Detecting and Responding to Keyboard Events

Windows Forms can detect key events. There are three types of keyboard events.

1 – KeyDown Event: This event raised as soon as the user presses a key on the keyboard, it repeats while the user holds the key down.

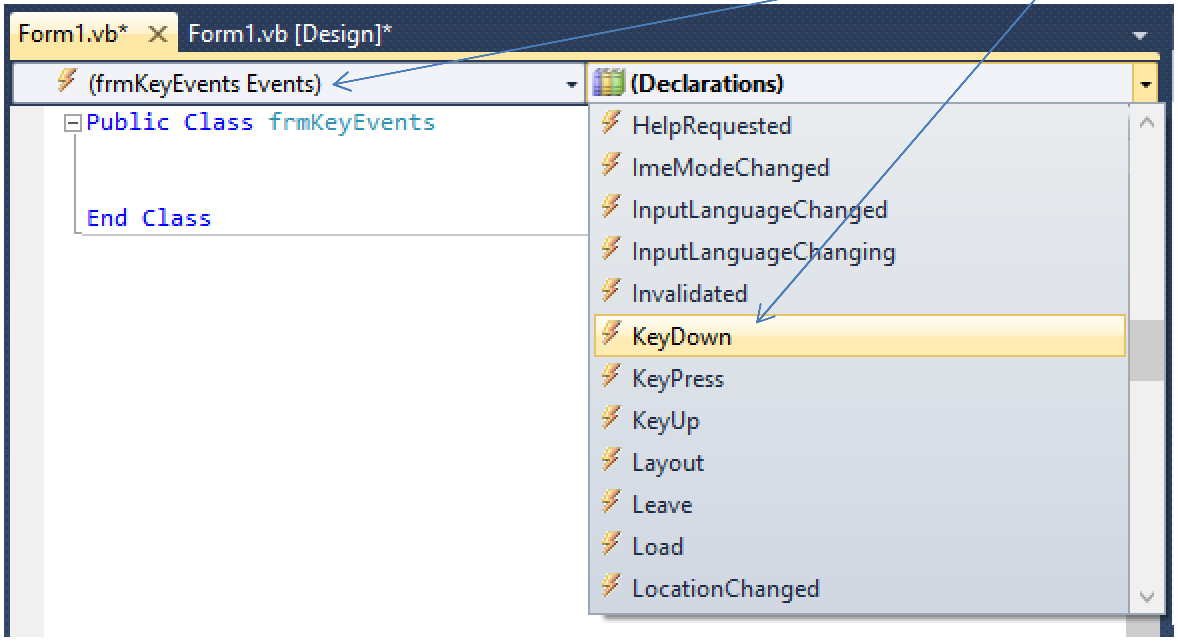
2 – KeyPress Event: : This event is raised for character keys while the key is pressed and then released by the user. This event is not raised by non-character keys (like arrow keys), unlike **KeyDown** and **KeyUp**, which are also raised for non-character keys. (*don't use this one*)

3 – KeyUp Event: This event is raised after the user releases a key on the keyboard.

Step 1 – Create a new Windows Forms application and add a Label and name it lblKeyInfo.

We are going to create a basic program that will detect when a key is pressed and released.

Once you have your Form and Label created and named, go to the Form event for **KeyDown**.



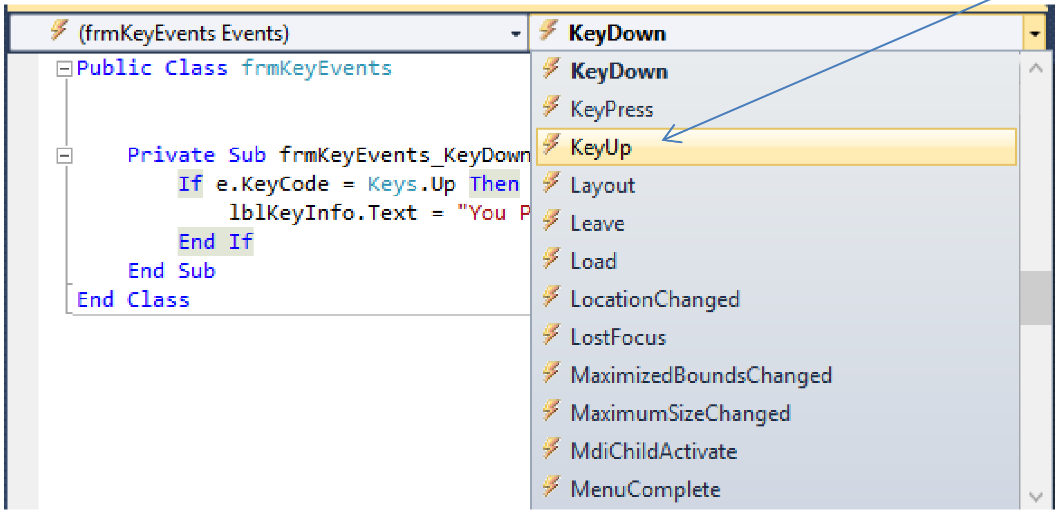
```
Public Class frmKeyEvents


End Class
```

We must tell our program how to react to specific KeyCodes. In order to do this we must use **If..Then** statements to act accordingly. The following code will display in a Label, a message that the Up-Arrow has been pressed. Put it in the **KeyDown** event.

```
If e.KeyCode = Keys.Up Then
            lblKeyInfo.Text = "You are pressing UP"
End If
```

The object '**e**' in the code above contains the information that triggered the event, so we will need to ask it what key was pressed by using **e.KeyCode**.

We also may want to detect when a key is released. To do this, go to the event menu for the **KeyUp** event.



```
Public Class frmKeyEvents


    Private Sub frmKeyEvents_KeyDown
        If e.KeyCode = Keys.Up Then
            lblKeyInfo.Text = "You P
        End If
    End Sub
End Class
```

Put the following code in the **KeyUp** event.

```
If e.KeyCode = Keys.Up Then
            lblKeyInfo.Text = "You released UP"
End If
```

Use **Else..If** statements to update lblKeyInfo with an appropriate message for the remaining arrow keys for the KeyUp and KeyDown events.

Things to try:

- Try detecting different key events
- Have the user change the background color to red, blue or green when the ‘r’, ‘b’ or ‘g’ keys are pressed
  - Change the color back to normal when they release the keys:
    - `Me.BackColor = DefaultBackColor`
- Move a PictureBox around using the arrow keys by changing its Location property (or its Left and Top properties).
- Update the **Moving a PictureBox with Points** tutorial to use key events instead of Button click events.

Troubleshooting: I added a Button, and now my Key Events aren’t working!!

- Your Form and each control has its own KeyDown/Up/Pressed events. Whatever control is in ***focus*** will receive the event.
  - This means that if you add a Button/TextBox (or any other control that can gain focus) to your Form and it gets ‘*focus*’, the Form KeyDown event will not be triggered.

Solution:

For your Form, set the **KeyPreview** property to **True** to have your Form detect key events when any control is in focus.