**Message Box class**

A [Message Box](https://msdn.microsoft.com/en-us/library/system.windows.forms.messagebox(v=vs.71).aspx) is a predefined dialog box that displays application-related information to the user. Message boxes are also used to request information from the user

Methods: - show (String, String, MessageBox Buttons, MessageBoxIcon)

**Events**

In our Window application there are two events are majorly

1. Key Press events
2. Mouse Events

**Key Press Events:** Occurs when a character. space or backspace key is pressed while the control has focus.

Key events occur in the following order:

1. [KeyDown](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.keydown(v=vs.110).aspx)
2. KeyPress
3. [KeyUp](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.keyup(v=vs.110).aspx)

The KeyPress event is not raised by non-character keys other than space and backspace; however, the non-character keys do raise the [KeyDown](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.keydown(v=vs.110).aspx) and [KeyUp](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.keyup(v=vs.110).aspx) events.

Use the [KeyChar](https://msdn.microsoft.com/en-us/library/system.windows.forms.keypresseventargs.keychar(v=vs.110).aspx) property to sample keystrokes at run time and to consume or modify a subset of common keystrokes.

To handle keyboard events only at the form level and not enable other controls to receive keyboard events, set the [KeyPressEventArgs.Handled](https://msdn.microsoft.com/en-us/library/system.windows.forms.keypresseventargs.handled(v=vs.110).aspx) property in your form's KeyPressevent-handling method to true.

**Validation in Window form using key press events**

Now most of the time the user needs validation in a Windows application, so before the user enters a wrong value (such as instead of text the user enters an integer) we can validate it using an event and some sort of coding.

public void charonly(KeyPressEventArgs e)  
 {  
   if (Char.IsNumber(e.KeyChar) || Char.IsSymbol(e.KeyChar) || Char.IsWhiteSpace(e.KeyChar) || Char.IsPunctuation(e.KeyChar))  
     {  
         MessageBox.Show("Only Char are allowed");  
         e.Handled = true;  
      }  
  }

        public void digitonly(KeyPressEventArgs e)  
        {  
            try  
            {  
                if (!(char.IsDigit(e.KeyChar) || char.IsControl(e.KeyChar) ||char.IsPunctuation(e.KeyChar)))  
                {  
                    e.Handled = true;  
                    MessageBox.Show("Enter only digit and decimal point.", "Alert!");  
                }  
            }  
            catch { }  
        }

**Mouse Events**

When you handle mouse input, you usually want to know the location of the mouse pointer and the state of the mouse buttons. This topic provides details on how to get this information from mouse events, and explains the order in which mouse click events are raised in Windows Forms controls.

1. [MouseDown](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.mousedown(v=vs.110).aspx) event.
2. [Click](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.click(v=vs.110).aspx) event.
3. [MouseClick](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.mouseclick(v=vs.110).aspx) event.
4. [MouseUp](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.mouseup(v=vs.110).aspx) event.