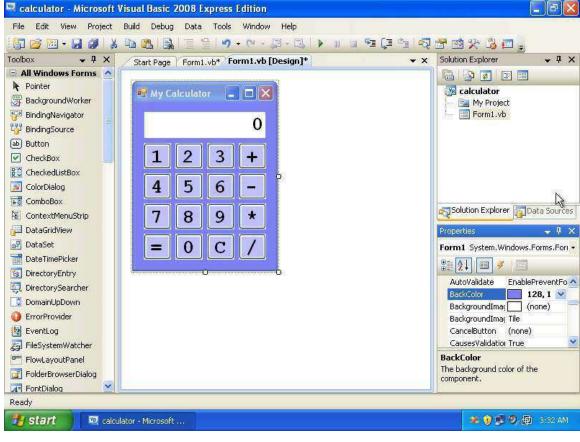
Try to get a result similar to this:

}			0	<b>)</b>  -
1	2	3	+	
4	5	6	_	
7	8	9	*	
=	0	C	/	

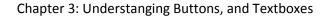
Next we will modify the form behavior so that it does not change size.

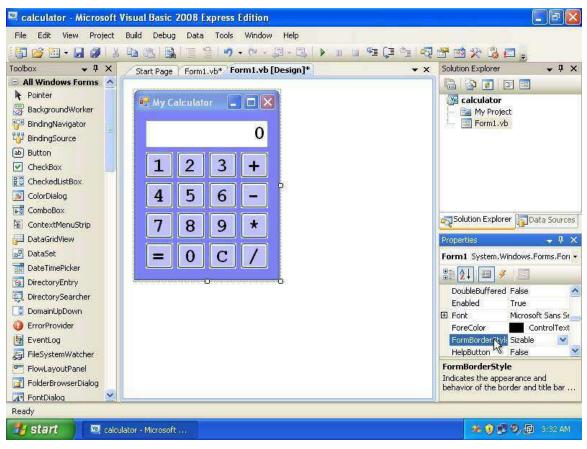
ile Edit View Project Build Debug	Data Format Tools Window Hel	N. Contraction of the second se			
	☐ 월 9 • @ • ₽ • ₽   ▶		1		
olbox	Form1.vb* Form1.vb [Design]*	<b>→</b> ×		<b>- - -</b>	¥ )
Pointer					
BackgroundWorker	lculator 📃 🗖 🔀		🚺 🌆 calculator	ect	
BindingNavigator			Form1.v		
BindingSource	Ο				
Button					
CheckBox	2 3 +				
CheckedListBox					
ColorDialog	5 6 -				
🖥 ComboBox				in the second	
ContextMenuStrip	8 9 *			er Data Sou	1rc
DataGridView			Properties	🖛 Q	Ą
🗄 DataSet 🛛 🗧 🗌	0 C /	Si	Form1 System.W	/indows.Forms.F	For
DateTimePicker		3			
DirectoryEntry	Ú		E Font	Microsoft Sans	< 5
DirectorySearcher			ForeColor	ControlTe	
DomainUpDown			FormBorderSty	le Sizable	
ErrorProvider			HelpButton	False	
EventLog				NoControl	
FileSystemWatcher			Font	Mocontrol	-
FlowLayoutPanel			The font used to a	display text in the	ne
FontDialog			control.		
				and the second se	_

First try to position the controls again to look more professional, you can move all the controls together by selecting them all. The selection works by drawing a box that embed them all. Next click on any empty space on the form so that you can change its properties

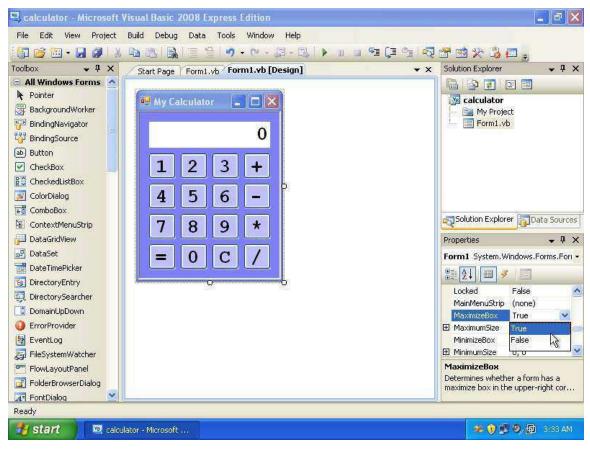


Change Backcolor property as you see fit.





Next change the **FormBorderStyle** property from **Sizable** to **FixedSingle**, this will prevent your calculator from being resized.



Finally change the **MaximizeBox** for your form to be **False**. This prevents the form from being maximized.

🖳 calculator (Running) - Microsoft Visual Basic 2008 Express Edition	🖃 🖻 🔀
	Solution Explorer 🗸 🖡 🗙
Image: Wy Calculator       Image: Wy Calculator         Image: Update of the control of th	Calculator Calculator Calculator Form1.vb
7 8 9 * = 0 C / 7 8 9 *	Solution Explorer Data Sources Properties • 4 × Form1 System. Windows.Forms.Form •
	Image: Contract of the second seco
Name Value Type	MainMenuStrip (nonė) MaximizeBox False MaximumSize 0, 0 = MinimizeBox True MinimumSize 0, 0 ✓
	MaximizeBox Determines whether a form has a maximize box in the upper-right corn
in the start Ready Calculator (Running)	🗯 👔 🛃 🧐 👜 - 3133 AM

So this is how your calculator should look like.

Just in case it does not work, the code should be as below assuming you used the same names:

```
Public Class Form1
Dim FirstNumber As Long
Dim Operation As String
Private Sub n1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n1.Click
If LCD.Text = "0" Then
LCD.Text = "1"
Else
LCD.Text = LCD.Text & "1"
End If
End Sub
```

```
Private Sub n2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n2.Click
       If LCD.Text = "0" Then
           LCD.Text = "2"
       Else
           LCD.Text = LCD.Text & "2"
       End If
   End Sub
    Private Sub n3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n3.Click
       If LCD.Text = "0" Then
           LCD.Text = "3"
       Else
           LCD.Text = LCD.Text & "3"
       End If
   End Sub
    Private Sub n4 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n4.Click
       If LCD.Text = "0" Then
           LCD.Text = "4"
       Else
           LCD.Text = LCD.Text & "4"
       End If
   End Sub
    Private Sub n5 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n5.Click
       If LCD.Text = "0" Then
           LCD.Text = "5"
       Else
           LCD.Text = LCD.Text & "5"
       End If
   End Sub
    Private Sub n6 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n6.Click
       If LCD.Text = "0" Then
           LCD.Text = "6"
       Else
           LCD.Text = LCD.Text & "6"
       End If
```

```
End Sub
    Private Sub n7 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n7.Click
       If LCD.Text = "0" Then
           LCD.Text = "7"
       Else
           LCD.Text = LCD.Text & "7"
       End If
   End Sub
    Private Sub n8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n8.Click
       If LCD.Text = "0" Then
           LCD.Text = "8"
       Else
           LCD.Text = LCD.Text & "8"
       End If
   End Sub
    Private Sub n9 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n9.Click
       If LCD.Text = "0" Then
           LCD.Text = "9"
       Else
           LCD.Text = LCD.Text & "9"
       End If
   End Sub
    Private Sub nO Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles n0.Click
       If LCD.Text = "0" Then
           LCD.Text = "0"
       Else
           LCD.Text = LCD.Text & "0"
       End If
   End Sub
    Private Sub bc Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bc.Click
       LCD.Text = "0"
   End Sub
```

```
Private Sub badd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles badd.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "+"
   End Sub
    Private Sub bsub Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bsub.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "-"
   End Sub
    Private Sub bmult Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bmult.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "*"
   End Sub
    Private Sub bdiv Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bdiv.Click
       FirstNumber = LCD.Text
       LCD.Text = "0"
       Operation = "/"
   End Sub
    Private Sub bequal Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bequal.Click
        Dim SecondNumber As Long
       Dim Result As Long
       SecondNumber = LCD.Text
        If Operation = "+" Then
            Result = FirstNumber + SecondNumber
       ElseIf Operation = "-" Then
            Result = FirstNumber - SecondNumber
       ElseIf Operation = "*" Then
            Result = FirstNumber * SecondNumber
        ElseIf Operation = "/" Then
            Result = FirstNumber / SecondNumber
```

```
End If
FirstNumber = Result
LCD.Text = Result
```

End Sub End Class **Chapter 4: Dialogs and Menus** 

# **Dialogs and Menus**

In this chapter we will be designing a simple Notepad application. Please notice that this chapter focuses only on the controls not the code. More about the code later...

New Project       Jobox       Gener       There       Orag ar       text to       My Templates       Search Online       Templates	installed templates Class Library WPF Application	WPF Browser Application Application			• <b></b>
Albox Geners There controls Drag ar text to	Class Library WPF Application	WPF Browser Console			• 7
Visual Studio i There controls Drag ar text to Windows Forms A My Templates Search Online Templates	Class Library WPF Application	WPF Browser Console			
control: Vrag ar text to t My Templates Search Online Templates	Class Library WPF Application	WPF Browser Console			
A project for creat					
	ting an application with a Wind	ows user interface (.NET Frame	ework 3.5)		
Name:	My Notepad	<i>₽</i>			
	Learn VB What's New? Beginner: Developer Lea Download Additional Co		0	K Cancel	
	MSDN Forums		×		
	1.22.1	29.1			

Create a new project and name it: My Notepad

🖾 My Notepad - Micr	osoft V	/isual Basic 2008 I	xpress Editio	n				×
File Edit View Proj	ject B	uild Debug Data	Tools Window	v Help				-
i 🕼 🗃 🖬 • 🖬 🖉	× 1	b 😤 🖳 🗏 😫	19 - (2 -	月-四十十	I 🛛 🖻 🕽 🖕	2 2 2 2 3	3 💷 🖕	
Toolbox 🗸 🗸 🖡		Form1.vb [Design]			▼ ×	Solution Explorer	<b>↓</b> ₽	×
😑 All Windows Forms							1 🔳	
Pointer		🕂 My Notepad				My Notepad		
🐺 BackgroundWorker		на му мотерао			2	My Project		
🚰 BindingNavigator						Form1.vb		
🚏 BindingSource								
ab Button								
CheckBox								
CheckedListBox								
😹 ColorDialog					6			
📑 ComboBox								
📓 ContextMenuStrip						Solution Explorer	Data Sources	
📕 DataGridView						Properties	🕶 ‡	×
避 DataSet						Form1 System.Wir	ndows.Forms.Form	
📷 DateTimePicker							10	
🛐 DirectoryEntry								1000
🖏 DirectorySearcher						∃ Size	300, 300 Auto	^
DomainUpDown			<u>.</u>		b	SizeGripStyle StartPosition	WindowsDefaultLo	r
ErrorProvider						Tag		1
🔄 EventLog						Text	My Not ad	-
👼 FileSystemWatcher						TopMost	False	~
FlowLayoutPanel						Text		
FolderBrowserDialog						The text associated	with the control.	
FontDialog								
Ready								
💏 start 🛛 📖	My Nob	epad - Microso					1 😎 🧐 📵 🕢 3:49 AT	MI -
	- All All All All All All All All All Al							

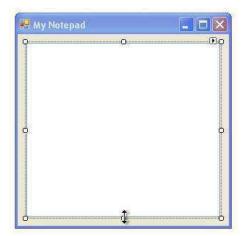
Rename the form title to My Notepad by modifying the Text property of the form.

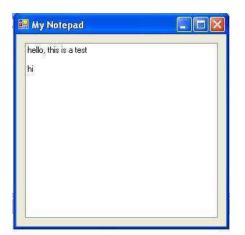
🕮 My Notepad - Microsoft	Visual Basic 2008 Express Edition			F X
File Edit View Project	Build Debug Data Format Tools Window Help			
1 🗃 🗃 🖬 • 🖬 🗿  X		2 2 2 2 2	3 💷 🗉	
Toolbox 🗸 🕈 🗙	Form1.vb [Design]* Start Page	Solution Explorer		• Ț ×
PrintPreviewControl				
	🔡 My Notepad	My Notepad		
Process		🦳 🤤 My Projec		
ProgressBar	0	Form1.vb		
PropertyGrid	Nonnecessite and a second s			
RadioButton				
RichTextBox				
🛃 SaveFileDialog				
🎾 SerialPort				
🚟 ServiceController		Solution Explore	# 🛐 Data Sourc	ces
SplitContainer		(3) (C)		
+ + Splitter		Propurties		• 4 ×
🔚 StatusStrip		TextBox1 System	n.Windows.Form	s.Text 🔹
TabControl		8≣ 2↓ 💷 🖋	(iii)	
📰 TableLayoutPanel 📄		🗄 Size	268, 20	~
abl TextBox		TabIndex	0	
🔅 Timer		TabStop	True	
💷 ToolStrip 🦰		Tag Text		-
ToolStripContainer		TextAlign	Left	~
🖳 ToolTip		Text		
□ □ □ □ □ □ □ □ □		The text associate	d with the contro	d,
TreeView				
Ready	廿 12,	12 I	268 × 20	
🛃 start 🔰 📴 My N	otepad - Microso		) 😻 🗶 🚇 🔅	50 AM

Drop a text box on the form.

🖳 My Notepad - Microsoft	Visual Basic 2008 Express Edition			
File Edit View Project	Build Debug Data Tools Window Help			
1 🚰 🗃 • 🖬 🖉 🐰	<b>B B B E E 9</b> • <b>0</b>	🛯 🗐 (J 🖄	2 2 2 2 2	3 🖂 🖕
Toolbox 🗸 🗣 🗙	Form1.vb [Design]* Start Page	<b>▼</b> ×	Solution Explorer	• ‡ ×
PrintPreviewControl				
RintPreviewDialog	🖷 My Notepad		My Notepad	
Process	0		My Project	
ProgressBar			- Form1.vb	
PropertyGrid	0			
RadioButton				
🛃 RichTextBox				
📰 SaveFileDialog				
🎾 SerialPort				
ServiceController			Solution Explorer	Data Sources
SplitContainer			9	
+ + Splitter			Properties	<b>~</b> ‡×
🔚 StatusStrip			1	Windows.Forms.Text 🝷
TabControl			2↓ 🗉 🖋	
TableLayoutPanel			🗄 MaximumSize	0, 0 🔥
abl TextBox			MaxLength	32767
🔅 Timer				0,0
ToolStrip			Modifiers Multiline	Friend 🔤
ToolStripContainer			PasswordChar	v late
Le ToolTip			Multiline	
0- TrackBar			Controls whether th	
TreeView			control can span mo	re than one line.
Ready				
🛃 start 📄 🖳 My No	otepad - Microso		× ()	😼 🗐 👰 3:50 AM

The problem with text boxes are that you can only use it to write a single line. To solve this, modify the text box property **Multiline** to **True**. This will allow you to write multiple lines in the text box and modify its height. Resize the text box to take the full size of the window





Run the application, and try to write some text. You can see it works fine.

hello, this is a test hi	

Next try to resize the window. Now you see there is a problem. The text box does not resize itself to match window size.

💌 My Notepad - Microsoft	Visual Basic 2008 Express Edition		- B 🗙
File Edit View Project	Build Debug Data Tools Window Help		
1 🚰 🗃 • 🖬 🗶 🐰	🖻 🚨 🗟 🗏 🗄 😫 🔊 • (* • 🖉 • 🖳 🕨	1 🛛 🖻 🎑 🖄	
Toolbox 🗸 🗣 🗙	Start Page Form1.vb [Design]	• ×	
😑 All Windows Forms 👩			
Pointer	💀 My Notepad		My Notepad
🐺 BackgroundWorker		And and a second se	📕 🚰 My Project
📅 BindingNavigator	0	De	Form1.vb
🚏 BindingSource			
ab Button			
CheckBox			
🚉 CheckedListBox			
🔊 ColorDialog			
📑 ComboBox	9	¢.	Termina - 1 - 1
📓 ContextMenuStrip			Solution Explorer
📁 DataGridView			Properties 🔶 🗣 🗙
🖃 DataSet			TextBox1 System, Windows, Forms, Text -
📷 DateTimePicker			2 2↓ □ 🖋 🖾
🛐 DirectoryEntry			Construction of Construction o
👯 DirectorySearcher			AccessibleDescript AccessibleName
C DomainUpDown			AccessibleRole Default
ErrorProvider			AllowDrop False
🔄 EventLog			Anchor Top, Left 💌
👼 FileSystemWatcher			AutoCompleteCust
FlowLayoutPanel			Anchor
📑 FolderBrowserDialog			Defines the edges of which a certain control
🔄 FontDialog 🛛 💌			
Ready			
🥙 start 🔰 🖳 My No	stepad - Microso		🦀 💓 🥵 🧶 🚇 3:51 AM

To solve this select the text box and change the **Anchor** property. This property specifies how should the text box resize itself when its parent container resize itself (the window). The anchor side specifies how far the edge of the control should be from the border of the outer window (or any other control). Specifying the Anchor to be **Top**, **Left**, **Right**, **Bottom** means that whenever the size of the window changes, the text box changes itself so that its sides keeps the same distance from window border. Try changing the anchor settings and test the application and resizing the window.

🖾 My Notepad - Microsoft	Visual Basic 2008 Express Edition			
File Edit View Project	Build Debug Data Tools Window Help			
1 🚰 🗃 • 🖬 🗿  X	B B B Ξ 2 9 · C · J · B I	) II II 🖼 🗐 🖓	2 2 2 2 3	- EE -
Toolbox 😽 🗣 🗙	Start Page Form 1.vb [Design]	▼ ×	Solution Explorer	• ₽ ×
📆 DirectorySearcher	-			
DomainUpDown	🖳 My Notepad		🛐 My Notepad	
ErrorProvider	0		🦳 📴 My Project	
EventLog			🔚 Form1.vb	
👼 FileSystemWatcher 🚽				
FlowLayoutPanel				
🔄 FolderBrowserDialog				
🔚 FontDialog				
CroupBox		L		
F1 HelpProvider		Ϋ́.	Solution Explorer	Data Sources
K HScrollBar			1. N.	
🗊 ImageList			Properties	≁ Ū ×
A Label			TextBox1 System.	Windows,Forms,Text 💌
A LinkLabel			2↓ 🗉 🖋	
E ListBox			AccessibleDescrip	
222 ListView			AccessibleName	
#- MaskedTextBox			AccessibleRole	Default
🚊 MenuStrip			AllowDrop	False
			Anchor AutoCompleteCus	Top, Bottom, Lef
MenuStrip Version 2.0.0.0 from Micros	soft Corporation		Anchor	
.NET Component	and the second		Defines the edges of	the container to
	nds and options grouped by		which a certain conti	rol is bound. When a
Ready functionality.			e.le	
				all de la constant
Start My No	tepad - Microso		<b>∞</b> ⊎	💕 🔍 👰 3:52 AM

Next we start working with menus. Search for a control called **MenuStrip** and drop it on the window.

🖾 My Notepad - Microsoft Visual Basic 2008 Express Edition		
File Edit View Project Build Debug Data Format Tools Window Help		
🕼 😂 • 🔜 🗶   X 📭 🖄   N   Ξ 😫   🤊 • (* • ) 💷 • R   ▶   I   Ξ 📬 📮 📬	2 2 2 3	🖅 🖕
Toolbox 🗸 🗘 X Start Page Form1.vb [Design]*	Solution Explorer	• ↓ ×
😨 DirectorySearcher	🖺 🗿 🛃 🗉	
🖸 DomainUpDown	My Notepad	
A Every Droutiday	👘 🔤 My Project	
Vertexpected by the second sec	Form1.vb	
FileSystemWatcher		
FlowLayoutPanel		
FolderBrowserDialog		
FontDialog		
Time GroupBox		
FI HelpProvider	Solution Explorer	Data Sources
KIZ HScrollBar		
🗊 ImageList	Properties	<b>→</b> ‡ ×
A Label	MenuStrip1 System	n. Windows. Forms. Me 💌
A LinkLabel	∄ <u>2</u> ↓ 🔲 🖋	100
E ListBox	AccessibleRole	Default 🔨
222 ListView	AllowDrop	False
# MaskedTextBox	AllowItemReorder	False
😫 MenuStrip	AllowMerge	True
MessageQueue	Anchor	Top, Left
MonthCalendar	AutoSize	True 🞽
with NotifyIcon	Anchor Defines the edges of	the container to
		ol is bound, When a
	14	
Ready to 0,	) ∓‼12	92 x 24
Start 🖳 My Notepad - Microso	🖉 🔍 🤅	💕 🧐, 📵 - 3152 AM

The control appears under the window as **MenuStrip1** which is the name of the control. The actual menu appears on the form itself. You can create the menu quickly by just start typing. Try typing **File**.

🤐 DirectorySearcher 📟		· · · · · · · · · · · · · · · · · · ·
DomainUpDown	😸 My Notepad	🛐 My Notepad
ErrorProvider	File Type Here	🔄 My Project
EventLog		 E E Form1.vb
👼 FileSystemWatcher 🚽	Type Here	
FlowLayoutPanel		
🔄 FolderBrowserDialog		
FontDialog		
( <sup>XV</sup> ) = 0		

Notice that when you write a menu entry, you can extend it horizontally, and vertically.

DirectorySearcher		
DomainupDown     ErrorProvider     ErventLog	He Type Here	My Notepad 
FileSystemWatcher	New Type Here	
FolderBrowserDialog		

Under the file menu add the New, Open, Save, and Close menu items.

Next you will write the code to handle the events for these menu items. Now in the workspace just double click the **New** menu item.

File Edit View Project	Build         Debug         Data         Tools         Window         Help           Image: Second seco	
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.	Form1.vb* Start Page Form1.vb[Design]*	
		Solution Explorer Data Sources

As with the previous tutorial you get the code editor. Write down the following:

```
TextBox1.Text = ""
```

This will clear the text box and allow you to write next text. Add the following code to the close menu item.

End

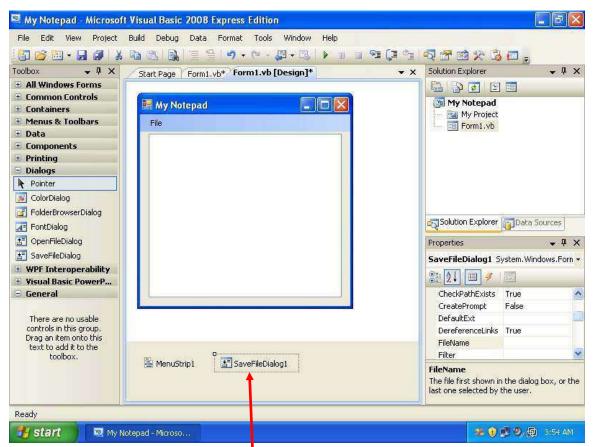
This will close the application. The code should look like this:

```
Private Sub NewToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles NewToolStripMenuItem.Click
    TextBox1.Text = ""
End Sub
```

```
Private Sub CloseToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles CloseToolStripMenuItem.Click
End
End Sub
```

Notice that I haven't change controls names (the menu items name property). You can change the name to a more friendly one as we did in the previous tutorial. Also the NewToolStripMenuItem.Click and CloseToolStripMenuItem.Click are on the same line as Handles .

Run the application, try to write some text, then select File->New. Then try File->Close. Next we see how to save text.



Search for a control called **SaveFileDialog** and drop it on the form. You won't see any visual change to the form itself. You can see there is save file dialog available under the window. This control allows you to specify where to save files on the file system and it looks like the window below:



Let us modify the filter property of the dialog. Click on **SaveFileDialog1** to display its properties.

Now change the **Filter** property to be like that:

Set it to be: Text files |\*.txt

This property prevents the user form mistakenly saving the file in formats other than text. The **Text files** part is displayed to the end user, while the **\*.txt** is used to filter the files and make sure you only select or overwrite text (ending with .txt) files.

The next step is to write the code to save the text written in your application into the disk. Add the code in the handler of the **Save** menu item by double clicking it then typing:

```
SaveFileDialog1.ShowDialog()
If SaveFileDialog1.FileName = "" Then
    Exit Sub
End If
' this part saves the file
FileSystem.FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
FileSystem.Print(1, TextBox1.Text)
FileSystem.FileClose(1)
```

The code is explained as follows:

SaveFileDialog1.ShowDialog(): this instructs the application to show the Save dialog on the screen so that you can specify the file name. When you select a name for the file, you can get it from the control using the **FileName** property.

```
If SaveFileDialog1.FileName = "" Then
    Exit Sub
```

## End If

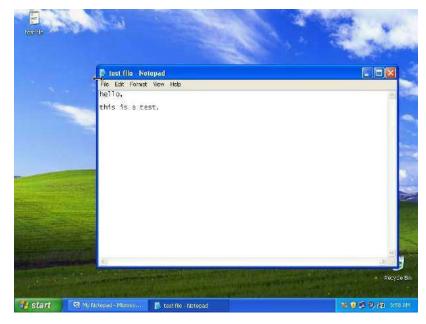
The code above checks if the **FileName** is not specified, in other words if you pressed the cancel button when the dialog is shown, the **FileName** will be empty. So in this case no saving will happen and the execution path of the code exits the subroutine.

```
FileSystem.FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
FileSystem.Print(1, TextBox1.Text)
FileSystem.FileClose(1)
```

This part saves the text into the file. The SaveFileDialog1.FileName property allows you to get the name of the file. TextBox1.Text gets the text from the text box.

🖳 My Notepad (Ru	and the second second second			tion				
Form1.vb Form	Project Build De <b>Save As</b> Save in:	bug Tools W			00	∙		⇔ <b>■ ₽ ₽ ₽</b> → ╄ ×
Public File	My Recent Documents Desktop My Documents My Computer	My Documen My Computer	ŕ				ad iett vb	Jata Sources ↓ ↓ × ↓
Watch Let Name	My Network	File name: Save as type:	test Text files				Save Cancel	
Ready	e Window				Ln 23	Col 1	Ch 1	INS
🛃 start 🔰 🚺	💐 My Notepad (Rur	ining 🔛 M	1y Notepad				🗶 🖲 🖉	🧐 👜 3:58 AM

Run the application, write some text, and then select **File->save**. Save your file into the desktop. You can open this text file via the standard notepad as shown below:



🖾 My Notepad - Microsof	Visual Basic 2008 Express Edition	
File Edit View Project	Build Debug Data Format Tools Window Help	
🔚 🚅 🖘 🖬 🗿 🐰	BBB BB E 2 9 • C • Ø • B  ▶ 11 I 9 (I G)	🗟 🚰 🐋 🛠 💪 🖾 🖕
Toolbox 🗸 🗣 🗙	Start Page Form1.vb* Form1.vb [Design]*	Solution Explorer 🚽 🗸 🗸
🗄 All Windows Forms		
🗄 Common Controls	🔡 My Notepad	My Notepad
🗄 Containers		My Project
🗄 Menus & Toolbars	File	Form1.vb
🗄 Data		
E Components		
Printing     Dialogs		
Pointer		
ColorDialog		
FolderBrowserDialog		
		Solution Explorer
FontDialog		
🔠 OpenFileDialog		Properties 🗸 🗣 🗙
🛃 SaveFileDialog		OpenFileDialog1 System.Windows.Forr -
WPF Interoperability		81 2↓ □ 🥖 🔟
Visual Basic PowerP		And the second statement of th
🕞 General		CheckFileExists True
There are no usable		DefaultExt
controls in this group.		DereferenceLinks True
Drag an item onto this text to add it to the		FileName OpenFileDialog1
toolbox.		Filter
	E MenuStrip1 5 SaveFileDialog1 5 OpenHeDialog1	FileName The file first shown in the dialog box, or the last one selected by the user.
Ready		
📲 start 📃 🖾 My N		🐲 💽 🕵 🧶 📵 - 3:59 AM
Start Start	otepad - Microso	

Now we work on the **File->Open** part. Search for the **OpenFileDialog** control and drop it on the form. Change **FileName** property and remove all the text from it. And change the filter property to: **Text files\*.txt**. and finally go to the **File->Open** event handler by double clicking the Open menu item, And then add the following code:

```
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName = "" Then
    Exit Sub
End If
' this part loads the file
Dim Tmp As String
Tmp = ""
FileSystem.FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
Do While Not FileSystem.EOF(1)
    Tmp = Tmp & FileSystem.LineInput(1)
    If Not FileSystem.EOF(1) Then
    Tmp = Tmp & Chr(13) & Chr(10)
    End If
```

```
Loop
FileSystem.FileClose(1)
TextBox1.Text = Tmp
```

The **OpenFileDialog** works very similar to the **SaveFileDialog** so there is no much to explain about it. After testing this part, we will work with the font. We want to add the ability to change the size and type of the font in the text box to make it easier to read. So add menu entries **Edit**, and **Font**, then add the **FontDialog** control.

🖳 My Notepad - Microsof	t Visual Basic 2008 Express Edition	
File Edit View Project	Build Debug Data Format Tools Window Help	Solution Explorer Jaka My Notepad My Project Form1.vb
controls in this group. Drag an item onto this text to add it to the toolbox.	볼 MenuStrip1 호 SaveFileDialog1 호 OpenFileDialog1	
Ready		
	lotepad - Microso	🐲 💓 🕵 🧶 📵 4:05 AM

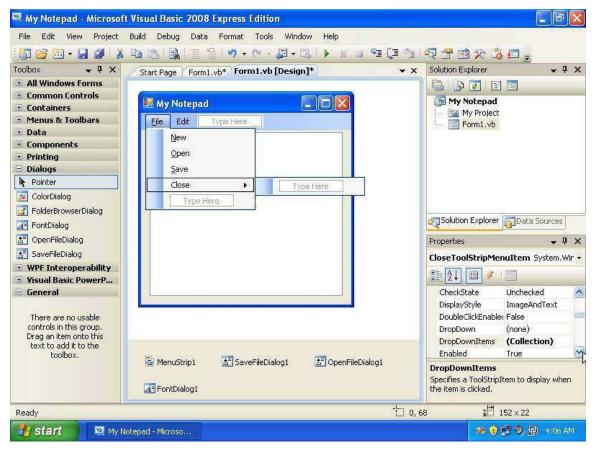
Next double click the **Font** menu item to built its handler.

FontDialog1.ShowDialog()
TextBox1.Font = FontDialog1.Font

Try to run the application and select the Font menu item



Now let us improve out menu now. Right now the menu cannot be accessed using the keyboard. You can make it accessible using the Alt key and some letter. For example click once on the file menu item. Now you can change the text displayed on the menu item. Modify it to be &File. This will have the effect of adding the ability to access the menu item using the Alt+F combination. Perform the same operation for other menu items to be &New, &Open, &Close, &Save, &Edit. The letter after the & symbol is always the access key.



Try to run the application, then pressing Alt+F, then O to show the open file dialog as a test to see if it works.

🛯 My Notepad - Microsof	t Visual Basic 2008 Express Edition	<b>_</b> 2 ×
File Edit View Project	Build Debug Data Tools Window Help	
i 🛅 🧉 • 🖬 🖉 i 🐰	B B B E E P • P • B • B ■ E E E E	💀 🕾 🖄 🖾 🖕
Toolbox 👻 🕈 🗙	Start Page Form1.vb* Form1.vb [Design]*	Solution Explorer 🚽 🗸 🗙
All Windows Forms		
🗄 Common Controls	💀 My Notepad 📃 🗖 🗙	🛐 My Notepad
Containers		🚽 🔤 My Project
Menus & Toolbars     Data	Elle Edit Type Here	Form1.vb
± Components	New	
Printing	Qpen	
Dialogs	Save	
Pointer	Close   Type Here	
ColorDialog	Type Here	
FolderBrowserDialog	Type Here.	
FontDialog		Solution Explorer Data Sources
		Properties • 4 ×
3 SaveFileDialog		
WPF Interoperability		CloseToolStripMenuItem System.Wir -
Visual Basic PowerP		2: 2↓ □ 🥖 🖾
🖯 General		
		RightToLeft No
There are no usable		RightToLeftAutoMi False
controls in this group. Drag an item onto this		ShortcutKeyDispla
text to add it to the		ShortcutKeys None K
toolbox.	MenuStrip1 🛃 SaveFileDialog1 🛃 OpenFileDialog1	Modifiers:
	Turanda Turana ana mananda Turana ana ana ana ana ana ana ana ana ana	Ctrl Shift Alt
	FontDialog1	iti Key:
Ready		Reset
🐉 start 🔰 🖳 My N	otepad - Microso	🦀 💓 🗾 🧐 👘 4:07 AM

Next we will add shortcut keys. This is easy, click the menu item once to show its properties. Change the **ShortcutKeys**, by marking the **Ctrl** check box and selecting the **X** button for the **Close** menu item.



Repeat the same step for other menu items



Run the application and press Ctrl+O and you will see the Open File Dialog directly.

💌 My No	otepad (Runnin	g) - Microsoft Vis	ual Basic 2008 E	xpress Edition				X
Fil BR M	iy Notepad	í		ols Window Help				
100 m			+ (*	• 📮 • 🖳 🖡 🕨	u 🖬 🖼 🗐 🖻	1 🖏	😁 🖄 📯 🙄 🖙	- 🗔 🗔 🖕
<b>File</b>	New Ctrl+1					• ×	Solution Explorer	<b>→</b> ‡ ×
						~		
	Open Ctrl+C						🛐 My Notepad	
	Save Ctrl+!	5					My Project	
	Close Ctrl+3					100	ear onni.vo	
		_						
						-	Colution Eveloper []	Dela casa de
						<u>×</u>		
	enuStrip1 🖄	SaveFileDialog1		FontDiald	ia1		Properties	≁ ‡ ×
		1221200000000000					EditToolStripMenu	tem System.W 🝷
							2 2 II 🦻	
Watch						ąх	☑ (ApplicationSetti	~
Name		Value			Туре	* *	(Name) Ec AccessibleDescri	litToolStripM
110000		19/96			1768		AccessibleDescri	
								efault
							Alignment Le	ft 🗾 🔽
						100	(ApplicationSetting	
Watch	Immediate Win	dow					Maps property setting application configurati	s to an on file.
	annicenace with	30m)				//	19	
Ready			-		力。	41, 2	1 <sup>1</sup> 37 × 2	
🛃 Sta	int 🔍 M	Notepad (Running	🔛 My Notepad					🥠 👜 4:09 AM

Next try to make dividing lines between menu items. To do so, write the text – (the minus sign) in the text part of the menu item. Notice that you can drag and change the position of the menu items, so try to position those dividing lines to be something similar to the above.

Now we start defining the **Copy**, **Cut**, and **Paste** commands. So, add these menu items under **Edit**, and write the code for each one of these:

1- for the copy

```
Clipboard.SetText(TextBox1.SelectedText)
```

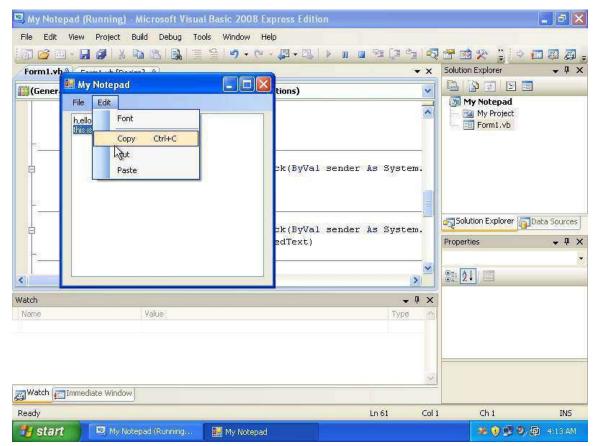
2- for the cut

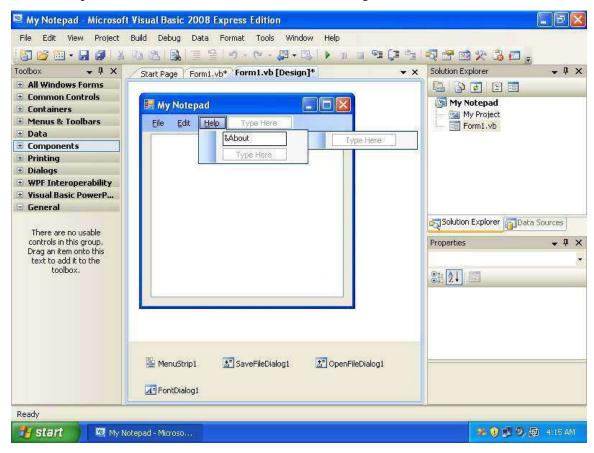
```
Clipboard.SetText(TextBox1.SelectedText)
TextBox1.SelectedText = ""
```

3- for the paste

```
TextBox1.SelectedText = Clipboard.GetText
```

Run the application and test the copy, cut, and paste.





Now let us just add an About menu item, under the help menu item.

## Add the following code in the **About** menu item.

MsgBox("This application is a test of making a notepad application out of VB.NET", MsgBoxStyle.OkOnly, "About My Notepad")

The final code should look like this:

```
Public Class Form1
Private Sub NewToolStripMenuItem_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles _
    NewToolStripMenuItem.Click
    TextBox1.Text = ""
End Sub
```

```
Private Sub CloseToolStripMenuItem_Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         CloseToolStripMenuItem.Click
        End
   End Sub
   Private Sub SaveToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         SaveToolStripMenuItem.Click
       SaveFileDialog1.ShowDialog()
        If SaveFileDialog1.FileName = "" Then
            Exit Sub
       End If
        ' this part saves the file
        FileSystem.FileOpen(1, SaveFileDialog1.FileName,
OpenMode.Output)
       FileSystem.Print(1, TextBox1.Text)
       FileSystem.FileClose(1)
   End Sub
   Private Sub OpenToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         OpenToolStripMenuItem.Click
        OpenFileDialog1.ShowDialog()
        If OpenFileDialog1.FileName = "" Then
           Exit Sub
       End If
        ' this part loads the file
        Dim Tmp As String
        Tmp = ""
        FileSystem.FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
        Do While Not FileSystem.EOF(1)
            Tmp = Tmp & FileSystem.LineInput(1)
            If Not FileSystem.EOF(1) Then
                Tmp = Tmp \& Chr(13) \& Chr(10)
           End If
       Loop
        FileSystem.FileClose(1)
       TextBox1.Text = Tmp
   End Sub
```

```
Private Sub FontToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         FontToolStripMenuItem.Click
       FontDialog1.ShowDialog()
        TextBox1.Font = FontDialog1.Font
    End Sub
    Private Sub CopyToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         CopyToolStripMenuItem.Click
        Clipboard.SetText(TextBox1.SelectedText)
    End Sub
    Private Sub CutToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         CutToolStripMenuItem.Click
        Clipboard.SetText(TextBox1.SelectedText)
        TextBox1.SelectedText = ""
    End Sub
    Private Sub PasteToolStripMenuItem Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs) Handles
         PasteToolStripMenuItem.Click
       TextBox1.SelectedText = Clipboard.GetText
    End Sub
    Private Sub AboutToolStripMenuItem Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs) Handles
          AboutToolStripMenuItem.Click
     MsgBox("This application is a test of making a notepad application
          out of VB.NET", MsgBoxStyle.OkOnly, "About My Notepad")
    End Sub
End Class
```

Notice that some lines are too long so I divided them to multiple lines. In visual basic, writing a command on multiple line requires you to add the underscore symbol (\_) to tell the compiler that these two (or more) lines are actually one long line. Try to run the application and test it.

**Chapter 5: Understanding variables** 

#### **Understanding variables**

When a computer processes information or data, that data should be placed in its memory first, then it performs different operations on that. You application as well should process the information in memory. To accomplish that the you allocate memory by defining variables. Simply, a variable is small piece of memory that allows the program to process data within it.

To define a variable in VB.NET you use the following format:

#### Dim MyInt As Integer

Where the Dim tells the computer that you are going to define a variable. Next you write variable name (in this case it is MyInt) and finally the data type As Integer which tells the computer you are going to perform integer operations on the data in that variables.

So, in order to try out integer variables, create a simple windows application, and place a button on your form, and modify its event so that it looks something similar to this:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Button1.Click
    Dim MyInt As Integer
    MyInt = 55 + 4
    MsgBox(MyInt)
End Sub
```

Try this code out... you see that MyInt stores the result of the operation. You can place any type of operation like +,-,\*,/ and more... but for now let us try to create another variable and use it with this one. Modify the code to be something like this:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Button1.Click
Dim MyInt As Integer
Dim A As Integer, B As Integer
A = 10
B = 30
MyInt = A + B + 2
MsgBox(MyInt)
```

#### End Sub

In this case we are using 3 variables (A,B,MyInt). We compute using the values inside the A,B and store the result in MyInt. So basically any of the following statements is valid:

MyInt = 22	places value 22 in MyInt
MyInt = A	copies the value of A into MyInt (10)
MyInt = B	copies the value of B into MyInt (30)
MyInt = A + B	adds A,B and places the result in MyInt (40)
MyInt = A - B	subtructs B from A (-20)
MyInt = A + A + B + 2	adds double value of A, value of B and 2 to the total
	(52)
MyInt = B*A-A	multiply A by B then subtructs A (290)
MyInt = A/2	divides A by 2 (10)
MyInt = B/A	divides B by A (3)
MyInt = MyInt + 1	gets the value of MyInt (usually 0) and adds one to
	that (total is 1)

and you can come up with any form of statement that helps you solve your problem.

Now we start understanding what is the data type. Each variable can be one data type. The data type tells the computer what you are going to process in this variable. For example **Integer** means you are working with whole numbers (like 30,40,55) and not fractions. So if you try out:

MyInt = 3.4

What you get is the value 3. That is because this data type can not store floating point numbers. To solve this issue you should use **Singe**. To make things clear try to execute the following code:

Dim I As Integer
I = 22 / 7
MsgBox("the PI (int) :" & I)

```
Dim S As Single
S = 22 / 7
MsgBox("the PI (single) :" & S)
```

When you run this code, you will see that it give you 3 as the value of PI (the integer case), then it gives you a value of 3.142857 for PI (the single case).

So why do you use integers and singles? Why not use single all the time? The answer is performance and storage. The computer processes integer operations faster than that of floating point numbers. And also some data types takes less storage than others. Because of that if you are going to develop an application that performs lots integer operations it makes sense to use integers to speed things up. In the end you select variable types depending on the nature of your problem.

There are similar data types to Integer and Single. These are Long (to store integers) and Double(to store floating point numbers). The difference between these and the previous ones is that they can represent wider range of numbers. To demonstrate this try out the following code:

```
MsgBox("integer " & Integer.MaxValue)
MsgBox("long " & Long.MaxValue)
MsgBox("single " & Single.MaxValue)
MsgBox("double " & Double.MaxValue)
```

The Integer.MaxValue gets the maximum value that an Integer can store. The same is true for other data types. Of course better representation requires more memory and also more processing time. Another example to demonstrate this is by calculating the value of PI using Double and Single. Try out the following code:

```
Dim I As Integer
I = 22 / 7
MsgBox("the PI (int) :" & I)
Dim S As Single
S = 22 / 7
MsgBox("the PI (single) :" & S)
```

```
Dim D As Double
D = 22 / 7
MsgBox("the PI (double) :" & D)
```

I did not write the Long data type here, but you can try it. It gives the same result as that of the integer case because Long data type can only store non-floating point values. Another type of variables is the ones used to store complete statements, the String data type.

```
Dim Str1 As String
Dim Str2 As String
Str1 = "hello"
Str2 = " my friend"
MsgBox(Str1)
MsgBox(Str2)
```

As you can see, this data type stores letters and symbols. And actually you can do a number of operations on these. The simplest is to use the & operator to combine two strings.

```
Dim Str1 As String
Dim Str2 As String
Dim Str3 As String
Str1 = "hello"
Str2 = " my friend"
Str3 = Str1 & Str2
MsgBox(Str3)
```

You can see that Str3 now holds the complete statement "hello my friend". More into string data type will come with time, but here we are focused on the very basics of variables.

Another important data type is the Boolean data type. This one is used to evaluate logic operations and it can only store True and False.

```
Dim B1 As Boolean
Dim B2 As Boolean
Dim B3 As Boolean
Dim B4 As Boolean
B1 = True
B2 = False
B3 = 88 > 10
B4 = "asmith" > "john"
MsgBox("b1 is:" & B1)
MsgBox("b2 is:" & B2)
MsgBox("88 > 10 is: " & B3)
MsgBox("asmith comes after john is:" & B4)
```

As you can see above B3 will check if 8 is greater than 10, and if so, it stores the value True, otherwise it stores False. B4 here shows how you can compare two strings. It checks to see if **asmith** comes alphabetically after **john** (which is the meaning of > sign in string comparison), and obviously this is not correct, so the value of B4 is false.

Finally you have the Date data type which is used to store the time and date. You can test it using the example below:

```
Dim D1 As Date
Dim D2 As Date
Dim D3 As Date
Dim D4 As Date
D1 = Now
D2 = Now.Date
D3 = "8:10:22 AM"
D4 = "2009-03-01"
MsgBox(D1)
MsgBox(D2)
MsgBox(D3)
MsgBox(D4)
```

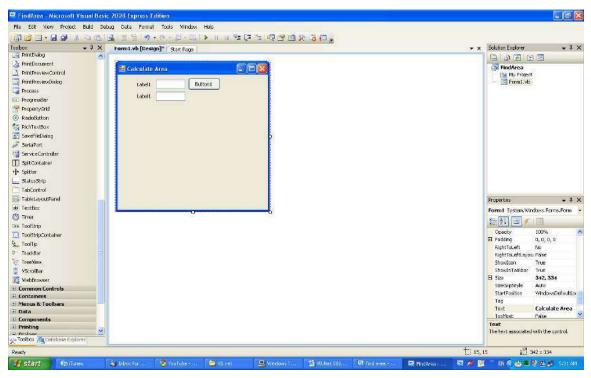
The **Now** in the above example is used to get current system date and time. **Now.Date** is used to get system date alone.

Chapter 6: Variables again, group box, list box

## Variables again, group box, list box

This chapter is about using Group boxes, and little about variables and list boxes. We will create a small application to find the area of squares, rectangles, and triangles. So let us proceed by creating a new windows application project, name it "find area" and save it.

Next set the form title to "Calculate Area", I assume now you already know how to do that. Next place a two labels, two text boxes, and a button on the form to look something like this:



Now change the first label to be "length:", second label to be "area:", and the button text to be "compute". (to do that just click the control and change the text property". You should get something like this:

Calculate An	ea	
ength:		

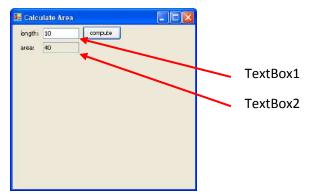
The user enters the length of the square in the first text box, and presses the compute button, and he/she should get the area in the second text box. The second text box should not be modified by the user. To do so, change the ReadOnly property of the second text box to be "true".

🔡 Calculate Area		
length: area:	compute	

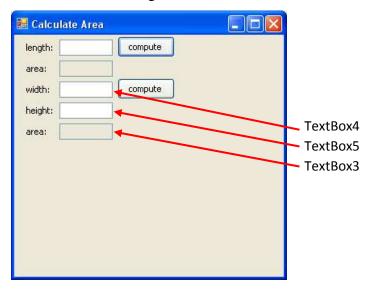
To write the code that gets the length, calculate the area, and display it, first check the name of the text boxes because you will need these. Next double click the compute button to write its event.

```
Dim L As Double ' define the length variable
Dim A As Double ' define the area variable
L = TextBox1.Text ' get the length from the window.
A = L * L ' calculate the area
TextBox2.Text = A ' display the area in the second
' text box
```

Keep in mind that TextBox1, and TextBox2 are the names of the controls used to get the data and display the result. And also remember you can change them if you want to. Now run the application, enter a value of 10 and press compute.



Chang the length and press compute again and see the new results. Next we will do something similar for the rectangle. Place three labels, three text boxes, and a command button to be something like this:



Note that control names here are just the way I got them, you might get the order differently, or you might want to rename the controls. Next double click the compute button for the rectangle and write the following code.

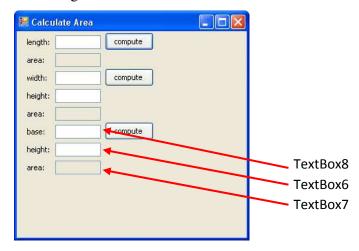
Dim	W	As	Double		define	the	width	variable
Dim	h	As	Double	1	define	the	height	t variable
Dim	А	As	Double		define	the	area v	variable

W = TextBox4.Text	' get the width
h = TextBox5.Text	' get the height
A = W * h	' compute the area
TextBox3.Text = A	' dsplay the area

Now run the application, and test it.

length:	0	compute	
area:			
width:	10	compute	
height:	20		
area:	200		
	1		

The last part is the triangle. Repeat what you did with the rectangle and you should get something similar to this:



And next set the code to compute the area to be:

```
Dim base As Double ' define the base
Dim Height As Double ' define the height
Dim Area As Double ' define the area
base = TextBox8.Text ' get the base
Height = TextBox6.Text ' get the height
```

```
Area = 0.5 * base * Height ' compute the area
' (1/2 x base x height)
TextBox7.Text = Area ' display the result.
```

and check the code. It works. Now look at our window.

length:	compute	
area:		
width:	compute	
height:		
area:		
base:	compute	
height:		
area:		

It does not look good actually and a little bit confusing because there are three compute buttons, and the fields are mixed. The solution to this is to use GroupBox control. Now search with the controls and add a GroupBox control on the form.

length:	compute	
area:		
width:	compute	
height:		
atea;		
base:	compute	
height:		
area:		
Group@ox1		

Next select the labels, textboxes and the button that is used for the square, and drag them so that they fit inside the group box. Click then on the group box and set its text property to be square. You should get something similar to this:

Square		
length	compute	
area:		
width:	compute	
height:		
area:		
base:	compute	
height:		
area:		

Notice that you might need to move the controls a little bit to make them fit. The group box can be moved using the arrows symbol that appears when you click inside it.

	0	
length:		compute
area:		ų
		Ö

Repeat the same thing for the rectangle and triangle to get something like this:

🔡 Calculate Area	
Square length:	compute
Rectangle width:	compute
area:	]
<ul> <li>Triangle</li> </ul>	
base:	compute
height:	
area:	

As you can see group box allows easy movement of controls on the form at design time and also gives a better view of your application. Now we start working with list box. It is used to view a list (which is no surprise). What we want to do here is displaying each area being computed on the right side in such a way the previous results and computations are displayed as well. To do so, add a group box to the right of your window and put inside it a listbox control. You should have something like this:

	ng Data Format Tools Window Help
🛐 💕 🛄 - 🖼 🕼 🙏 🐴 🛝 📑 Teoloox — 🗸 4 X	\] <u>∃</u> ∃   ୬ · ୯ - ∅ - ₿ · ▶ ॥ ⊒ 역 [ 4 ] 역 <b>전 2</b> 2 2 <b>.</b> .
Fill CombcBox 🔷 🕅 R: ContextMenuStrip J-DataSildVew	Start Page Form 1.vb* Form 1.vb [Design]*
DataSet DataS	Square     History       length:     compute       area:     ListBox1       Hectangle     compute       width:     compute       height:     0       Triangle     corrpute       height:     0       area:     0
HScrolBar     ImagsList     A Label     LinLabel     Utstox	

The ListBox1 that appears here does not appear while the application is running. What you see now is the name of the listbox control. Now we modify the code of the square so that we add the area computed into the list box. Double click on the compute button of the square, and modify its code to be like this:

Dim L As Double	' define the length variable
Dim A As Double	' define the area variable
L = TextBox1.Text	' get the length from the window.
A = 4 * L	' calculate the area
TextBox2.Text = A	' display the area in the second text
	' box
ListBox1.Items.Add(	"the area of the square is:" & A)
	' add the item

ListBox1 is used to communicate with the list, .Items is used to access the list of items that it is displaying (which is at the beginning of execution is empty), .Add is used to add the information or message into the Items of the list box. "the area of the square is:" & A is evaluated first, and then the result is added into the list box items.

For the rectangle, modify its compute button to be something like this:

Dim W As Double ' define the width variable

```
Dim h As Double ' define the height variable
Dim A As Double ' define the area variable
W = TextBox4.Text ' get the width
h = TextBox5.Text ' get the height
A = W * h ' compute the area
TextBox3.Text = A ' dsplay the area
Dim S1 As String
Dim S2 As String
S1 = "The area of rectangle is:" ' the message
S2 = S1 & A ' write the message then the area, and
 ' store the new message in S2
ListBox1.Items.Add(S2) ' add the message to the list box
```

The same method to add item to the list box is called, but the way we generate the message is a little bit different. Finally the rectangle code should be:

Now you see also that the message is generated in a different way again. So you can choose the way that best suits you. Now run the application and calculate few areas.

🔡 Calculate A	rea		
Square length: 3 area: 12 Rectangle width: 20 height: 30 area: 600	compute compute	History the area of the square is:40 The area of rectangle is:600 The area of thangle is: 1000 the area of the square is:40 the area of the square is:12	
Triangle base: 40 height: 50 area: 1000	compute		

You should get something similar to this at run time. You can compute more than once and see that the list box keeps adding new results. Next we see how to make a clear button to clear the content on the form (for the listbox and the textboxes). Simply add a button on the form, you should get something similar to this:

🔡 Calculate Area		
Square	History	
length:	ListBox1	
Rectangle width:	compute	
height:		5
area:		
Triangle base:	compute	
height:		
area:		lear

And add the following code in the clear button's event:

TextBox1.Text = ""	1	clear	all	the	text	boxes
TextBox2.Text = ""						
TextBox3.Text = ""						
TextBox4.Text = ""						
TextBox5.Text = ""						
TextBox6.Text = ""						
TextBox7.Text = ""						
TextBox8.Text = ""						
ListBox1.Items.Clear()	1	clear	the	list	box	

As you can see text boxes are cleared by setting their text property to empty string "", while listboxes you just give a Clear command for them. Run the application, compute a number of areas, and finally hit clear and see how the listbox and all textboxes are cleared from the text.

So basically we worked just a little bit with variables, saw how to work with groupbox, and saw the basic operation of listboxes. More details will be given into listboxes, but for now we need to focus more on the programming aspect of the language.

**Chapter 7: IF statement** 

### IF statement

In VB.NET there are many forms for the IF statement. They all work by evaluating some expression and if the expression is correct (evaluated to true) then the code within the IF block is executed. Now check out the first simple form of IF statement

```
If expression Then
Statement
Statement
...
End If
```

The expression here is logical one. For example A>10, A<99, B>=A and so on. If the expression is correct, the statements inside the IF block get executed. The statement could be any valid VB.NET statement (even another IF statement). Now here is an example of the IF statement that always get executed:

```
If 10 < 100 Then
    ' display a friendly message
    MsgBox("You must see this message")
End If</pre>
```

Since 10 is always smaller than 100 the condition always evaluates to true and you will always see the message. Now change it to be:

```
If 10 > 100 Then
    ' display a friendly message
    MsgBox("You must see this message")
End If
```

The code within the block will never get executed. Now start a new project, put a button on the form and go to its event handler and add the following code:

```
Dim A As Integer
Dim B As Integer
A = InputBox("enter the value of A")
B = InputBox("enter the value of B")
```

```
If A > B Then
	MsgBox("A is greater than B")
End If
If A < B Then
	MsgBox("A is smaller than B")
End If
If A = B Then
	MsgBox("A is equal to B")
End If
```

The InputBox is a function that reads a value from the keyboard. So the program reads two numbers and check their status. Run the program and try different values for A and B to see how it works. Also debug the program (by pressing F10 to execute one statement at a time) and see how the code get executed internally.

Another variation of the IF statement is the IF ... ELSE. It has he following format:

```
If expression Then
Statement
Statement
...
Else
Statement
...
End If
```

The statements in black get executed when the expression is true, while the statements in red are ignored. However if the expression is evaluated to false then the statements in black are ignored while the statements in red are executed. To see how it works consider the following example:

```
If 10 > 100 Then
    ' this message is never displayed
    MsgBox("10 is greater than 100")
Else
    ' this message is always displayed
```

```
MsgBox("10 is smaller than 100, what a surprise!!!")
End If
```

The last form of the IF statement is the IF...ELSEIF... statement. Think of it as a multiple if statements combined into one. The form is as follows:

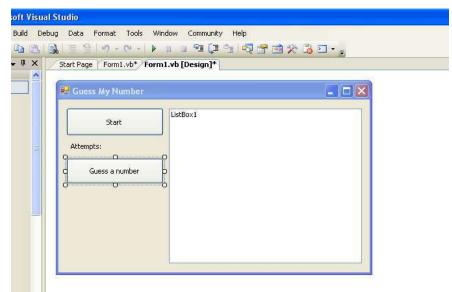
```
If expression1 Then
    Statement
    Statement
    ...
ElseIf expression2 Then
    Statement
    Statement
ElseIf expression3 Then
    Statement
    Statement
    ...
Else
    Statement
    Statement
    ...
End If
```

In this case if expression1 is evaluated to true, then its statements are executed and then the rest of the IF statement is ignored. If not, the expression2 is evaluated and its corresponding statements are executed and the rest of the checks are ignored... so check out the example below to have an idea about how it works:

```
If MyAge < 13 Then
    ' you must be a child
    MsgBox("Child")
ElseIf MyAge < 20 Then
    ' you are a teenager
    MsgBox("Hello Teenager")
ElseIf MyAge < 35 Then
    ' Your age is acceptable
    MsgBox("Hi there young man")
Else
    ' the person is old</pre>
```

```
MsgBox("Hello there old man")
End If
```

So basically this is how the if statement works. We will create a simple Number Guessing Game and see how the IF statement helps us to do it. So basically start a new project, and Create a form with two buttons, a list box and a label as shown below:



Now for the Start button's event write the following code:

```
Dim SecretNumber As Integer
Dim Attempts As Integer
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Randomize()
SecretNumber = Int(Rnd() * 100)
Attempts = 0
ListBox1.Items.Clear()
Label1.Text = "Attempts:" & Attempts.ToString
End Sub
```

The variables SecretNumber and Attempts are declared outside the subroutine so that their value will persist during program execution. The statements

```
Randomize()
SecretNumber = Int(Rnd() * 100)
```

Are used to generate a random number. The numbers are usually generated using some pattern. Each execution the same pattern of numbers appears. The first statement Randomize()makes sure that does not happen. The Rnd()function is used to generate a random number between 0 and 1. Multiply that by 100 you get a value between 0 and 100.

```
Attempts = 0
ListBox1.Items.Clear()
Label1.Text = "Attempts:" & Attempts.ToString
```

These statements resets the number of guessing attempts the play has made, and clears the listbox from previous attempts.

The code for the second button is:

```
Private Sub Button2 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim MyNumber As Integer
        Dim Tmp As String
        Tmp = InputBox("Enter a number between 1 and 100", "Guessing
game")
        If IsNumeric(Tmp) Then
            MyNumber = Tmp
        Else
            MsgBox("you should enter a number")
            Exit Sub
        End If
        If MyNumber = SecretNumber Then
            MsgBox("You Guessed the correct number", MsgBoxStyle.OkOnly)
        ElseIf MyNumber > SecretNumber Then
            ListBox1.Items.Add("you should enter a lower number")
            MsgBox("your guess is wrong")
        Else
            ListBox1.Items.Add("you should enter a higher number")
            MsgBox("your guess is wrong")
        End If
        Attempts = Attempts + 1
        Label1.Text = "Attempts:" & Attempts.ToString
```

End Sub

The code is explained as follows:

```
Dim MyNumber As Integer
Dim Tmp As String
Tmp = InputBox("Enter a number between 1 and 100", "Guessing game")
```

Here we define a number variable to store our guess in. We also need a string variable. This one will hold the value enter by the user so that we can check if it is a number of not (because the user can enter text value instead of a number).

```
If IsNumeric(Tmp) Then
    MyNumber = Tmp
Else
    MsgBox("you should enter a number")
    Exit Sub
End If
```

The IsNumeric is a function that is used to check if a string represent a number or not. So this part will assign the number inside Tmp into MyNumber if it is a proper number representation. Otherwise you get a message telling you about the error and the execution to the subroutine terminates because of the Exit Sub statement. Next:

```
If MyNumber = SecretNumber Then
    MsgBox("You Guessed the correct number", MsgBoxStyle.OkOnly)
ElseIf MyNumber > SecretNumber Then
    ListBox1.Items.Add("you should enter a lower number")
    MsgBox("your guess is wrong")
Else
    ListBox1.Items.Add("you should enter a higher number")
    MsgBox("your guess is wrong")
End If
```

This is the important part were we check the number against what the computer generated. If the numbers are a match then we display a message telling the user about

his guess. If not the user get a wrong guess message and the computer tells if you should guess a higher or lower number. Finally:

```
Attempts = Attempts + 1
Label1.Text = "Attempts:" & Attempts.ToString
```

Will only update the number of attempts.

So this is a very simple introduction to IF statements, and we will be using these more often in later chapters.

**Chapter 8: FOR statement** 

### FOR statement

Almost every language has some kind of looping statement (in case you don't know what that does, it allows the execution of a number of statements several times). In VB.NET there are a number of looping statements, these are REPEAT, DO and FOR. We will talk about the easiest of them all which is the FOR loop. The FOR loop is written like this:

```
For variable = Min To Max Step JumpStep
Statement
...
Next
```

The code will execute the statements between the For and Next parts by setting the variable to Min, increasing it by one every time until it reaches Max. To make things clear consider this example

```
For A = 1 To 10
    MsgBox("The value of A is:" & A)
Next
```

The result of executing the code above is ten message boxes telling you the value of A every time.

Now let us consider another example. Here you have a form with a textbox and a ComboBox. You select font size from the combo box and the text size changes accordingly.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
                      System.EventArgs) Handles MyBase.Load
   ' this part fills the combobox with the sizes of font that we
   ' can pick from
   Dim I As Double
   For I = 12 To 70
        ComboBox1.Items.Add(I)
   Next
End Sub
```

You notice two things in the example, first the loop does not start from 1, it starts from 12, you can start from any value you like, for example start from 283732, -12, 0, 88888, etc. Second the data type of the variable I is double. You can use Single, Double, Integers, Long... You are not restricted here.

If we want to display the numbers between 5 and 50 by adding 5 to the previous in each step then:

```
Dim Counter As Integer
For Counter = 5 To 50 Step 5
    MsgBox(Counter)
Next
```

Assume we need the values 0, 0.1, 0.2, 0.3, 0.4, 0.5... 1.0. This can be done in two ways:

```
Dim Counter As Integer
Dim V As Double
For Counter = 0 To 10
V = Counter / 10.0
MsgBox(V)
Next
```

This method requires extra variable, and does not take advantage of the for loop. A better way is to use the STEP keyword with double or single data type to make it easy for us:

```
Dim Counter As Double
For Counter = 0 To 1 Step 0.1
    MsgBox(Counter)
Next
```

One last important thing to notice is that the initial value of the variable should always be smaller than or equal to the value after the To keyword, otherwise the for loop does not get executed and it is skipped. For example:

```
For Counter = 10 To 1
MsgBox(Counter)
Next
```

Will never give you message box at all. To fix this and make the count down work, just put a negative step value:

```
For Counter = 10 To 1 Step -1
   MsgBox(Counter)
Next
```

These are most of the details needed to work with the For loop. The next example is a simple one showing how to use the FOR loop to identify Prime number.

Prime numbers are numbers that can only be divided by themselves and 1 with remainder=0. So this means if we have number 9212, we should check the remainder of dividing this number over all the values from 9212 to 2 and it should never give a zero if it is a prime. Without for loop this is very hard to compute. The code to calculate the prime number is:

```
If MyNumber Mod I = 0 Then
    ' mark that the remainder is not zero
    RemainderIsZeroFlag = True
End If
Next
' if there was any remainder then tell the user
' that the number is not prime, else it is.
If RemainderIsZeroFlag Then
    MsgBox("The number is not prime")
Else
    MsgBox("The number is prime")
End If
```

Next chapter we will start working with arrays and collections, and things will get more exciting.

**Chapter 9: Arrays** 

## Arrays

In many cases you need to perform processing on huge amount of data. For example you may want to find an average of 7000 values in a file, or reading unknown number of people names into your application and recalling them back. In such case it is inconvenient to define all these variables separately. Instead you define an array which holds all the records.

Think of an array as a big variable that you can get and store values in it by specifying position. For example, below is an array of integer, let us call it **A**:

Element 0 in A	A 33
Element 1 in A	A 123
Element 2 in A	A 3
Element 3 in A	A -5
Element 4 in A	A 19
Element 5 in A	A 77
Element 6 in A	A 1212
Element 7 in A	A 0

The cells in green are just shown here to show you the position of the variable, while the cells in purple show you the content of each variable. As you can see here this array can store 8 variables (starting from 0 and ending at 7). The first location is always 0, some people might find it confusing. You can ignore the first location and avoid any confusion, but you need to know where does the indexing of the array start.

In VB.Net to create an array like the above, you write the following:

### Dim A(7) As Integer

This statement generates an array that can store integers. The array has 8 elements (starting with index 0, and ending with 7). When the statement above is executed, you get the following result in memory:

Element	0	in	А	0
Element	1	in	Α	0
Element	2	in	А	0
Element	3	in	Α	0
Element	4	in	Α	0
Element	5	in	А	0
Element	6	in	А	0
Element	7	in	Α	0

To fill the array, you use something like below:

' fill element in locations 0, 1, 2, and 7 (no need to fill in order)  $A\left(1\right) \ = \ 77$ 

A(2) = 14 A(0) = -1 A(7) = 19' fill element in location 3 (can use expression to find the location) A(2 + 1) = 321' fill element in location 5 (can use variable to find the location) Dim J As Integer J = 5 A(J) = -1000

Element 0 in A -1 Element 1 in A 77 Element 2 in A 14 A(1) = 77Element 3 in A 321 A(2) = 14Element 4 in A 0 A(0) = -1-1000 Element 5 in A A(7) = 19Element 6 in A 0 A(2+1) = 321Element 7 in A 19 A(J) = -1000 (J is 5)

In order to understand how this works, check out the graph below:

Now, let us fill the array with numbers from 0 to 7:

```
' define array first
Dim MyNumbers(7) As Double
' define a counter
Dim I As Integer
' fill the array
For I = 0 To 7
MyNumbers(I) = I
Next
```

If we want to fill the values from the keyboard (let us say, we want the user to enter 10 names):

' define the array
Dim My10Names(9) As String

```
' define a counter
Dim C As Integer
' fill the array
For C = 0 To 9
My10Names(C) = InputBox("Enter the number number " & C)
Next
```

To get the values out of the array, you use the same format used above. For example the following code shows the content of an array in a list box:

```
' display the result
ListBox1.Items.Clear()
For I = 0 To 7
ListBox1.Items.Add(My10Names(I))
Next
```

To test what we have learned, you can find a simple application on the web site that shows you how to use arrays, by filling them, finding max and min values, and finding average. You need to add a DataGridView, and 4 CommandButtons to your form. The code is:

```
Public Class Form1
' we define two arrays one to store names, another to store marks
Dim Names(9) As String
Dim Marks(9) As Integer
' we define a variable to store how many element of the array we
' used
Dim StCount As Integer = 0
```

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

```
' read the name and mark and put them in the next empty slot
Names(StCount) = InputBox("Enter the name of student")
Marks(StCount) = InputBox("Enter the mark")
' the new name and mark should be displayed on the data grid
```

```
DataGridView1.Rows.Add(Names(StCount), Marks(StCount))
        ' move the counter to the next empty slot
       StCount = StCount + 1
   End Sub
   Private Sub Button2 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        ' find the maximum mark
       Dim I As Integer
                                   ' used for counting
       Dim MaxPos As Integer ' used to remember the index of
                                   ' maximum mark
                                   ' assume first mark is the maximum
       MaxPos = 0
       For I = 1 To StCount - 1 ' loop over all other slots
' is there an element with a mark greater than the current maximum?
           If Marks(I) > Marks(MaxPos) Then
                                                ' we found a new max,
               MaxPos = I
                                                ' update our maximum
           End If
       Next
       MsgBox("student " & Names(MaxPos) & " has the maximum mark")
   End Sub
   Private Sub Button3 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        ' find minimum
        ' it is identical to the previous, except for the condition
       Dim I As Integer
       Dim MinPos As Integer
       MinPos = 0
       For I = 1 To StCount - 1
           If Marks(I) < Marks(MinPos) Then</pre>
               MinPos = I
           End If
       Next
       MsgBox("student " & Names(MinPos) & " has the minimum mark")
   End Sub
```

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
Dim I As Integer ' I is counter
Dim AVG As Double ' Used to store the sum and finding
' the average
AVG = 0 ' The avg is zero
For I = 0 To StCount - 1 ' Loop over all elements in the
' array
AVG += Marks(I) ' Add each element to the some of
```

```
' the previous ones
Next
AVG = AVG / StCount ' divide the total by number of
' elements to get the average
MsgBox("the average is:" & AVG)
End Sub
```

End Class

Next chapter we will start working with collections, and see how they are simpler than arrays.

**Chapter 10: Collections** 

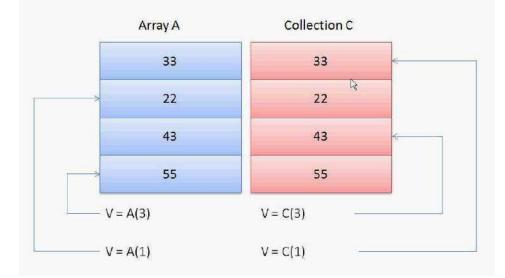
# Collections

Last time we spoke about arrays, and saw how to work with them. Today we check out something similar and easier to use, and that is collections.

Collections are very similar to arrays. They are used to store a number of values (or variables), so that you can process them all.

Array A	Collection C
33	33
22	22
43	43
55	55

There are a number of differences between arrays and collections. First difference is that the indexing for arrays starts with zero, while for a collection it starts with 1. To understand this, consider the image below:



Now the statement V = A(3) will place the value 55 in V because the indexing start at zero in arrays. However the similar statement V = C(3) will place 43 instead because collection indexing is different. The same applies for the second statement.

Another important difference is the data type. All Array elements has the same data type. So if you have an array:

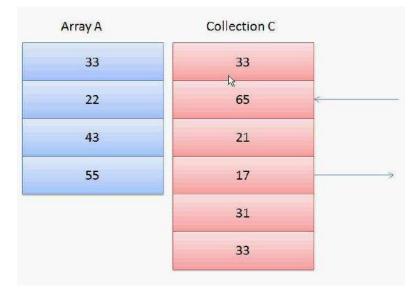
### Dim A(O To 9) As Integer

then A(0), A(1), A(2)... A(9) are all Integers. Collections on the other hand do not require this. You can store integers, reals, strings, bytes,... etc. in the same collection. This can be illustrated below:

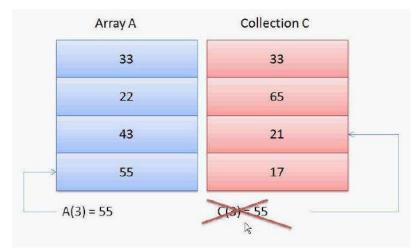
Array A	Collection C
33	33
22	22.5
43	"Hello"
55	2010-01-01

Usually you will use the same data type for all elements of the collection; however you still have the option to use different data types whenever you need to.

Another difference between arrays and collections is that collections can add elements and remove elements directly and change in size without any need for some kind of processing, while arrays are fixed in size, and you cannot insert values at specific locations at random.



Finally array elements can be updated and overwritten, while collections are not. To make this clear check out this:



In this example if you write C(3) = 55 you get an error, that is because collection does not allow you to update or overwrite the content of an element. However there is a way to overcome this. We will discuss this later.

Now let us see how to define a collection. There are two ways to define a collection:

```
Dim C As New Collection
```

In this example you create a collection object that is ready to be used. So you can add, remove elements, get the number of items, or do whatever you want with the collection directly. Another way is:

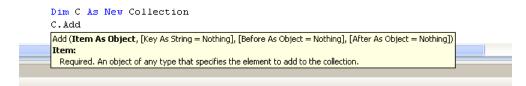
```
Dim C As Collection
```

here C is not ready yet to be used. It does not point to a collection yet. You can never use it. To be able to use it later on in the code you should write:

C = New Collection

This will allow you to use the collection without any problem. The first method of defining a collection is the one you will probably use the most.

Now in order to add elements to the collection you use the add method.



When you write C.Add the compiler shows you the parameters that you should provide. The ones in the square brackets are optional. The parameters are:

- Item: is the value you want to store
- Key: you can provide a text value to quickly access the elements of collection instead of providing numbers to access them. For example the code below will store the value 40 in V:

```
Dim C As New Collection
C.Add(33, "Smith")
C.Add(40, "Michel")
C.Add(77, "John")
Dim V As String
V = C("Michel")
```

- Before: is the index of the item you want the new element to be inserted before.
- After: is the index of the item you want the new element to be inserted after.

Next is removing elements. This simple, you just provide the index of the element you want to remove:

C.Remove(3)

This removes the 3<sup>rd</sup> element from the collection.

Also getting the number of elements is as easy. You use the count function.

I = C.Count

Finally we describe how to simulate the functionality replacing or updating an item in the array using collection.

If we write :

```
C.Add(N, , , 7)
C.Remove(7)
```

This will have exactly the same effect as:

A(6) = N

So this is a basic introduction about collections. Next is a simple application demonstrating the use of collections. Create a windows form, and make it similar to what you see below:

ListBox1	
Clear Collection	
Add Name	
Get Number of Items	
View Items in Listbox	
Remove Element	
Replace a name	

Next write the code below:

```
Public Class Form1
```

Dim MyCollection As New Collection

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

```
' this method clears all the elements in the collection
MyCollection.Clear()
End Sub
```

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

```
' read a name
        Dim Name As String
        Name = InputBox("enter a name")
        ' add the name into the list
       MyCollection.Add(Name)
   End Sub
    Private Sub Button3 Click (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
       MsgBox("the number of items is:" & MyCollection.Count)
   End Sub
    Private Sub Button4 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        ' clear old content
        ListBox1.Items.Clear()
        ' insert the items into the list box
        Dim I As Integer
        For I = 1 To MyCollection.Count
           ListBox1.Items.Add(MyCollection(I))
        Next
   End Sub
    Private Sub Button5 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        ' get element position
        Dim I As Integer
        I = InputBox("enter the element number you want to remove:")
       MyCollection.Remove(I)
    End Sub
    Private Sub Button6 Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        ' get element position
        Dim I As Integer
        Dim N As String
        I = InputBox("enter the element number:")
        N = InputBox("enter the new name:")
        MyCollection.Add(N, , , I)
        MyCollection.Remove(I)
```

End Sub

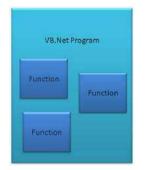
End Class

The code is very simple and at your level you should understand it very easily. If you have any problem with it just send me notes about it. Next chapter we will start working with functions, and see how it makes coding much easier for us.

**Chapter 11: Functions** 

## Functions

Up to this point, when you want to write an application you would simply write all your statements into one single block and try to solve all the problems in that simple block. However there is a better approach to write applications by dividing the application into a number of blocks that form together one application. These what functions are. Think about functions as small or mini-programs that each is designed to address a simple problem.



Functions make your applications easier to write and easier to understand. Now let us consider this example: you need to find the factorial for three variables, A,B and C. (The factorial of 4 is 4x3x2x1, and factorial of 7 is 7x6x5x4x3x2x1).

Usually to solve this issue you write something like:

```
A = 10
B = 5
C = 4
FA = 1
For I = 1 To A
FA = FA * I
Next
FB = 1
For I = 1 To B
FB = FB * I
Next
FC = 1
For I = 1 To C
FC = FC * I
```

#### Next

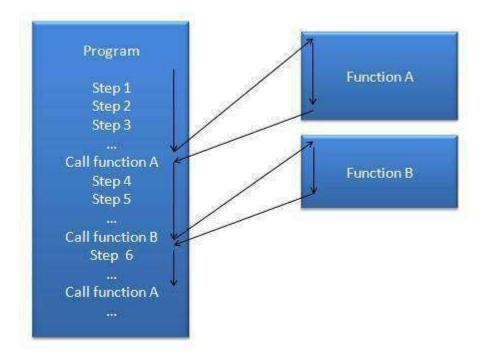
So if you have for example 30 variables, you might need to rewrite the code 30 times (assuming you are not using arrays). However if you think about this in a different way, that is: **Could I have a statement that get me the factorial?** Then the code would be something like:

```
A = 10
B = 5
C = 4
FA = Factorial(A)
FB = Factorial(B)
FC = Factorial(C)
```

Which is very easy to write and understand. All you need is to tell the computer what Factorial really is:

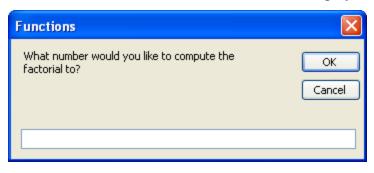
```
Function Factorial(ByVal N As Integer) As Double
Dim F As Double
' the factorial total
Dim I As Integer ' the counter
F = 1 ' the initial value of F
For I = 1 To N ' this loop to calculate the factorial
F = F * I
Next
Return F ' return the result
End Function
```

Now no matter how many times you need to calculate the factorial, you can just call it whenever you need. You don't have to worry about the loop or initializing F or anything else. You do it only once and you can use it anywhere.



Consider the above figure. It shows how the program behaves when it encounters a function call. The program executes normally, until it reaches a function call. At that time it will transfer its execution to function A. It will execute the function, get the result, and return back to the main program. The execution continues from the function call, and again, when there is another function call the process in repeated.

The functions you use could be built in functions, which means they are already written and available in the compiler, or user defined functions which are the ones you write. Built in functions are used to do general operations needed in most programs, and to make life easier for programmers. Examples of these are the **InputBox** function which shows a small window to read values from the display.



Other examples are math functions like **Math.Pow** - which finds the power of a number - and **Math.Abs** - which finds the absolute value of a number - and string functions like UCase and LCase.

As for user defined functions, you can define a function as follows:

```
' the factorial function
Function Factorial (ByVal N As Integer) As Double
Dim F As Double
' the factorial total
Dim I As Integer
F = 1 ' the counter
F = 1 ' the initial value of F
For I = 1 To N ' this loop is used to calculate the
' factorial
F = F * I
Next
Return F ' return the result
End Function
```

Function, End Function parts define where the code of the function starts and when it ends.

Factorial is the name of the function. You can choose any name however you might want to choose a meaningful name to remind you what this function does.

ByVal N As Integer is the parameter of the function. For now ignore the ByVal part. The parameter is a value passed from your program to the function so that the function could work on it. You define all the parameters between the brackets.

As Double which comes after the brackets tells the compiler what data type this function returns. When a function finish its processing it should return the result in the specified data type.

Return is used to tell the compiler what is the result of executing this function.

The following is an example of a function:

```
' the sum function
Function Sum(ByVal V1 As Integer, ByVal V2 As Integer, ByVal V3 As
Integer) As Integer
Return V1 + V2 + V3 ' calculate the sum and return the result in
```

```
' one single step
```

End Function

Now let us check the application included with the tutorial and see how it works:

Max Value in Array
AVG of an array

It allows you to compute the factorial of a number, find the sum of 3 numbers, read and find max value or average of an array.

The factorial function is:

```
' the factorial function
Function Factorial (ByVal N As Integer) As Double
Dim F As Double
' the factorial total
Dim I As Integer ' the counter
F = 1 ' the initial value of F
For I = 1 To N ' this loop is used to calculate the
factorial
F = F * I
Next
Return F ' return the result
End Function
```

and the code of the factorial button is:

```
' read the number from screen
Dim MyNumb As Integer
Dim R As Double
MyNumb = InputBox("What number would you like to compute the
factorial to?")
R = Factorial(MyNumb)
MsgBox("the result is:" & R)
```

Check out how to call the function, you use its name and pass the parameter value. You can even call the function in the following ways:

R = Factorial(MyNumb + 2) R = Factorial(6) R = Factorial(77/2) R = Factorial(A - B)

and the value of the expression between the brackets is evaluated and passed to the function and placed in the variable N.

The rest of the code is straight forward. However I would like to highlight the functions that process functions:

```
' the max value in an array
Function GetMax(ByVal A() As Integer) As Integer
   Dim I As Integer
   Dim Max As Integer
                                    ' assume the max value is the
   Max = A(0)
                                    ' first value
   For I = 1 To A.Length - 1
                                  ' loop over all other values in
                                   ' an array
       If Max < A(I) Then
                                   ' if we find another value
                                  ' larger than max then
                                   ' update max
           Max = A(I)
       End If
   Next
   Return Max
                                  ' return the result
End Function
' the avg value of an array
Function GetAVG(ByVal A() As Integer) As Integer
   Dim I As Integer
   Dim sum As Integer
   sum = 0
                                    ' assume the sum value is 0
                                  ' loop over all other values in
   For I = 0 To A.Length - 1
                                   ' an array
       sum = sum + A(I)
```

```
Next
   Return sum / A.Length ' return the result
End Function
' read array
Function ReadArray() As Integer()
   ' the counter
   Dim I As Integer
    ' the number of elements
   Dim N As Integer
    ' read the number of elements from the display
   N = InputBox("how many elements in the array")
    ' create the array
   Dim A(O To N - 1) As Integer
    ' read the elements of the array
   For I = 0 To N - 1
       A(I) = InputBox("enter the element:" & I.ToString)
   Next
    ' return the result
   Return A
End Function
```

When you pass array to a function you use brackets () to tell the function that it is going to receive the array. In ReadArray() When the function returns an array, you add brackets after the return data type.

Next chapter we discuss functions more.

Chapter 12: ByVal & ByRef

## ByVal & ByRef

One of the useful properties of functions is that it allows you to propagate changes in a parameter back to its original variable. For example consider this simple functions:

```
Function TestByVal(ByVal N As Integer) As Integer
N = 0
Return N
End Function
Function TestByRef(ByRef N As Integer) As Integer
N = 0
Return N
End Function
```

Now if you call the functions like this:

```
Dim K1 As Integer = 100
TestByVal(K1)
MsgBox(K1)
Dim K2 As Integer = 100
TestByRef(K2)
MsgBox(K2)
```

The first MSGBOX will display the value 100, while the second will show the value zero. This can be explained as follows:

When the computer sees the ByVal keyword it will create an independent copy of the variable you are passing to the function and work on it. In our example TestByVal, the function sets the variable N to zero. Since it is passed by value this won't affect the orginal parameter K1.

However when you use ByRef keyword the computer will create a copy of the variable that is linked to the original parameter. In other words the variable N in the example is another name for the variable K2, so when you change N in the second

function, K2 changes as well. So this is why you get the value zero from the second MSGBOX.

Now let us create a useful example to swap two numbers:

```
' swap function
Function Swap(ByRef V1 As Integer, ByRef V2 As Integer)
    Dim Tmp As Integer
    Tmp = V1
    V1 = V2
    V2 = Tmp
End Function
```

Here you are passing two numbers into the function and the function swaps them. Now to call the function you can try something like this:

```
Dim N1 As Integer
Dim N2 As Integer
N1 = InputBox("Enter N1")
N2 = InputBox("Enter N2")
Swap(N1, N2)
MsgBox("N1:" & N1)
MsgBox("N2:" & N2)
```

So whenever you call the function the numbers are changed. Try to modify the code of the function to have some errors, for example let it be like this:

```
' swap function modified
Function Swap(ByVal V1 As Integer, ByVal V2 As Integer)
    Dim Tmp As Integer
    Tmp = V1
    V1 = V2
    V2 = Tmp
End Function
```

Try the same code and you will see it will not work as we clarified before. Also try modifying the function to be like this:

```
' swap function modified
Function Swap(ByVal V1 As Integer, ByRef V2 As Integer)
    Dim Tmp As Integer
    Tmp = V1
    V1 = V2
    V2 = Tmp
End Function
```

Here you are going to get strange result, one of the values will be changed, while the other not simply because one of the parameters if by reference and the other is by value. So just in case you are getting unexpected results and your function code appear to be correct, just make sure that the ByVal and ByRef keywords are used correctly.

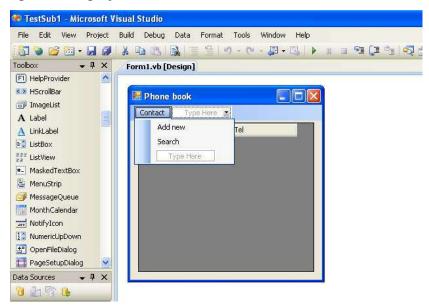
Next chapter deals with subroutine, and try to create a useful sort program.

**Chapter 13: Subroutines** 

## Subroutines

In vb.net you can write other type of coding blocks called subroutines. Subroutines are almost exactly similar to function except they don't return a value. In this chapter we are going to create a simple application that stores person's name and telephone number and allows you to search the names. If you did not read the chapter about working with functions, then you won't be able to understand this one.

Open a new project, and then create the interface similar to the below:



You just add a DataGridView control and a menustrip. Set fields for the DataGridView control.

Next go to the code window and write the following:

```
' define the main variables
Dim Names As New Collection
Dim Tels As New Collection
```

These two collections are going to store the names and telephone numbers, next write:

```
' add new person
Sub AddContact(ByVal CName As String, ByVal CTEL As String)
Names.Add(CName)
Tels.Add(CTEL)
End Sub
```

This one is a subroutine that will add the name and telephone numbers to the collections. As you can see there is no return value, and the definition is very similar to the functions. The next subroutine is used to view the content of collections in the DataGridView control.

```
' display the names in the grid
Sub ViewContacts(ByVal DGV As DataGridView)
DGV.Rows.Clear()
Dim I As Integer
For I = 1 To Names.Count
DGV.Rows.Add(Names(I), Tels(I))
Next
End Sub
```

Now, we need to add the code for the "Add New" person menu item, go to the user interface, and double click the Add New menu item, and write the code:

```
' the handler for the add new contact command
    Private Sub AddNewToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddNewToolStripMenuItem.Click
        Dim N As String
        Dim T As String
        N = InputBox("Enter the name of the contact:")
        If N = "" Then
            Exit Sub
        End If
        T = InputBox("Enter the tel number:")
        If T = "" Then
            Exit Sub
        End If
        AddContact(N, T)
        ViewContacts(DataGridView1)
    End Sub
```

If you check the definition of the block:

```
Private Sub AddNewToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddNewToolStripMenuItem.Click
```

you realize that the event handler is actually a subroutine. So we have been using subroutines all the time. Next create a search function and the event handler for the search menu item:

```
' the search function
   Function GetTelForName (ByVal Name As String) As String
       Dim I As Integer
       For I = 1 To Names.Count
            If Names(I) = Name Then
                Return Tels(I)
           End If
       Next
       Return ""
   End Function
   Private Sub SearchToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SearchToolStripMenuItem.Click
       Dim N As String
       Dim T As String
       N = InputBox("Enter the name you are searching for:")
        If N = "" Then
            Exit Sub
       End If
       T = GetTelForName(N)
        If T = "" Then
           MsgBox("Name not found")
       Else
           MsgBox("the tel number is:" & T)
       End If
   End Sub
```

Finally run the application, try to add some names, and perform a search. So this is a quick example about using subroutines.

Chapter 14: Do Loop

# Do Loop

In VB.NET, you can use For loop which we saw how to use in one of the previous tutorials. Another type of loop is the Do loop. The format of this loop can be one of the following:

```
Do
      Statement...
      Statement...
      ...
Loop While Condition
Or
Do
      Statement...
      Statement...
      ••••
Loop Until Condition
Or
Do While Condition
      Statement...
      Statement...
      ...
Loop
Or
Do Until Condition
      Statement...
      Statement...
      ...
Loop
```

The location of the condition tells when the checking of exiting or staying in the loop should be performed. For example:

```
Do While K<10
Statement...
Statement...
Loop
```

Here the check is performed every time before the loop is executed. It is possible in this example for the loop to never get executed. However if you write it like this:

```
Do
Statement...
Statement...
...
Loop While K<10
```

The check is performed after the loop is executed. So here the loop is executed at least once.

Now let us test the Do Loop with an example:

Count from 1 to 10 FOR LOOP	
Count from 1 to 10 using DO LOOP	
Find SQRT	

Create a form similar to the above picture, and then for the first button write the code:

```
Dim I As Integer
ListBox1.Items.Clear()
For I = 1 To 10
ListBox1.Items.Add(I)
Next
```

This is the standard For loop, Now we will create a similar loop using the Do loop. Add the following code for the second button.

```
Dim I As Integer
I = 1
ListBox1.Items.Clear()
Do
    ListBox1.Items.Add(I)
    I = I + 1
Loop While (I < 11)</pre>
```

Run the application and see how both loops give the same result. The thing about the Do loop here is that you much initialize the counter (I) yourself, and increase its value, and check when you should stop the loop. This is something you don't have to do with the For loop.

The Do loop is not used to replace the For loop, but it is used when you don't know how many times you need to execute the code. For example, assume you want to find the square root for an unknown number of values, and you want to stop when you enter a negative number. The way you do it is as follows:

Here you can stop after entering one value, or after entering ... say 500 +ve numbers. Try to add the code to the third button in the form above and check it out. The full code for the form should be similar to the following:

```
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)Handles Button1.Click
    Dim I As Integer
```

```
ListBox1.Items.Clear()
      For I = 1 To 10
           ListBox1.Items.Add(I)
      Next
      End Sub
      Private Sub Button2 Click (ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
      Dim I As Integer
     I = 1
     ListBox1.Items.Clear()
      Do
            ListBox1.Items.Add(I)
            I = I + 1
      Loop While (I < 11)
      End Sub
      Private Sub Button3 Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
Handles Button3.Click
            Dim I As Double
            Do
                  I = InputBox("Enter a +ve number:")
                  If I >= 0 Then MsgBox("the root of your number is:" &
            Math.Sqrt(I))
            Loop While I > 0
      End Sub
End Class
```

Now let us try another example with the Do Loop, this time we use it to perform bubble sort. Create a form similar to the one below:

Read Values	
Sort	

Next write down the code of the form to be as follows:

```
Public Class Form1
      Dim A() As Integer
      Private Sub Button1 Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
Handles Button1.Click
      Dim Count As Integer
      Count = InputBox("enter the number of values:")
      ReDim A(0 To Count - 1)
     Dim I As Integer
      For I = 0 To Count - 1
            A(I) = InputBox("Enter the value " & I.ToString)
     Next
      ViewArray(A, ListBox1)
      End Sub
      Public Sub ViewArray (ByVal Ar() As Integer, ByVal L As ListBox)
           L.Items.Clear()
            Dim I As Integer
            For I = 0 To Ar.Length - 1
                 L.Items.Add(Ar(I))
            Next
      End Sub
      Private Sub Button2 Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
Handles Button2.Click
            ' sort
            Dim I As Integer
            Dim J As Integer
            Dim Flag As Boolean
            Do
                  Flag = False
                  For I = 0 To A.Length - 2
                        If A(I) > A(I + 1) Then
                              Flag = True
                              J = A(I)
                              A(I) = A(I + 1)
                              A(I + 1) = J
                        End If
                  Next
            Loop Until Flag = False
            ViewArray(A, ListBox1)
```

End Sub End Class

Check out the code for the sort operation:

```
' sort

Dim I As Integer

Dim J As Integer

Dim Flag As Boolean

Do

Flag = False

For I = 0 To A.Length - 2

If A(I) > A(I + 1) Then

Flag = True

J = A(I)

A(I) = A(I + 1)

A(I + 1) = J

End If

Next

Loop Until Flag = False
```

The Flag is used to tell when any two values in the array are exchanged, which tells the computer to perform another loop on array elements. Notice that this bubble sort is not very efficient, and it can be improved. However it is used here to just show you an example of when to use the Do Loop.

**Chapter 15: Structures** 

#### Structures

In VB.NET, you can combine a number of related variables together and treat them as one unit. This simplifies programming and makes updating the source of your applications easier.

Let us first consider the following example; you want to store the information about a person/employee. The information include name, telephone number and salary. So far we learned that to store such information you should define 3 distinct variables:

```
Dim Name As String
Dim TEL As String
Dim Sal As Decimal
```

Later on you fill these variables with values, and uses them. Now what is you have two employees? Obviously you define another 3 variables:

```
Dim Name2 As String
Dim TEL2 As String
Dim Sal2 As Decimal
```

Now what if you have a 1000 employee? Well a better solution is to use arrays. However now you need to define 3 arrays:

```
Dim Names() As String
Dim TELs() As String
Dim Sals() As Decimal
```

The first array stores the names, the second stores telephone numbers, and last one stores salary. So arrays handles the information for large amount of data pretty well. But what is you need to add another property such as address? The solution is to add another array:

```
Dim Address() As String
```

And if you need to store another property you need to store another array. If you have

14 property for an employee, then you have to store and manage 14 different arrays. In such situations structures are useful. You define a structure to combine the different properties like this:

```
Structure PersonInfo
Dim Name As String
Dim Tel As String
Dim Sal As Decimal
Dim Address As String
End Structure
```

Now you have a new data type called PersonInfo which contains inside it a name, a telephone, a salary and an address for that particular employee/person.

Dim A As PersonInfo
A.Name = "Smith"
A.Tel = "555-22-332"
A.Sal = 400

So A here is the name of the variable and it stores all the attributes or properties of employee/person. To access a specific property you use the dot (.) followed by the property. For example A.NAME access the name property of that employee. Now to define another employee/person you write:

```
Dim B As PersonInfo
B.Name = "Micehl"
B.Tel = "111-22-332"
B.Sal = 700
```

So now you have two employees A & B. Now if you want to define an array of such structure you can do so by:

```
Dim Info() As PersonInfo
```

And you can access the elements of the array normally

Dim I As Integer

```
Dim N As Integer
N = InputBox("Enter the number of people")
ReDim Info(0 To N - 1)
' read info here
For I = 0 To N - 1
Info(I).Name = InputBox("enter the name of person")
Info(I).Tel = InputBox("enter the telephone number")
Info(I).Sal = InputBox("enter the salary")
Next
```

So as you can see structures are used the same way as normal variables do. Now if you want to copy the information of a structure then :

A = B

This will allow you to copy all the content and attributes of structure B into A. So it does not matter how many attributes you have, they will all be copied in one single step. Now let us work on an example. Create a simple form containing two buttons and a data grid view. One of the buttons should read the data and the other should sort the data. Create the following columns in the data grid :

 $\cdot$  Name

 $\cdot$  TEL

 $\cdot$  Sal

· License

Then go to the code page of the form and define the following structure:

```
' this is the structure to store person information
Structure PersonInfo
    Dim Name As String
    Dim Tel As String
    Dim Sal As Decimal
    Dim LincenseNumber As String
End Structure
```

Then define an array of structure:

' this is the array to store persons' info

Dim Info() As PersonInfo

Next create a subroutine to read the information of the array

```
' read the information and store it in an array
Public Sub ReadInfo()
    Dim I As Integer
    Dim N As Integer
    N = InputBox("Enter the number of people")
    ReDim Info(0 To N - 1)
    ' read info here
    For I = 0 To N - 1
        Info(I).Name = InputBox("enter the name of person")
        Info(I).Tel = InputBox("enter the telephone number")
        Info(I).Sal = InputBox("enter the salary")
        Info(I).LincenseNumber = InputBox("enter license number")
    Next
End Sub
```

After that add the following code to display the content of the array

```
' fill the data grid with array info
Public Sub FillDGV(ByVal DAT() As PersonInfo, ByVal DGV As DataGridView)
DGV.Rows.Clear()
Dim I As Integer
For I = 0 To DAT.Length - 1
DGV.Rows.Add(DAT(I).Name, DAT(I).Tel, DAT(I).Sal,
DAT(I).LincenseNumber)
Next
```

End Sub

Next is to add the code of the read button

ReadInfo()
FillDGV(Info, DataGridView1)

Next is the sort method:

```
' the sort subroutine
Public Sub Sort(ByRef Arr() As PersonInfo)
```

```
Dim Flg As Boolean
Dim I As Integer
Dim Tmp As PersonInfo
Do
Flg = False
For I = 0 To Arr.Length - 2
If Arr(I).Name > Arr(I + 1).Name Then
Tmp = Arr(I)
Arr(I) = Arr(I + 1)
Arr(I + 1) = Tmp
Flg = True
End If
Next
Loop While Flg
End Sub
```

Look how simple the swap operation is. No need to swap the 4 attributes, you just use one swap operation. For the second button add the following code:

Sort(Info)
FillDGV(Info, DataGridView1)

So the code should be:

```
Public Class Form1
      ' this is the structure to store person information
      Structure PersonInfo
            Dim Name As String
            Dim Tel As String
            Dim Sal As Decimal
            Dim LincenseNumber As String
     End Structure
      ' this is the array to store persons' info
     Dim Info() As PersonInfo
      ' read the information and store it in an array
      Public Sub ReadInfo()
           Dim I As Integer
            Dim N As Integer
            N = InputBox("Enter the number of people")
            ReDim Info(0 To N - 1)
```

```
' read info here
      For I = 0 To N - 1
            Info(I).Name = InputBox("enter the name of person")
            Info(I).Tel = InputBox("enter the telephone number")
            Info(I).Sal = InputBox("enter the salary")
            Info(I).LincenseNumber = InputBox("enter license
            number")
      Next
End Sub
' fill the data grid with array info
      Public Sub FillDGV(ByVal DAT() As PersonInfo, ByVal DGV As
      DataGridView)
      DGV.Rows.Clear()
      Dim I As Integer
      For I = 0 To DAT.Length - 1
            DGV.Rows.Add(DAT(I).Name, DAT(I).Tel, DAT(I).Sal,
            DAT(I).LincenseNumber)
      Next
End Sub
' the sort subroutine
Public Sub Sort(ByRef Arr() As PersonInfo)
      Dim Flg As Boolean
      Dim I As Integer
      Dim Tmp As PersonInfo
      Do
            Flg = False
            For I = 0 To Arr.Length - 2
                  If Arr(I).Name > Arr(I + 1).Name Then
                        Tmp = Arr(I)
                        Arr(I) = Arr(I + 1)
                        Arr(I + 1) = Tmp
                        Flg = True
                  End If
            Next
      Loop While Flg
End Sub
Private Sub Button1 Click (ByVal sender As System.Object, ByVal e
As System.EventArgs)
Handles Button1.Click
      ReadInfo()
      FillDGV(Info, DataGridView1)
End Sub
```

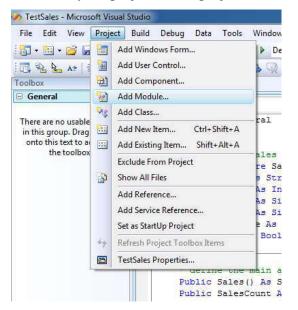
```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs)
Handles Button2.Click
Sort(Info)
FillDGV(Info, DataGridView1)
End Sub
End Class
```

The source file contains a simple added code, you should try to use that to add more functionality to the example. Try to include car information.

**Chapter 16: Modules** 

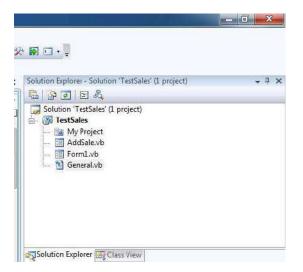
# Modules

In vb.net you can write place your code in different files. Some of these files are called modules. A module is a file that contains vb code only (i.e. functions, structures, subroutines...). It does not include GUI like buttons, lists, menus ...etc. Now to add a module to your project select project then add module



After that you provide module name:

ategories:		<u>T</u> emplates:		
Common Items Code Data		Visual Studio installed templates About Box Class Diagram	열 Class 안 Code File	
General Web Windows Fo Reporting Workflow WPF	orms	COM Class Custom Control Dialog LINQ to SQL Classes Local Database Cache MDI Parent Form Report Resources File Splash Screen	<ul> <li>Component Class</li> <li>DataSet</li> <li>Explorer Form</li> <li>JScript File</li> <li>Local Database</li> <li>Login Form</li> <li>Module</li> <li>Report Wizard</li> <li>Service-based Database</li> <li>Text File</li> </ul>	
		User Control	User Control (WPF)	
A file for storing gr	oups of functio	ns		
ame:	General			



You can see the module file is added into your project. The code you see is:

```
Module General
End Module
```

Now you can write the functions and subroutines in this module. For example:

```
Module General

' define the sales structure

Public Structure SalesStruct

Dim Item As String

Dim Count As Integer

Dim Price As Single

Dim Total As Single

Dim TheDate As String

Dim Flg As Boolean

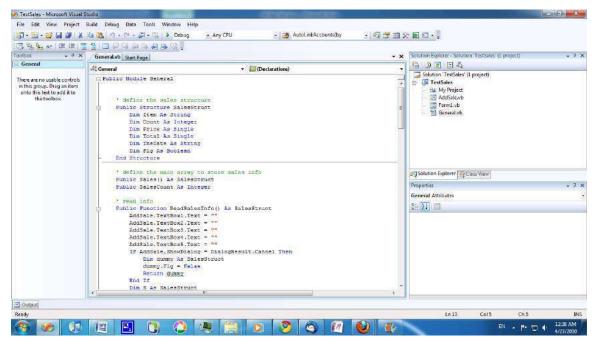
End Structure

End Module
```

So you might be wondering what difference does modules make in a program? Well modules helps you place related functions, subroutines, and other coding that you make in one place, so that it becomes easier for you to find, and easier for you to work with, and make it possible for other parts of your program to use the code.

To illustrate this consider that you have a two or three forms that require some sort operation. Instead of writing the code in one form which makes it part of that form, you place the code in a module, and sort function/subroutine becomes available to all the forms. Later on you can even take the code of the sort operation and add that to another project and you will find it works without modification (assuming the code is done correctly).

Now the following example (the sales vb project included on the web site). This project contains one module:



The idea of this project is to add a number of items you were able to sale, and later on you could find the total of sales, save or load the files.

If you check the module you will find the code:

```
' define the sales structure
Public Structure SalesStruct
  Dim Item As String
  Dim Count As Integer
  Dim Price As Single
  Dim Total As Single
  Dim TheDate As String
  Dim Flg As Boolean
End Structure
```

Which defines the main structure, then

' define the main array to store sales info

```
Public Sales() As SalesStruct
Public SalesCount As Integer
```

Which defines the array and its number of elements

```
' read info
Public Function ReadSalesInfo() As SalesStruct
    AddSale.TextBox1.Text = ""
    AddSale.TextBox2.Text = ""
    AddSale.TextBox3.Text = ""
    AddSale.TextBox4.Text = ""
    AddSale.TextBox5.Text = ""
    If AddSale.ShowDialog = DialogResult.Cancel Then
        Dim dummy As SalesStruct
        dummy.Flg = False
        Return dummy
    End If
    Dim S As SalesStruct
    S.Item = AddSale.TextBox1.Text
    S.Count = AddSale.TextBox2.Text
    S.Price = AddSale.TextBox3.Text
    S.Total = AddSale.TextBox4.Text
    S.TheDate = AddSale.TextBox5.Text
    S.Flq = True
    Return S
End Function
```

This function uses a dialog called AddSale to read the information of an item. The first part just clears the text boxes on the form/dialog, and the if statement part shows the window and tells you if the user canceled the data entry, and the last part fills the structure from the form and returns the result.

```
' display the information of the strucutre in the data grid view
Public Sub DisplayArray(ByVal Arr() As SalesStruct, ByVal DGV As
DataGridView)
        DGV.Rows.Clear()
        Dim I As Integer
        For I = 0 To Arr.Length - 1
            DGV.Rows.Add(Arr(I).Item, Arr(I).Count, Arr(I).Price,
Arr(I).Total, Arr(I).TheDate)
```

Next End Sub

This part displays the information of the array in a data grid view

```
' remove an item from array based on item name
    Public Sub RemoveItemBasedOnName (ByVal Name As String, ByRef Arr()
As SalesStruct, ByRef IC As Integer)
       Dim I As Integer
        Dim J As Integer
        For I = 0 To Arr.Length - 1
            If Name = Arr(I).Item Then
                For J = I + 1 To Arr.Length - 1
                    Arr(J - 1) = Arr(J)
                Next
                ReDim Preserve Arr(0 To Arr.Length - 2)
                IC = IC - 1
                Exit Sub
            End If
       Next
    End Sub
```

This one removes an item based on its name

```
' save the sales info
    Public Sub SaveFile(ByVal FileName As String, ByVal Arr() As
SalesStruct)
        FileSystem.FileOpen(1, FileName, OpenMode.Output,
OpenAccess.Write)
        Dim I As Integer
        FileSystem.PrintLine(1, Arr.Length)
        For I = 0 To Arr.Length - 1
            FileSystem.PrintLine(1, Arr(I).Item)
            FileSystem.PrintLine(1, Arr(I).Price)
            FileSystem.PrintLine(1, Arr(I).TheDate)
            FileSystem.PrintLine(1, Arr(I).Total)
        Next
        FileSystem.FileClose(1)
    End Sub
    ' load the file info
```

```
Public Sub LoadFile(ByVal FileName As String, ByRef Arr() As
SalesStruct, ByRef IC As Integer)
FileSystem.FileOpen(1, FileName, OpenMode.Input,
OpenAccess.Read)
Dim I As Integer
IC = FileSystem.LineInput(1)
ReDim Arr(0 To IC - 1)
For I = 0 To Arr.Length - 1
Arr(I).Item = FileSystem.LineInput(1)
Arr(I).Price = FileSystem.LineInput(1)
Arr(I).TheDate = FileSystem.LineInput(1)
Arr(I).Total = FileSystem.LineInput(1)
Next
FileSystem.FileClose(1)
End Sub
```

These two saves and load the information

```
' get total sum
Public Function GetTotalSales(ByVal Arr() As SalesStruct) As Single
    Dim S As Single = 0
    Dim I As Integer
    For I = 0 To Arr.Length - 1
        S += Arr(I).Total
    Next
    Return S
End Function
```

This last one finds the total. As you can see there is almost no difference in the code that is inside the module. It is exactly the same as the code you use in forms. Now if you open the main form of the application

🧑 TestSales - Microsoft Visual			and the second se	i x	
File Edit View Project	Build Debug Data Format Tools Window Help				
🖓 • 🗃 • 🗃 🖬 🗿 🗡	🔄 🖳 🔊 - 🕫 - 💭 - 🖏 🖡 Debug 🔹 Any CPU 🔹 - 🔯 AutoLinkAccounts(by 🔹 - 🧐 🖄	· 🖻 🔀 🖬 🖸 • 📮			
部長さ山市のよ	·唐国市市的新建业务教会(国际)合作国际				
Toolbox - # ×		- × Solution Explorer - Solution 'T	est5ales' (Eproject)	+ ∓ ×	
All Windows Forms	and a second second	6 9 2 3 3 4			
😑 Common Controls	🥡 Sales	Solution 'TestSales' (1 pr	piect)		
Pointer		TestSales			
Button	File Sales Tools	My Project			
CheckBox	tem Count Price Total Date	- 🛃 AddSale.vb			
R CheckedListBox		- 🧾 Form1.vb			
ComboBcx		🚽 🐚 General vis			
T DateTimePicker					
A Label					
A LinkLabel	6				
🔣 ListBox					
12 ListView					
<ul> <li>MaskedTedBox</li> </ul>					
MonthCalendar		Solution Explorer	View		
T Notifylcon		Properties	-wear	* 7 ×	
12 NumericUpDown		Form1 System.Windows.Form	in Easter		
PictureBox			is com		
III ProgressBar		部分 日 🐔 🗐			
RadioButton		Padding	0, 0, 0, 0	*	
Sa RichTextBox		RightTaLeft	No		
abi TextBox		RightToLeftLayout	False		
L ToolTip		Shewlcon	True		
TreeView		ShowInTaskbar	True		
🛃 WebBrowser		E Size	507, 341		
E Containers		SizeGripStyle	Auto		
🗄 Menus & Toolbars	문 OFD 🗟 ManuStrip1 문 SFD	StartPosition	WindowsDefaultLocation	100	
🙃 Data		Teg	Sales		
🗄 Components		Text	Sales	+	
- Output					
Ready					
			1000		
A 🐼 🔊	12 🗳 🚺 🜔 🗶 🗒 🗿 🚫 🙆 🙋 🤤	4		27/2010	

This one contains a menu strip and a data grid view, with open files dialog and save file

dialog. If you check the code of the form

```
Private Sub AddToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddToolStripMenuItem.Click
       Dim SR As SalesStruct
        SR = General.ReadSalesInfo
        If SR.Flg Then
            SalesCount = SalesCount + 1
            ReDim Preserve Sales(0 To SalesCount - 1)
            Sales(SalesCount - 1) = SR
            DisplayArray(Sales, DGV)
        End If
   End Sub
    Private Sub RemoveSaleToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RemoveSaleToolStripMenuItem.Click
        If DGV.SelectedRows.Count = 0 Then
            Exit Sub
        End If
        RemoveItemBasedOnName(DGV.SelectedRows.Item(0).Cells(0).Value,
Sales, SalesCount)
        DisplayArray(Sales, DGV)
```

```
End Sub
    Private Sub ExitToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
        End
   End Sub
    Private Sub SaToolStripMenuItem Click(ByVal sender As System.Object,
ByVal e As System. EventArgs) Handles SaToolStripMenuItem. Click
        SFD.Filter = "*.txt|*.txt"
        If SFD.ShowDialog = Windows.Forms.DialogResult.Cancel Then
            Exit Sub
        End If
        SaveFile(SFD.FileName, Sales)
   End Sub
    Private Sub LoadToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
LoadToolStripMenuItem.Click
        If OFD.ShowDialog = Windows.Forms.DialogResult.Cancel Then
            Exit Sub
        End If
        LoadFile(OFD.FileName, Sales, SalesCount)
        DisplayArray(Sales, DGV)
    End Sub
    Private Sub FindTotalToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
FindTotalToolStripMenuItem.Click
       MsgBox("the total sales:" & GetTotalSales(Sales).ToString)
   End Sub
```

You can see the code is much smaller here, because it is just a call to the code in the module. In fact later on if you want to modify the user interface, the code of the module

is not affected. Also since the code is much smaller, it is easier for other programmers to understand you code and update it.

So to sum things up, modules:

- 1- Are vb files
- 2- Used to store functions/subroutines, and other vb coding
- 3- Makes your program easier to maintin
- 4- Makes your program easier to understand
- 5- You can use/not use them, it is up to you
- 6- You can use any number of modules in a vb project
- 7- Make it easy to port your code to another application
- 8- Helps you isolate the interface design from program logic.

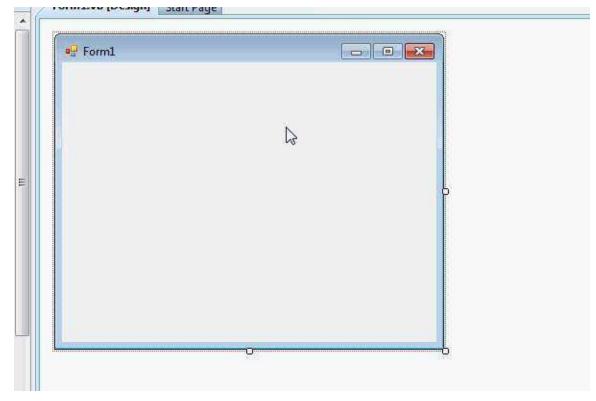
**Chapter 17:Classes** 

# Classes

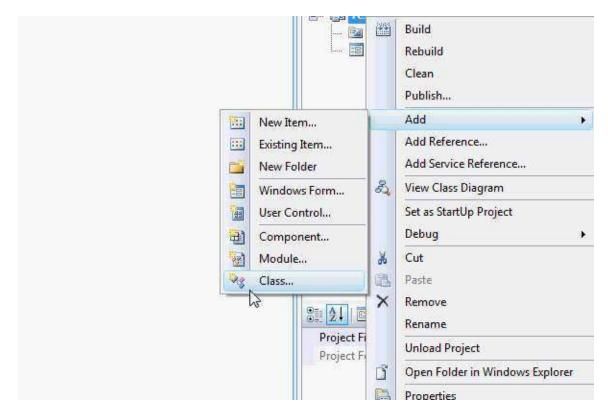
Previously we saw how to work with structures, and we examined how we can combine related information into one logical unit. Classes are very similar to structures, except that they allow you to combine the functions and subroutines that work on your information as well. It also has many other useful features that allows you to create and use frameworks to reuse the code.

To start understand classes we are going to develop a simple address book application. The application will allow you to store user information (name, address and telephone number).

Start visual studio, and create a new project.



First thing we are going to do is to create a simple class that will describe the information for each contact. Usually each class is placed in a separate file. The process is similar to adding module, or adding another form to your project. Right click your project->Add->Class



Next provide the name of the class (ContactInfo)

Report Resources File Splash Screen User Control	Report Wize Service-bas Text File User Control Text File
n ntactinfo.vb	
	Besources File

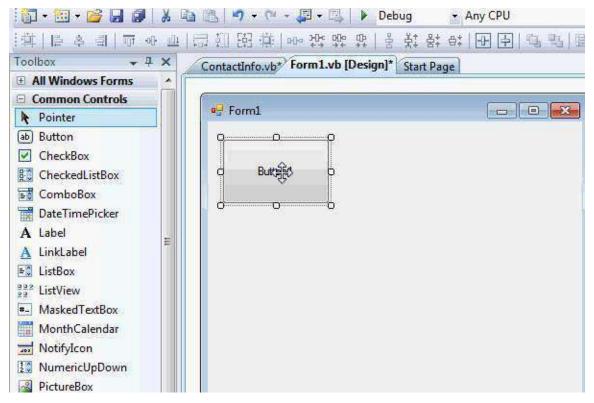
Next you will see the following:

	월 😤 🔄 - 연 - 💭 - 🖾   ▶ Debug 🔹 Any CPU ♀ ♀ ♀ ♀ ₽ ♀ ♀	z 🖄 FillTex
→ 구 ×	ContactInfo.vb Form1.vb [Design]* Start Page	(Declarations)
ible controls rag an item :o add it to box.	End Class	

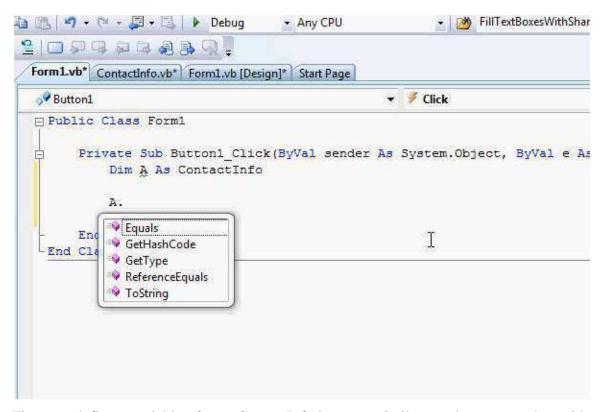
Now you should start writing the code for your class. The contact for each person should include person Name, Address and Telephone, therefore you define three variables as shown below:

	🖹 🖺 🖣 🔹 🍋 🗸 💭 🕶 🗔 📔 🕨 Debug	Any CPU	- 10	FillTextBoxesV
4 ×	ContactInfo.vb* Form1.vb [Design]* Start	Page		-
		÷	(Declarati	ons)
itrols tem : to	Public Class ContactInfo Dim Name As String Dim Address As String Dim Tel As String Dim Notes As String	I		
	-End Class			

To test this go to the form and place a button



Next try to add the following to the code of the button



First you define a variable of type ContactInfo in a way similar to what you used to with structures, however, when you want to access the variables of the class you will find that the editor does not list them. Actually even if you write them manually you won't be able to run the program. This is because the variables within the class are protected from access outside the class code. This helps hiding complex code and the variables you don't want to be accessed by mistake.

Now to make any variable accessible just change the Dim keyword in front of the variable to Public. This will grant this variable public access from any code within the project.

View Project	Build Debug Data Tools Window Help	
🚰 🖬 🕼   🐰	🛍 🕄 🔄 🔹 🖓 🗸 🖓 🗸 🖏 🕨 Debug 💽 💽 Any CPU	👻 🖄 Fill
A≥   律 律   🧮	2 I 🗆 🖓 📮 🖓 🗛 🖓 📮	
<b>→</b> ∓ ×	Form1_vb* ContactInfo.vb* Form1.vb [Design]* Start Page	
	ContactInfo	(Declarations
usable controls 5. Drag an item ext to add it to oolbox.	Public Name As String Dim Address As String Dim Tel As String Dim Notes As String	
	-End Class	

Now if you try to access the name property, you will see that the editor can detect that,

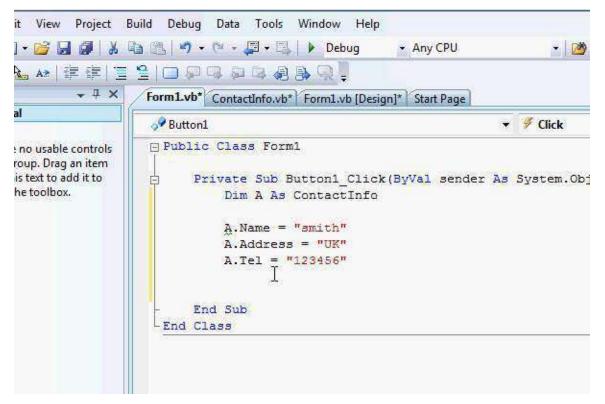
and the property is listed when you press the (.) after the variable name.

ral	Putton1	+	💰 Click
e no usable controls group. Drag an item his text to add it to the toollsox.	Public Class Form1 Private Sub Button1_Click(ByVal sender Dim À As ContactInfo A. End Class Form1 End Class Form1 Private Sub Button1_Click(ByVal sender Dim À As ContactInfo A. End Class Form1 Private Sub Button1_Click(ByVal sender Dim À As ContactInfo A. Private Sub Button1_Click(ByVal sender Private Sub Button1_Cli	As	System.

Make the Address and Tel variables within the class public similar to the way below:

View Project	Build Debug Data Tools Window Help	
🚰 园 🕼   🐰	🗈 🔁 🔄 🔹 🖓 🔹 🖓 🔹 🖾 📔 🕨 Debug 💦 💽 Any CPU	👻 🙋 Fill
▶   读 读   🧮		
<b>→</b> ∓ ×	Form1.vb* ContactInfo.vb* Form1.vb [Design]* Start Page	
	A ContactInfo	▼ 10 (Declarations)
usable controls 5. Drag an item ext to add it to oolbox.	Public Class ContactInfo Public Name As String Public Address As String Public Tel As String Private Notes As String I End Class	

Next add the following code to the event handler of the Button1



Now your code is completely correct from syntax point of view, however it will not run correctly. If you run the code and then hit the button, then you get the following error:

Form1.vb <sup>@</sup> ContactInfo.vb <sup>@</sup> Fo	orm1.vb [Design] 🔒
Sutton1	👻 🍠 Click
Private Sub Butto Dim A As Cont A.Name = "smi A.Address; = "	Lth"
A.Tel = "1234	156" A NullReferenceException was unhandled Object reference not set to an instance of an object.
A.Tel = "1234 - End Sub	
	Object reference not set to an instance of an object.
- End Sub	Object reference not set to an instance of an object. Troubleshooting tips: Use the "new" keyword to create an object instance. Check to determine if the object is null before calling the met

This brings us to the second difference of class from structure. The variable A in the example is just a pointer to where the actual data is stored in memory, and there is not memory resources allocated to store the name, address and tel values for A. This is why you are getting the error.

To clarify things more, Let us say we have a structure to store exactly the same information:

```
Structure ContactInfoStruct
Dim Name As String
Dim Address As String
Dim Tel As String
End Structure
```

When you write

Dim V As ContactInfoStruct

Then what happens in memory is the following:

	v	Name Address Tel		
Memory A		0	bject Add Tel	

The variable V is allocated all the required memory resources. Unlike A, it only points to where the actual data are located in memory. So if there are no memory resources allocated, then A cannot point to them, and this is why you get the error. Now if you write:

# Dim A As ContactInfo

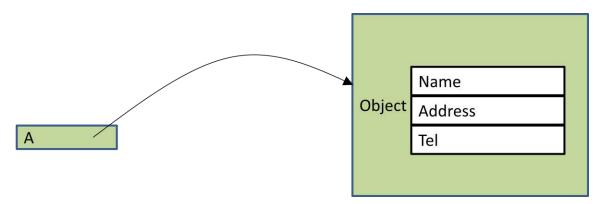
This creates a variable that points to no actual object (Nothing):



But if you write:

Dim A As New ContactInfo

Then an object is created, and A points to it:



Now use the New keyword, and test the code, you will see it runs without an error.

Dim A As New ContactInfo

Another method to do it is by using two steps:

Dim A As ContactInfo
A = New ContactInfo

This will have exactly the same effect. It is up to you to select which way to use. However in some cases you need to use the second format specially if you want to create and destroy the object linked by the same variable multiple times.

Now go to the class file and write down the following:

```
Public Sub SetContactInfo(ByVal NME As String, ByVal Addr As String,
ByVal Telephone As String)
    Name = NME
    Address = Addr
    Tel = Telephone
    End Sub
```

This subroutine allows you to fill the variables in the class. It is a normal subroutine except for the Public keyword placed before it. This means that you can call this subroutine form any other code block. It is similar to using Public with variables. This is useful if you want to hide complex functions and subroutines from outside access and provide small number of function to use with your class. Now to test this subroutine, Modify the Button1 event handler to be like this:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim A As ContactInfo
    A = New ContactInfo
    A.SetContactInfo("Smith", "UK", "123456")
End Sub
```

As you can see it is used the same way variables are accessed. You write the variable name (in this case A), followed by dot (.), followed by the function/subroutine (SetContactInfo). This is interpreted as call the function (SetContactInfo) and use the fields/attributes of A. If you the subroutine code:

Name = NME Address = Addr Tel = Telephone

This is translated to:

A.Name = NME A.Address = Addr A.Tel = Telephone

Now if you are using another object:

```
Dim B As ContactInfo
B = New ContactInfo
B.SetContactInfo("Michel", "US", "123456")
```

The subroutine call will be interpreted as:

B.Name = NME
B.Address = Addr
B.Tel = Telephone

and so on.

Now write down the following code in the event handler and run it:

```
Private Sub Buttonl_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Buttonl.Click
    Dim A As New ContactInfo
    Dim B As New ContactInfo
    A.SetContactInfo("Smith", "UK", "123456")
    B.SetContactInfo("Michel", "US", "456789")
    MsgBox(A.Name)
    MsgBox(A.Address)
    MsgBox(A.Tel)
    MsgBox(B.Name)
    MsgBox(B.Address)
    MsgBox(B.Tel)
End Sub
```

As you can see the code is easier to understand, and you don't have to fill the fields/attributes of the contact one by one. Now we will improve the way we enter the data by reading the information from a dialog. Right click your project and select Add->Windows Form

Script File

🛃 Login Form

🐼 Module 🔊 Report Wizard

Text File

📙 Local Database

Service-based Database

-

Ŧ

User Control (WPF)

	- ×	Solution Explor	er -	TestClass
				国 &
	Ę	Solution "		Class' (1 project)
Val e As System.	EventArgs) Hand			Build Rebuild Clean Publish
	🛅 New It	em.a		Add 🕨
	Carden and and and and and and and and and an	g Item older		Add Reference Add Service Reference
	🛅 Wind	ws Form	Ł	View Class Diagram
		ontrol		Set as StartUp Project
	Compo	onent		Debug •
	Modul	e	*	Cut
	Class		3	Paste
Select Dialog, and a	assign the name Read	ContactInfoE	Dial	log:
ciai 2 dows Forms	: COM Class 오 Custom Contr	ol		월 Component Class 题 DataSet
orting	🔜 Dialog			📰 Explorer Form

r Windows Forms applications

rkflow F

Ad

📸 Interface

🔝 Report 🛃 Resources File

LINQ to SQL Classes

MDI Parent Form

Splash Screen

🔠 User Control

🔟 Local Database Cache

Then select Add, the dialog design appears.

ms <u>^</u>	ReadContactInfoDialog	
E	ka S	
n		OK Cancel

Add three labels, and three text boxes, and make the dialog look like this:

100 00		- Contraction
<u>10</u> 10 10 10	目前題報 ●発幕# 咖啡	
- 4 ×	ReadContactInfoDialog.vb [Design]* Start Page Form1.vb ContactI	nfo.vb Form1.vb [Des
*	Read Contact Information	
	Name	
	Address	
	Telephone	
bars	OK Cancel	
rDialog		
pc		

If you check the code of the OK & Cancel buttons, you will find that it is already written. This code is the default behavior for a dialog, so leave it as it is.

Next we will add a subroutine to read contact information. Go to the class file and write the following:

```
Public Sub ReadContactInfo()

ReadContactInfoDialog.TextBox1.Text = ""
ReadContactInfoDialog.TextBox2.Text = ""
ReadContactInfoDialog.TextBox3.Text = ""

If ReadContactInfoDialog.ShowDialog = DialogResult.Cancel Then
        Exit Sub
End If
Name = ReadContactInfoDialog.TextBox1.Text
Address = ReadContactInfoDialog.TextBox2.Text
Tel = ReadContactInfoDialog.TextBox3.Text
End Sub
```

The first part clears the text boxes from all previous input. The if statement part checks if the user hit the cancel button, and exits the subroutine if so. If not, the execution continues to the last part, there the content of the text boxes are copied into the variables of the class. To test it modify the code of the Button1 for the main window (Form1) to be like this:

```
Private Sub Buttonl_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Buttonl.Click
    Dim A As New ContactInfo
    Dim B As New ContactInfo
    A.ReadContactInfo()
    B.ReadContactInfo()
    MsgBox(A.Name)
    MsgBox(A.Address)
    MsgBox(A.Tel)
    MsgBox(B.Name)
    MsgBox(B.Address)
    MsgBox(B.Tel)
```

End Sub

Now run the code, and hit the button, you should see something like this:

	*後   ▲ • 및 : 四, 地 勉 勉 A *   译 译   필 별   □ & · · · · · · · · · · · · · · · · · ·	
• Thre	ad: 🔄 🍸 🏹 Stack Frame:	
A ReadCont	tactInfoDialog.vb [Design] Form1.vb ContactInfo.vb F	orm1.vb [Design] 🗎
ml	🖳 Form1 🖂 🖾	
Button1_C		cem.EventArgs)
ContactI	Buttl Read Contact Information	
s New Cont		
ContactIn		
1223	Name	
ontactInfo		
ontactInfo	Address	
A.Name)	Telephone	
A.Address)		
A.Tel)	OK Cancel	
50.00 mm # 11		
3.Name)		-
3.Address)	S. 27 S.	-
3.Tel)		
· · ·		

Enter the information and hit OK. Another window appears, fill the information of the second contact and hit OK. After that you should be able to see the details of each contacts appear in separate message boxes.

Instead of using the message box to display the contact information, we will create another dialog to display such info. Just Add another dialog to the project as we did before and name it DisplayContactInfo. And make it look like the following:

w	Project	Build	Debug	Data	Format	Tools	Window	v Help			
		Qa (	3 - 7 -	(2 - 1	3 • D.	De De	bug	• Any CPL	J.	- 12	FillText
1	<u>ii</u> -()o <u>t</u> ()		10 EE3 :	ji:   060	)-[]-c D[]o +-+ →+	₩ **	· · · · · · · · · · · · · · · · · · ·	e*   🗗 🛛	10.81		
	- 7 ×	Di	isplayCont	actInfo.	vb (Desig	gn]* Sta	rt Page	ReadContact	InfoDialog.vb	ReadCon	tactInfol
bai	5		DisplayCor Name Address Telephone	ntactInfo	ОК		×				
erDi	alog										

Make sure to only remove the cancel button, and keep the OK button there. Also make sure that all textboxes are read only. Go next to the class file and add the following subroutine:

```
Public Sub DisplayContact()
   DisplayContactInfo.TextBox1.Text = Name
   DisplayContactInfo.TextBox2.Text = Address
   DisplayContactInfo.TextBox3.Text = Tel
   DisplayContactInfo.ShowDialog()
End Sub
```

This is much smaller code since it just displays the information of the object. To test that, update the code of Button1 in Form1

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim A As New ContactInfo
    Dim B As New ContactInfo
    A.ReadContactInfo()
    B.ReadContactInfo()
```

```
A.DisplayContact()
B.DisplayContact()
```

End Sub

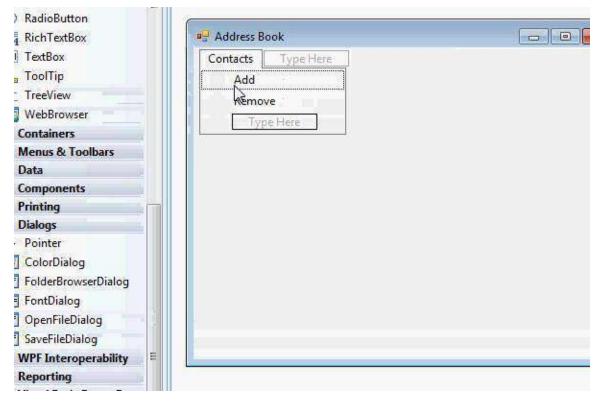
Run the code and you will see that you can display the information in the form:

15 mm

k(ByVal Info	Form1	Euster Ohi	iaat Rul			Args) Handles
					20 AL	
	But	tton1		577	D	
		DisplayConta	ictInfo	X		-
		Name	smith			
		Address	\$5	67		
		Telephone	ff	*0		
			ОК			
		L_	OR			

As you can see the code in Button1\_Click subroutine is very straightforward and easy to understand. You don't have to worry about the internal details of the class. All you need is to break your problem/your program into a number of logical units/classes each has its own data and functions, and then you combine them together to solve the main problem. Classes makes such thing easier to do.

Now our simple class is almost ready, so we are starting to create the main user interface now. Remove the Button1 from the Form1 window and add a menu strip control. Create the menu entries shown below:



Also add a data grid view, and call it DGV, add three columns to it (one for name, one for address, and one for tel). Disable adding, editing and deletion of rows. You should have something similar to the following:

File Edit View Project Bu	ild Debug Data Format	Tools Window He	lp		
🕼 • 🔤 • 💕 🗶 🕼 🛝 🖄	日日・ロ・ロ・ロ	🕨 Debug 🔹 A	ny CPU	• 🙆	FillTextBo
(1) [1] [1] [2] [2] [2] [2] [2] [2] [2] [2] [2] [2	司回路會會略點	망 은 한 방 아	日日  もち  日		
Toolbox + P X	Form1.vb <sup>+)</sup> Form1.vb [Desig				tactinfo.vi
RichTextBox	12				
abl TextBox	Address Book				
La ToolTip	Contacts				
TreeView	Name	Address	Telephone		
🛐 WebBrowser	name	Address	Telephone		
Containers					
🗄 Menus & Toolbars					
🗄 Data					
E Components					
Printing				6	
Dialogs					
Pointer					
S ColorDialog					
FolderBrowserDialog					
FontDialog					
🛃 OpenFileDialog					
🛫 SaveFileDialog					
WPF Interoperability					
Reporting					
Visual Basic PowerPa	3			6	

Double click the form and the editor opens, add the following code after Class Form1

Dim ContactList(0 To 999) As ContactInfo

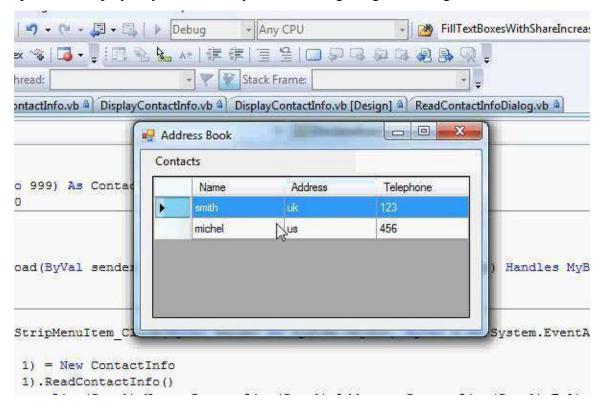
```
Dim C As Integer = 0
```

The ContactList is an array of type contact info. Each element of this array can point to an object of type ContactInfo, but when the array is created it is pointing to Nothing. C is used to tell how many objects are there in the array. When the program starts the number of elements is Zero.

Next add the following code to the Add menu item:

```
Private Sub AddToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddToolStripMenuItem.Click
    C = C + 1
    ContactList(C - 1) = New ContactInfo
    ContactList(C - 1).ReadContactInfo()
    DGV.Rows.Add(ContactList(C - 1).Name, ContactList(C -
1).Address, ContactList(C - 1).Tel)
    End Sub
```

This subroutine will add new contact, read the information of that contact, and then update the display. Try this out and you should be getting something like this:



Now the remove code should be like this:

```
Private Sub RemoveToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RemoveToolStripMenuItem.Click
        If DGV.SelectedRows.Count = 0 Then
           Exit Sub
       End If
       Dim I As Integer
       Dim N As String
       N = DGV.SelectedRows(0).Cells(0).Value
        For I = 0 To C - 1
            If ContactList(I).Name = N Then
                DGV.Rows.Remove(DGV.SelectedRows(0))
                Dim J As Integer
                For J = I + 1 To 999
                    ContactList(J - 1) = ContactList(J)
                Next
                C = C - 1
                Exit Sub
            End If
       Next
   End Sub
```

Notice that we are removing the contact from the display and from the array itself. Try adding and removing few contacts and see how it works. So this concludes the chapter. There will be more about classes in the next chapter. However there is some important things that you must keep in mind. A variable of a class is a pointer only. A good example to understand this is if you write the following code:

```
Dim A As New ContactInfo
Dim B As ContactInfo
A.Name = "Smith"
B = A
B.Name = "John"
```

In the end of execution of such code, both A and B will have John as the name value. Any change to A or B will affect the other one because simply they both point to the same location in memory (point to the same object in memory). But if A & B are structures:

```
Dim A As ContactInfoStruct
Dim B As ContactInfoStruct
A.Name = "Smith"
B = A
B.Name = "John"
```

Then A will be independent of B and changes in A will not affect B and vice versa.

**Chapter 18: Classes Initialization and Finalization** 

### **Classes Initialization and Finalization**

The previous chapter showed how to create a class, and how to add methods, and attributes to it. Today we see how to initialize the objects using the New method. First open the previous class example "testclass". Add a new class to the project, and call it: ContactList. This class will be used to store the contact information in the array and manage it . In the class file add the following code:

```
Dim ContactArr() As ContactInfo ' the array of object,
all elements points to nothing
Dim C As Integer ' the number of objects
in the array
```

These are used to store the contact information, and the number of elements in the array used. Next add the following method:

```
Public Sub AddNewContact()
C = C + 1 ' the number of objects
increases by one
ContactArr(C - 1) = New ContactInfo ' create the object
ContactArr(C - 1).ReadContactInfo() ' read the information
End Sub
```

This one adds a new contact, then add:

```
Public Sub RemoveContact(ByVal Name As String)
' search for the contact
For I = 0 To C - 1
If ContactArr(I).Name = Name Then
```

```
' next remove the contact from the array by shifting the
other objects
Dim J As Integer
For J = I + 1 To 999
ContactArr(J - 1) = ContactArr(J)
Next
' the number of elements reduces by one
```

```
C = C - 1
' exit the block
Exit Sub
End If
Next
End Sub
```

Which will remove a contact based on name. Also, add the following method to fill the data grid view:

```
Public Sub FillDGV(ByVal DGV As DataGridView)
    ' clear the data grid view
    DGV.Rows.Clear()
    Dim I As Integer
    ' loop over all the contacts
    For I = 0 To C - 1
        ' add contact information
        DGV.Rows.Add(ContactArr(I).Name, ContactArr(I).Address,
ContactArr(I).Tel)
    Next
End Sub
```

Now comes the constructor, write down the following:

```
Public Sub New()
    ' first constructor, set the number of elements to zero, and set
array size to 1000
    C = 0
    ReDim ContactArr(0 To 999)
End Sub
```

The name of this method is: New, and by default, when the compiler sees this, it knows that this method should be called automatically as soon as the object is created. So basically this method tells the computer to set the value of the counter C to zero, and

make the array capable of storing 1000 objects as soon as the ContactList object is created. To test this, Go to the form, and modify the code to be like this:

```
Public Class Form1
   Dim OBJ As ContactList
    Private Sub AddToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddToolStripMenuItem.Click
        OBJ.AddNewContact()
        OBJ.FillDGV(DGV)
   End Sub
    Private Sub RemoveToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RemoveToolStripMenuItem.Click
        ' check if no rows are selected, if so no need to execute
further code, exit the subroutine
        If DGV.SelectedRows.Count = 0 Then
            Exit Sub
        End If
        Dim N As String
        ' get the selected name, it is the first column (cell zero)
        N = DGV.SelectedRows(0).Cells(0).Value
        OBJ.RemoveContact(N)
        OBJ.FillDGV(DGV)
   End Sub
```

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

OBJ = New ContactList()

```
OBJ.FillDGV(DGV)
```

End Sub

End Class

Check out the code, and specifically the Form1\_Load subroutine. When the line:

OBJ = New ContactList()

is executed, the New subroutine is called directly. You don't have to do the call yourself, it is automatic. Also the constructor is executed only once.

So basically the constructor helps you prepare your object before using it. To make sure the constructor is being called, try to add a **MsgBox** call in the New method and see how it works.

You can actually create a number of different constructors, and later on you can choose which one to use based on the parameters you pass to it. For example, let us add another constructor to our class:

```
Public Sub New(ByVal NoOfReads As Integer)
   ' second constructor, set number of elements to zero, and set
array size to 1000
   C = 0
   ReDim ContactArr(0 To 999)
   ' add the contacts
   Dim I As Integer
   For I = 0 To NoOfReads - 1
        Me.AddNewContact()
   Next
End Sub
```

This constructor allows you to read a number of contacts as soon as you initialize the object without the need of going to the menu and select add contact. In order to call it, simply use it like this:

```
OBJ = New ContactList(3)
```

When the compiler sees the parameters (3), it searches for the constructor that accepts an integer as a parameter and calls it. You can create as many constructors as you need. The important thing is that the constructor name is always New, and each constructor should have different parameters (either in number or in data type to help the compiler distinguish them). Try the new constructor, and see how it works.

The last thing is the destructor. A destructor or finalizer is a method that is called when an object is destroyed, i.e. its resources are returned into memory. Try adding the code below:

```
Protected Overrides Sub Finalize()
   ' this is how to terminate a class
   Dim I As Integer
   For I = 0 To C - 1
        ContactArr(I) = Nothing
   Next
   MyBase.Finalize()
```

End Sub

Don't worry about the **MyBase**, or **Protected**, or the **Overrides** keywords for now, we will check these in later tutorials, but for now, just keep in mind that this one is being called when the object is destroyed. As you can see what we are doing here is we are looping over all the contactinfo objects and set them to nothing (which means we don't need them anymore, and we want them to be destroyed). Then after that we destroy the object. Try this code out, and see what happens when you place an MsgBox in this method.

**Chapter 19: Classes and Inheritance** 

### **Classes and Inheritance**

In the previous two tutorials, the definition of classes, methods, and their initialization is discussed. This tutorial is about how to perform inheritance. The same example used in the last tutorial is being used here as well.

In many cases you want to take an existing class and extend its functionality. In our previous example the class **ContactList** has **ContatArr()** which is an array used to store the contacts, and the counter **C** which is used to tell how many elements we are using in the array. It also has methods to add a contact, remove a contact, and display the contacts in a **DataGridView**. What we want here is to create a new class that has the same methods and properties as **ContactList**, and also has a **Sort** method which allows you to sort the contacts by name.

To do so simply add another class to your project, and call it **ContactsWithSort**. The first line of code in the class should be:

#### Inherits ContactList

The keyword **Inherits** here tells the complier that the class has behave in the same way as **ContactList**. In other words it is like copying the code of ContactList and pasting it to the new class (for now you can think about it like this, it makes things easier). Now let us try to add the new method to the class...

```
Public Sub Sort() ' this class has another method called sort.
Dim I As Integer
Dim F As Boolean
Dim Contact As ContactInfo
Do
F = False
For I = 0 To C - 2
If ContactArr(I).Name > ContactArr(I + 1).Name Then
F = True
Contact = ContactArr(I)
ContactArr(I) = ContactArr(I + 1)
ContactArr(I + 1) = Contact
End If
```

Next Loop While F End Sub

Now if you try to run the program (even though you did not use the Sort method or the new class itself) you will get an error. The error is for using C and ContactArr. The error says that these variables are private. This brings up the issue of the accessibility of variables.

When you define a variable in a class you can set its accessibility level to the following:

**Public** : this means that the variable can be access inside or outside the class.

**Private**: this means that the variable can be accessed only inside the original class it is created in.

**Protected**: this means that the variable can be accessed only in the class and all inherited classes.

So let us check this using the following example:

```
Public Class test
   Dim A As Integer
   Private B As Integer
   Public C As Integer
   Protected D As Integer
    Public Sub SetA(ByVal I As Integer)
       A = I
   End Sub
   Public Sub SetB(ByVal I As Integer)
        B = I
   End Sub
    Public Sub SetC(ByVal I As Integer)
        C = I
   End Sub
    Public Sub SetD(ByVal I As Integer)
       D = I
```

```
End Sub
End Class
```

In the example, **A** is treated as private. So if you add this method to the class:

```
Public Sub SetA(ByVal I As Integer)
        A = I
End Sub
```

It works perfectly fine. However, if you add the following code into a form or module:

```
Dim Q As New test Q.A = 10
```

This would trigger an error because A should only be accessed from within the class. Now let us check B which is private. If this is a method in the class, then it works.

```
Public Sub SetB(ByVal I As Integer)
        B = I
End Sub
```

But if you add the following code into any place other than the class test, you get an error.

```
Dim Q As New test Q.B = 10
```

So it works exactly like private. Next let us try to work with C which is Public.

```
Public Sub SetC(ByVal I As Integer)
    C = I
End Sub
```

This obviously works fine since it is in the same class (test). If you write the following code in any other place other than the class test, then it works perfectly fine.

```
Dim Q As New test
Q.C = 10
```

This works because the variable C here is public which means it can be accessed from any other place. Now let us check the last one D which is protected. The method within the class again has no problem

```
Public Sub SetD(ByVal I As Integer)
   D = I
End Sub
```

If you want to access the variable D from outside the class it is treated like private, but it has some special treatment, which we will see later.

```
Dim Q As New test Q.D = 10
```

So this triggers an error. Now let us go back to our example and see why we can't access the variable **C** and **ContactArr**. We used (Dim) for these two which means they are treated like private. As we have seen before that private variables in a class can not be accessed from outside the class itself. So we want to make them accessible. Making these variables public means that they will be accessed from any part of the project, which is not a good idea. If you change these variables' visibility to protected, then the classes inherited from them will be able to access these. An access from any other location is denied. To test this try to create a class test2 inherited from test.

```
Public Class test2
Inherits test
Public Sub SetAll()
A = 10 ' error
B = 20 ' error
C = 30 ' correct
D = 40 ' correct
End Sub
End Class
```

Here A is not accessible in this class simply because it is private in the original class. B is the same so it causes the same problem. C is public in class test, so it is accessible here and everywhere else. D is protected so it is accessible in test2. The table below summarizes how these work:

Accessibility	<b>Base Class</b>	Inherited Class	Outside the Class
Dim	Accessible	Not Accessible	Not Accessible
Private	Accessible	Not Accessible	Not Accessible
Public	Accessible	Accessible	Accessible
Protected	Accessible	Accessible	Not Accessible

So going back to our example, set each of C and ContactArr in ContactList class to protected. You will see the code now is correct.

Next, modify the object in the form to use the new class:

Dim OBJ As ContactsWithSort

And modify the code of initialization of OBJ in the load event of the form:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        OBJ = New ContactsWithSort()
        OBJ.FillDGV(DGV)
    End Sub
```

Finally, add a menu item to sort the contacts, and write the following in the event handler:

OBJ.sort()
OBJ.FillDGV(DGV)

Run the code check it out. Below is the full code of the **ContactList** class:

```
Public Class ContactList
```

```
Protected ContactArr() As ContactInfo ' the array of object,
all elements points to nothing
    Protected C As Integer
                                              ' the number of objects
in the array
    Public Sub AddNewContact()
       C = C + 1
                                               ' the number of objects
increases by one
       ContactArr(C - 1) = New ContactInfo ' create the object
       ContactArr(C - 1).ReadContactInfo() ' read the information
    End Sub
    Public Sub RemoveContact (ByVal Name As String)
        ' search for the contact
       For I = 0 To C - 1
            If ContactArr(I).Name = Name Then
                ' next remove the contact from the array by shifting the
other objects
                Dim J As Integer
                For J = I + 1 To 999
                    ContactArr(J - 1) = ContactArr(J)
                Next
                ' the number of elements reduces by one
                C = C - 1
                ' exit the block
               Exit Sub
           End If
       Next
    End Sub
    Public Sub FillDGV(ByVal DGV As DataGridView)
        ' clear the data grid view
        DGV.Rows.Clear()
       Dim I As Integer
        ' loop over all the contacts
```

```
For I = 0 To C - 1
            ' add contact information
            DGV.Rows.Add(ContactArr(I).Name, ContactArr(I).Address,
ContactArr(I).Tel)
       Next
   End Sub
    Public Sub New()
        ' first constructor, set the number of elements to zero, and set
array size to 1000
       C = 0
       ReDim ContactArr(0 To 999)
   End Sub
    Public Sub New(ByVal NoOfReads As Integer)
        ' second constructor, set number of elements to zero, and set
array size to 1000
       C = 0
       ReDim ContactArr(0 To 999)
        ' add the contacts
        Dim I As Integer
        For I = 0 To NoOfReads - 1
           Me.AddNewContact()
       Next
   End Sub
   Protected Overrides Sub Finalize()
        ' this is how to terminate a class
       Dim I As Integer
        For I = 0 To C - 1
            ContactArr(I) = Nothing
       Next
       MyBase.Finalize()
   End Sub
End Class
```

Next is the code for the ContactsWithSort class

```
Public Class ContactsWithSort
    Inherits ContactList
      ' this tells the compiler that this class has the same behaviour
      of ContactList
   Public Sub Sort() ' this class has another method called
sort.
       Dim I As Integer
       Dim F As Boolean
        Dim Contact As ContactInfo
        Do
            F = False
            For I = 0 To C - 2
                If ContactArr(I).Name > ContactArr(I + 1).Name Then
                    F = True
                    Contact = ContactArr(I)
                    ContactArr(I) = ContactArr(I + 1)
                    ContactArr(I + 1) = Contact
                End If
            Next
        Loop While F
    End Sub
End Class
And finally the code of the form:
Public Class Form1
    Dim OBJ As ContactsWithSort
    Private Sub AddToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AddToolStripMenuItem.Click
       OBJ.AddNewContact()
       OBJ.FillDGV(DGV)
   End Sub
```

```
Private Sub RemoveToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RemoveToolStripMenuItem.Click
        ' check if no rows are selected, if so no need to execute
further code, just exit the subroutine
        If DGV.SelectedRows.Count = 0 Then
            Exit Sub
        End If
        Dim N As String
        ' get the selected name, it is the first column (cell zero)
        N = DGV.SelectedRows(0).Cells(0).Value
        OBJ.RemoveContact(N)
        OBJ.FillDGV(DGV)
    End Sub
    Private Sub Form1 Load (ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
       OBJ = New ContactsWithSort()
       OBJ.FillDGV(DGV)
   End Sub
    Private Sub SortToolStripMenuItem Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
SortToolStripMenuItem.Click
```

```
OBJ.sort()
OBJ.FillDGV(DGV)
End Sub
End Class
```

The rest of the files don't need modification, they are the same. So as you can see inheritance allows us to extend the functionality of an existing class, and add some features to them. Chapter 20: Try & Catch

# Try & Catch

When you develop a vb.net application it is very common to get what is known as Runtime error. These are errors that might happen due to some wrong input for example, or some computational operation during program execution. To demonstrate the idea, create a simple vb.net application that read two values from the display and divide them. The code should be similar to this:

```
Dim A As Integer
Dim B As Integer
Dim C As Integer
A = TextBox1.Text
B = TextBox2.Text
C = A / B
MsgBox("The result is:" & C.ToString)
```

This code is correct, and should work fine. However if the value of B is zero (by entering a value of zero in textbox2), then you will get a runtime error. This is simply because you can not divide any number by zero (C=A/B).

This is just a simple example of the errors that you might get. VB allows you to catch such errors so that your program don't crash, and you can give a friendly message to the end user or treat the error. The way to do it is by using the try statement. It should be similar to the following:

```
Try

The code that could cause error goes here

Catch ex As Exception

The treatement of the error goes here

End Try
```

To use this one, you can rewrite the code as follows:

```
Dim A As Integer
Dim B As Integer
Dim C As Integer
A = TextBox1.Text
B = TextBox2.Text
```

```
Try
C = A / B
MsgBox("The result is:" & C.ToString)
Catch ex As Exception
MsgBox("Error")
End Try
```

What happens here is that the statements between Try and Catch are monitored for any errors. If an error happens, then the execution will be interrupted, and a new execution starts in the Catch part. So in the example above if the division is by zero, then a friendly message is displayed telling the end user that there is some kind of problem is there. You program will not crash in this case.

Another thing is that there is an object called **ex**. This one holds details about the error. You can get some details about the error itself. For example:

```
Try
C = A / B
MsgBox("The result is:" & C.ToString)
Catch ex As Exception
MsgBox(ex.Message)
End Try
```

Here the program will give the end user the detail of the error (the message property describes the error here). You can display other error details, or store them for debugging purposes. Some of these are ex.StackTrace which gives you the calls that caused the errors, and where the error exactly happened. So it is very useful.

The try statement can catch different set of errors, so using ex.Message is useful because it can tell you what kind of error you are getting and hence you can identify where the error is.

Last thing is that you can use a finally part with the Try statement

```
Try

The code that could cause error goes here.

Catch ex As Exception

The treatement of the error goes here.

Finally

A number of statements that always get executed.

End Try
```

This part is always executed regardless of the state of execution errors. This part can be eliminated, and you can place your code after the try statement resulting in exactly the same effect.

So this concludes the last chapter in this simple programming book. Hope you enjoyed learning the language and practicing some of the included examples.

Thank you.

mkaatr.