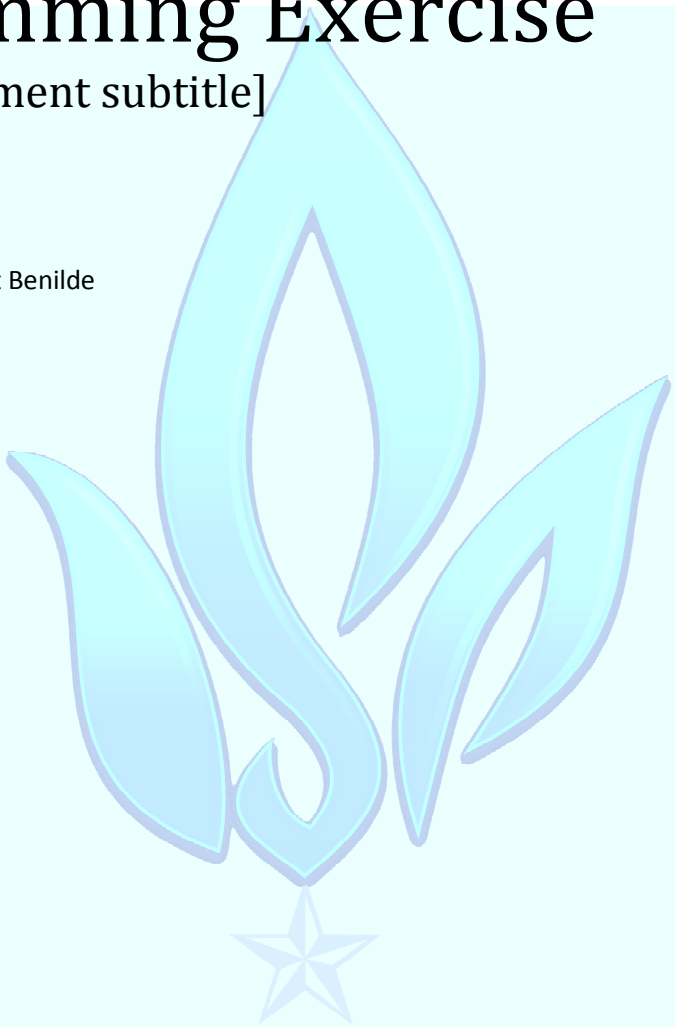
Basic Event-Driven Programming Exercise

[Type the document subtitle]

[Pick the date]

De La Salle-College of Saint Benilde
Eisen Lewis Sy



Title: Basic Event-Driven Programming Exercise (Properties, Events and Methods)

Activity Type: Exercise, Homework (Hands-On)

Pre-requisites: Before undergoing this activity, students must have knowledge in the following topics:

1.) Creation of a C# Windows Form Application projects using Visual Studio 2010
2.) Working knowledge on how to use the Visual Studio Integrated Development Environment (IDE)
    a. How to view the properties of an object using the Properties Window
    b. How to assign values to properties of an object using the Properties Window
    c. How to toggle from properties view to events view in the Properties Window
3.) How to manage projects to create, save, open or modify an existing project

Objective(s):

1.) Facilitate the understanding of event driven programming in C#
2.) Facilitate the exploration of objects and their commonly used properties
3.) Introduce simple object methods through code writing
4.) Introduce the importance of the Name property for all objects in a Windows Form Application
5.) Make use of the following objects, events and properties:

Object: Form	
Properties	**Events**
Name	
Text	
ControlBox	
FormBorderStyle	
Size	

Object: TextBox	
Properties	**Events**
Name	DoubleClick
Font	TextChanged
Multiline	
ReadOnly	
PasswordChar	
Anchor	
Size	
Location	

Object: Label	
Properties	**Events**
Name	Click
Text	
AutoSize	
TextAlign	
BackColor	
ForeColor	
Font	
Anchor	
Size	
Location	

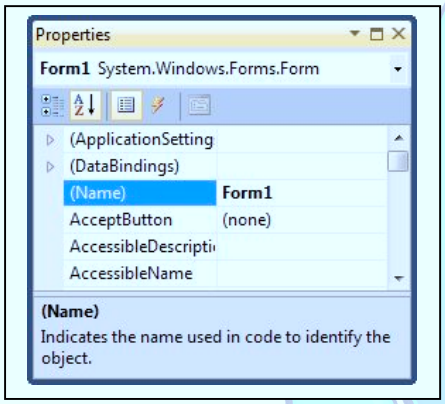
Object: Button	
Properties	**Events**
Name	Click
Text	MouseEnter
Font	MouseLeave
Enabled	
Visible	
Size	
Location	

Instructions:

Create a new C# Windows Form Application Project. Name the project as "***[Last Name]_[First Name]_[ID Number]_BEDP***" and ensure that the Toolbox, Solution Explorer and the Properties Window are accessible from your IDE.

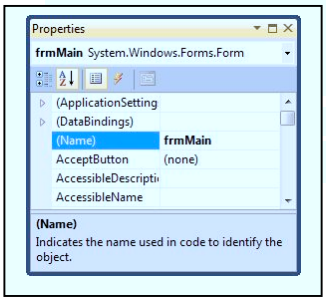
Part 1: Customizing the Form

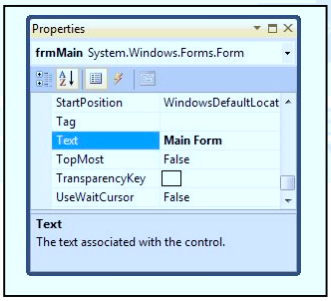
1.) Customize your form by clicking on it once, as soon as you do this the Properties Windows should automatically display properties related to the form object. You can confirm that the properties reflected on the Properties Window belong to the form by inspecting the Properties Window's drop down control. (See figure below)

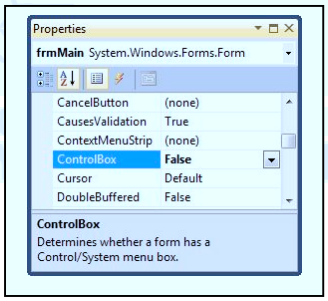


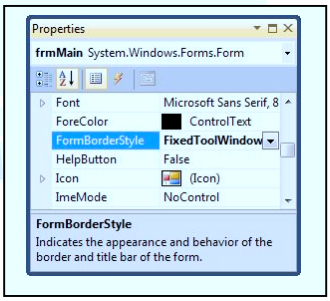
2.) Assign the specified values to the following form properties

Property	Property Value
(Name)	frmMain
Text	Main Form
ControlBox	False
FormBorderStyle	FixedToolWindow





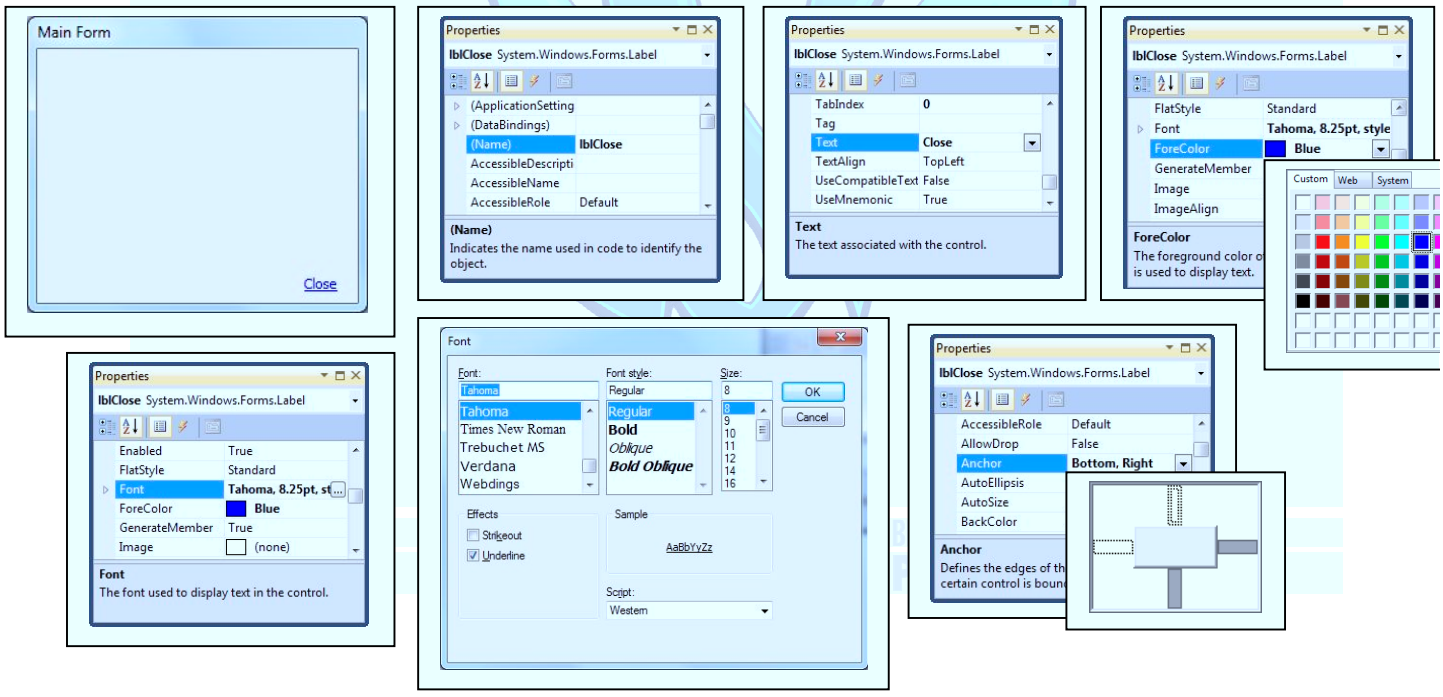




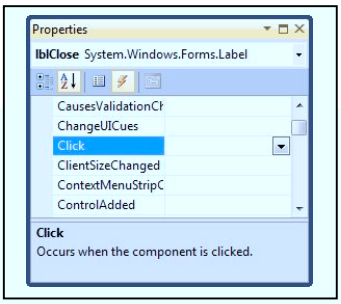
Part 2: Creating the Close Feature

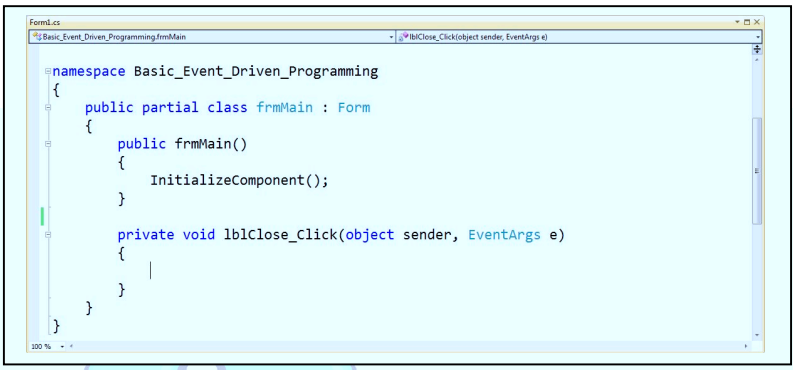
1.) Create a label and position it at the lower-right corner of the form and set the following properties to the corresponding values. (See figure below)

Property	Property Value
(Name)	lblClose
Text	Close
ForeColor	Blue
Font	Font: Tahoma Size: 8 Effects: Underline
Anchor	Bottom, Right This ties the label down so that it follows the bottom and right edge of the parent control (in this case the windows form). Try it out! Try to resize your form and see how the label reacts to it!

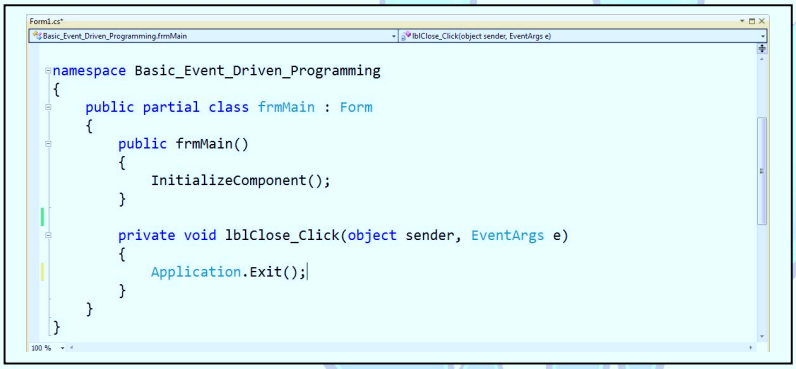


2.) To add the close functionality when you click this label, you will need to write codes on the Click event of the ***lblClose*** label. You can do this by switching to event view in the Properties Window, and double-clicking on the Click event of the said label. This brings you to the code view of the current form. (See figure below)





3.) You can close the application using a call to the code **`Application.Exit()`** or a call to the code **`Close()`**. The codes below makes use of **`Application.Exit()`**.



Note: You can now exit the program by clicking on the Close label.

Part 3: Creating the Save Feature

1.) Set the Size property of the form to (320, 316) (that is 320 pixels wide and 316 pixels high).

2.) Create 5 labels and change the property values of each label to the following:

Property	Property Value
(Name)	lblLastName
AutoSize	False
Text	Last Name
BackColor	Light Gray
Font	Font: Tahoma Size: 8 Effects: Underline
TextAlign	MiddleRight
Size	93, 17 (93 pixels wide, 17 pixels high)
Location	12, 29 (Set the location of the label to 12 pixels from the left edge of the form and 29 pixels from the top edge of the form. This essentially positions the label within the form)

Property	Property Value
(Name)	lblFirstName
AutoSize	False
Text	First Name
BackColor	Light Gray
Font	Font: Tahoma Size: 8 Effects: Underline
TextAlign	MiddleRight
Size	93, 17 (93 pixels wide, 17 pixels high)
Location	12, 55 (Set the location of the label to 12 pixels from the left edge of the form and 55 pixels from the top edge of the form. This essentially positions the label within the form)

Property	Property Value
(Name)	lblAddress
AutoSize	False
Text	Mailing Address
BackColor	Light Gray
Font	Font: Tahoma Size: 8 Effects: Underline
TextAlign	MiddleRight
Size	93, 17 (93 pixels wide, 17 pixels high)
Location	12, 81 (Set the location of the label to 12 pixels from the left edge of the form and 81 pixels from the top edge of the form. This essentially positions the label within the form)

Property	Property Value
(Name)	lblPassword
AutoSize	False
Text	Desired Password
BackColor	Light Gray
Font	Font: Tahoma Size: 8 Effects: Underline
TextAlign	MiddleRight
Size	93, 17 (93 pixels wide, 17 pixels high)
Location	12, 143 (Set the location of the label to 12 pixels from the left edge of the form and 143 pixels from the top edge of the form. This essentially positions the label within the form)

Property	Property Value
(Name)	lblType
AutoSize	False
Text	Type
BackColor	Light Gray
Font	Font: Tahoma Size: 8 Effects: Underline
TextAlign	MiddleRight
Size	93, 17 (93 pixels wide, 17 pixels high)
Location	12, 170 (Set the location of the label to 12 pixels from the left edge of the form and 170 pixels from the top edge of the form. This essentially positions the label within the form)

3.) Create 5 textboxes and change the property values of each textbox to the following:

Property	Property Value
(Name)	txtLastName
Font	Font: Tahoma Size: 8 Effects: Underline
Size	191, 21 (191 pixels wide, 21 pixels high)
Location	111, 28 (Set the location of the label to 111 pixels from the left edge of the form and 28 pixels from the top edge of the form. This essentially positions the label within the form)
Anchor	Top, Left, Right

Property	Property Value
(Name)	txtFirstName
Font	Font: Tahoma Size: 8 Effects: Underline
Size	191, 21 (191 pixels wide, 21 pixels high)
Location	111, 54 (Set the location of the label to 111 pixels from the left edge of the form and 54 pixels from the top edge of the form. This essentially positions the label within the form)
Anchor	Top, Left, Right

Property	Property Value
(Name)	txtAddress
Font	Font: Tahoma Size: 8 Effects: Underline
Size	191, 56 (191 pixels wide, 56 pixels high)
Multiline	True
Anchor	Top, Left, Right
Location	111, 80 (Set the location of the label to 111 pixels from the left edge of the form and 80 pixels from the top edge of the form. This essentially positions the label within the form)

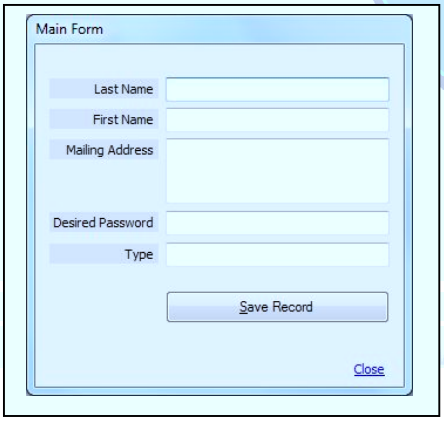
Property	Property Value
(Name)	txtPassword
Font	Font: Tahoma Size: 8 Effects: Underline
Size	191, 21 (191 pixels wide, 21 pixels high)
Location	111, 142 (Set the location of the label to 111 pixels from the left edge of the form and 142 pixels from the top edge of the form. This essentially positions the label within the form)
Anchor	Top, Left, Right
PasswordChar	*

Property	Property Value
(Name)	txtType
Font	Font: Tahoma Size: 8 Effects: Underline
Text	Normal
Size	191, 21 (191 pixels wide, 21 pixels high)
Location	111, 169 (Set the location of the label to 111 pixels from the left edge of the form and 169 pixels from the top edge of the form. This essentially positions the label within the form)
Anchor	Top, Left, Right
ReadOnly	True

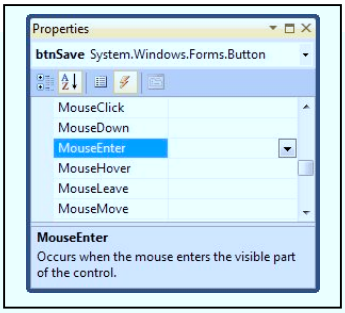
4.) Create a button and change the property values of the button to the following:

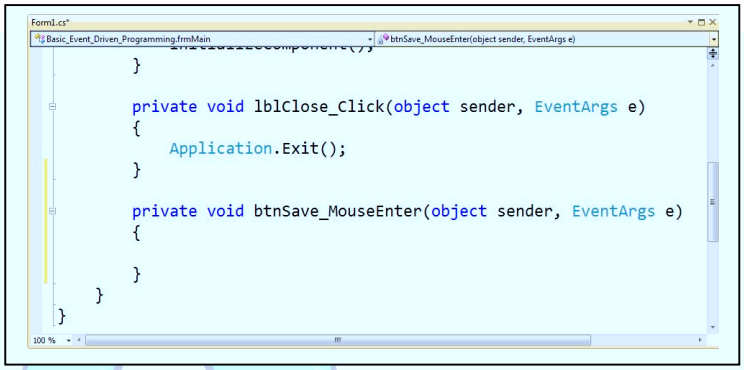
Property	Property Value
(Name)	btnSave
Font	Font: Tahoma Size: 8 Effects: Underline
Text	&Save Record (The ampersand symbol & allows you to use short cut key to click the button) Alt-S will trigger clicking the button.
Size	191, 28 (191 pixels wide, 28 pixels high)
Anchor	Top, Left, Right
Location	111, 210 (Set the location of the label to 111 pixels from the left edge of the form and 210 pixels from the top edge of the form. This essentially positions the label within the form)

5.) Your form should look like this at this point

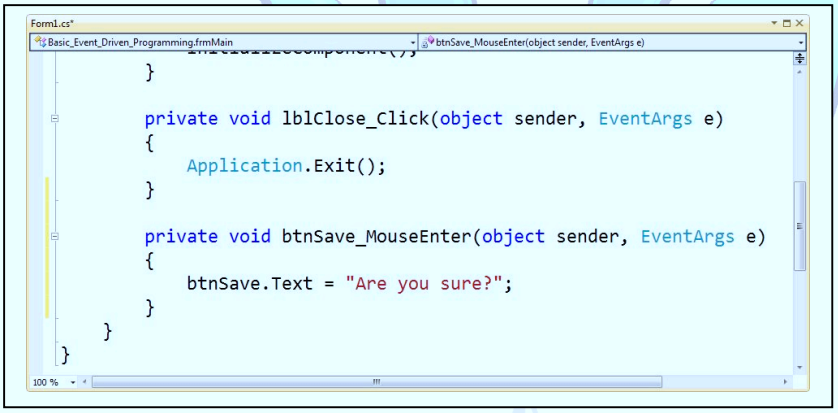


6.) Let's finish off this section by implementing the Save Record button, although we will not actually create codes to save the data, this item will show you how use a message box and make use of 2 more events to make clicking the button a little more interesting. To do this we'll need to write codes on three events Click, MouseEnter and MouseLeave. Let us begin with MouseEnter event, you can write codes to this event by double-clicking the event on the Properties Window. (See figure below)

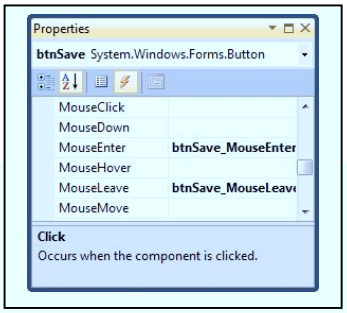


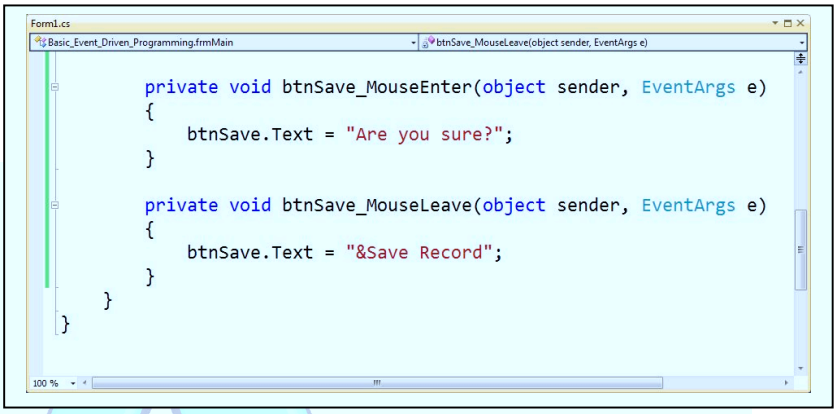


7.) We want the button text to change whenever the mouse enters the button's client area, so we can supply the codes below to change the button's Text property.

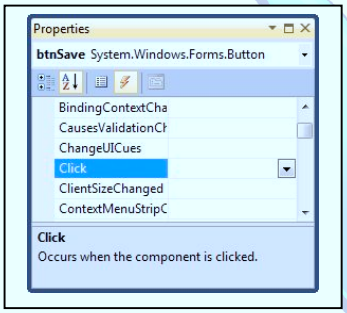


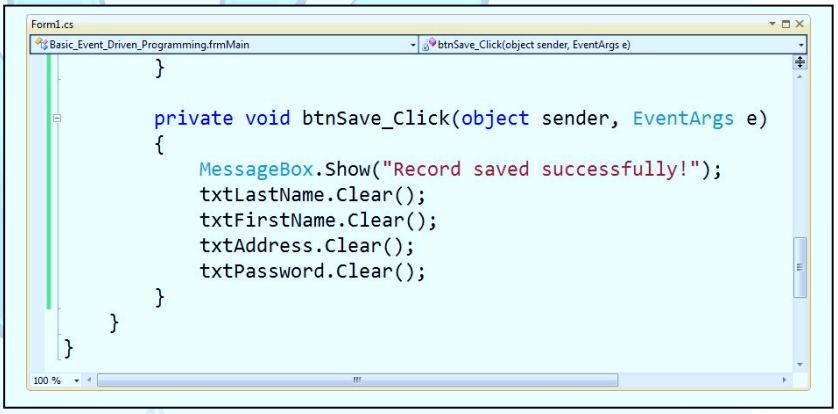
8.) We want the button to display “Save Record” when the mouse leaves the button’s client area, so we can supply the following codes to the button’s MouseLeave event.





9.) Finally, clicking the button should display a prompt to the user that saving was successful. We’re going to use a message box to show this message to the user. This can be done with the use of the button’s Click event and supplying the codes below. We’ll also add codes to clear the text for most text boxes right after the message box is acknowledged by the user.

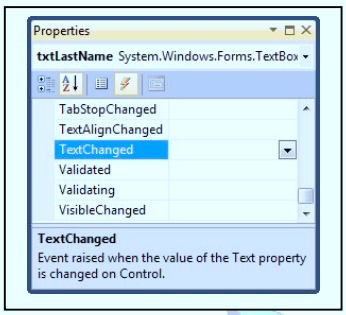


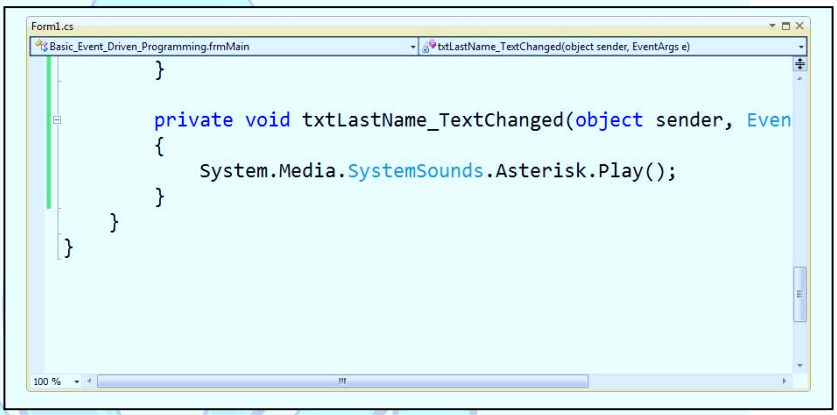


Part 4: Playing with Window Sounds

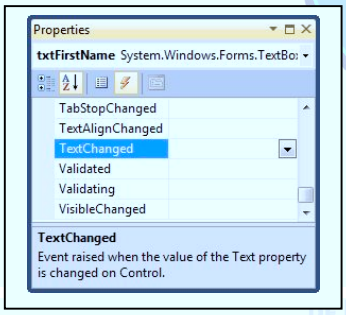
1.) This part of the exercise will teach you how to play basic windows sounds. The specific objective will be to play a short windows sound every time the user makes any changes to each text box in the form. Admittedly, this is not a practical thing to do as it is likely to irritate the user than to be helpful, but nonetheless it will help you to remember how to play simple sounds in a windows form application.

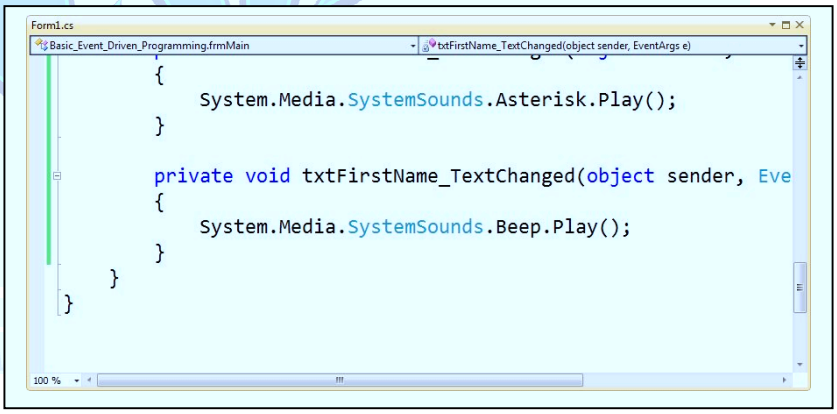
2.) Let's begin by adding this functionality to the first text box txtLastName. The codes would have to be written in this text box's TextChanged event so that each change made to the value on the text box would trigger the sound. The codes are supplied below.



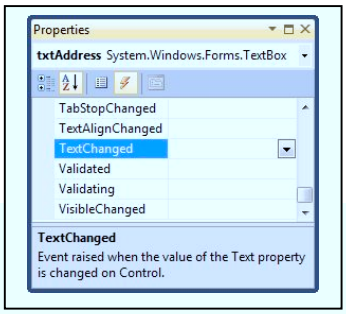


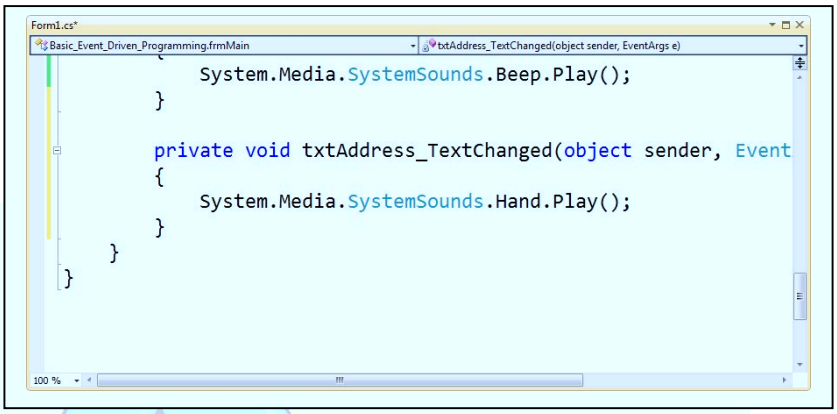
3.) Let's do the same thing for the next text box txtFirstName. The event and codes are shown below, although a different sound is played on this text box.



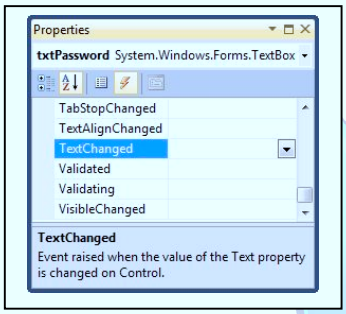


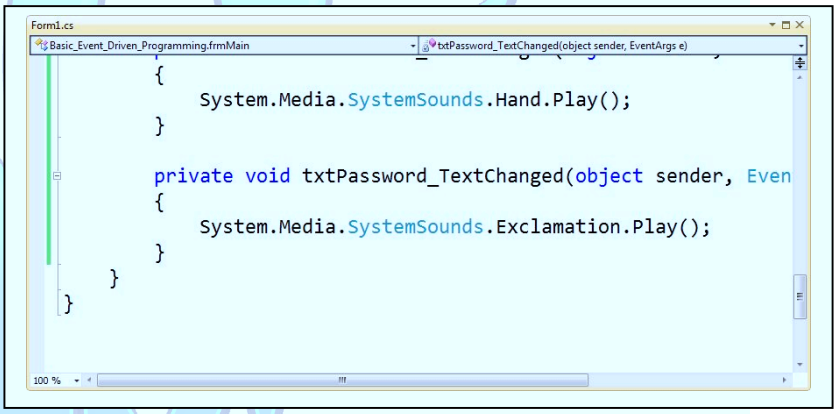
4.) Continuing on with the next text box txtAddress, we have the following event and codes.





5.) And finally for the last text box that we will do txtPassword, we have the following event and codes.



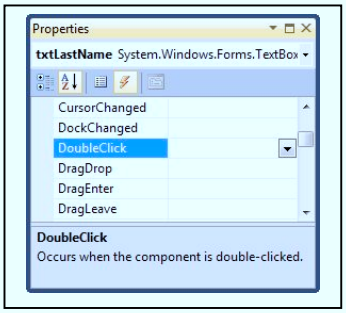


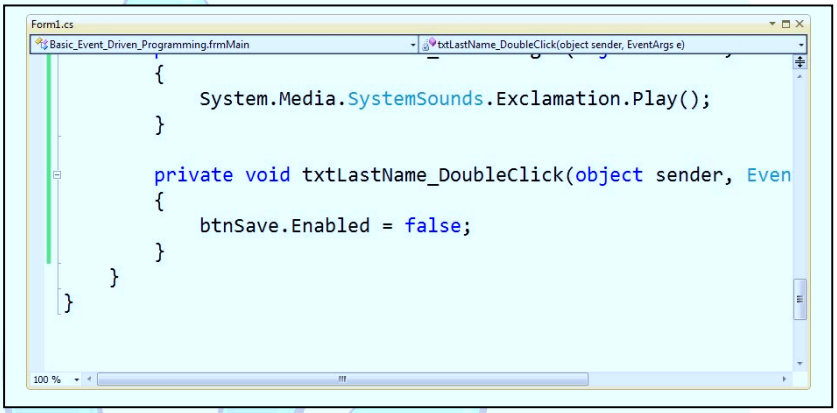
6.) No need to do this for txtType since we made txtType ReadOnly so that we won't be able to change the value written on txtType. If you try and run your program you should notice that typing text on the text boxes produces different sounds for different text boxes. You might find this interesting now but long term, it's quite irritating.

Part 5: Finishing the Program with Secret Features

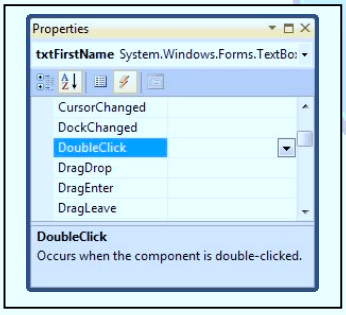
1.) The final part of the exercise will teach you how to enable and disable buttons or render them visible or invisible and will show you another event to work with Double-Click. The basic idea is that you can double-click a text box and this can enable/disable the button or render them visible/invisible.

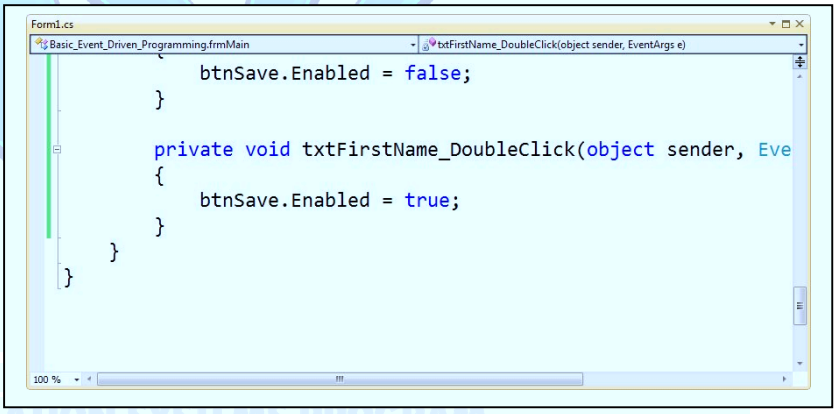
2.) Let's begin by using the DoubleClick event of a text box and write codes to disable the button btnSave. Let's do this for txtLastName, the event and codes are shown below.



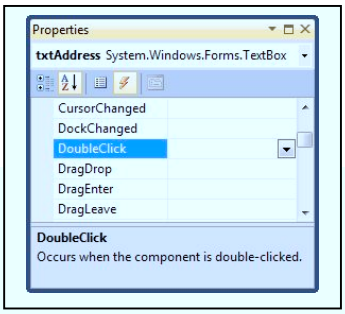


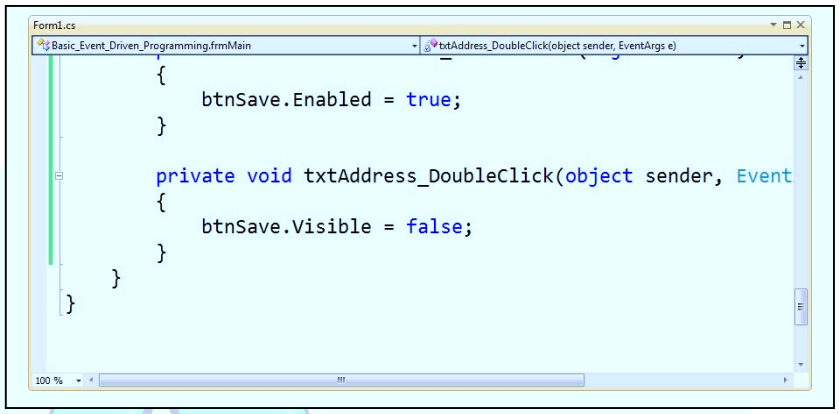
3.) Double-clicking txtFirstName should enable the button btnSave. The event and codes are shown below.



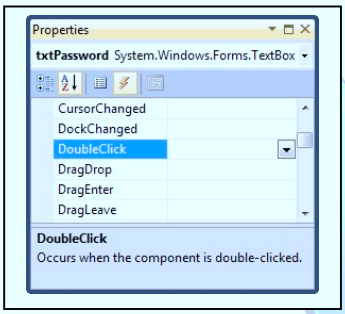


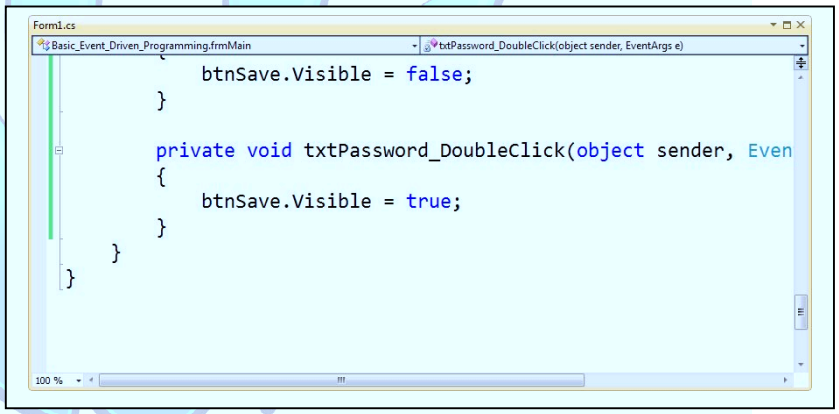
4.) Double-clicking txtAddress should render btnSave invisible. The events and codes are shown below.





5.) Finally, double-clicking txtPassword should make btnSave visible again. The events and codes are shown below.





6.) You can now try double-clicking one of the text boxes and see how the button behaves.
7.) This completes your activity.