C# .NET PROGRAMMING: A GRAPHICAL APPROACH CLASS 6

ING. HAZAEL FERNANDO MOJICA GARCÍA

- EXCEPTION HANDLING
- GUI
- STRINGS

EXCEPTION HANDLING

An **exception** is an indicator that something is wrong with the execution of our program. The name “exception” comes from the fact that the problem can happen not very frequently, it’s an “exception to the rule”.

Handling exceptions allow programmers to make robust programs that can run even when some conditions prevents the program to be execute normally and without fatal errors.

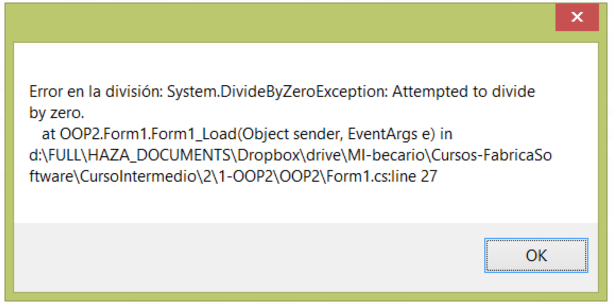
The class **System.Exception** is the main class for extending exceptions in C#.

A try block is used by C# programmers to partition code that might be affected by an exception. Associated catch blocks are used to handle any resulting exceptions. A finally block contains code that is run regardless of whether or not an exception is thrown in the try block, such as releasing resources that are allocated in the try block. A try block requires one or more associated catch blocks, or a finally block, or both.

The way of building a **try…catch** block is as follows.

```
try
{
        //ESTE CODIGO PODRIA CONTENER UN ERROR
        //QUE QUEREMOS MANEJAR
}
catch(Exception ex)
{
        //SI OCURRE UN ERROR EL FLUJO DEL PROGRAMA
        //CONTINUARA AQUI
}
finally
{
        //EXISTA O NO ERROR EL FLUJO DEL PROGRAMA
        //LLEGARA AQUI
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    int numero1 = 0;
    int numero2 = 0;
    try
    {
        numero1 = 5 - 5;
        numero2 = 3 / numero1;
    }
    catch(Exception ex)
    {
        numero2 = 3;
        MessageBox.Show("Error en la división: " + ex.ToString());
    }
    finally
    {
        string msg = "";
        msg += "numero1 = " + numero1.ToString() + "\n";
        msg += "numero2 = " + numero2.ToString() + "\n";
        MessageBox.Show(msg);
    }
}
```



```
Error en la división: System.DivideByZeroException: Attempted to divide
by zero.
   at OOP2.Form1.Form1_Load(Object sender, EventArgs e) in
d:\FULL\HAZA_DOCUMENTS\Dropbox\drive\MI-becario\Cursos-FabricaSo
ftware\CursoIntermedio\2\1-OOP2\OOP2\Form1.cs:line 27
```

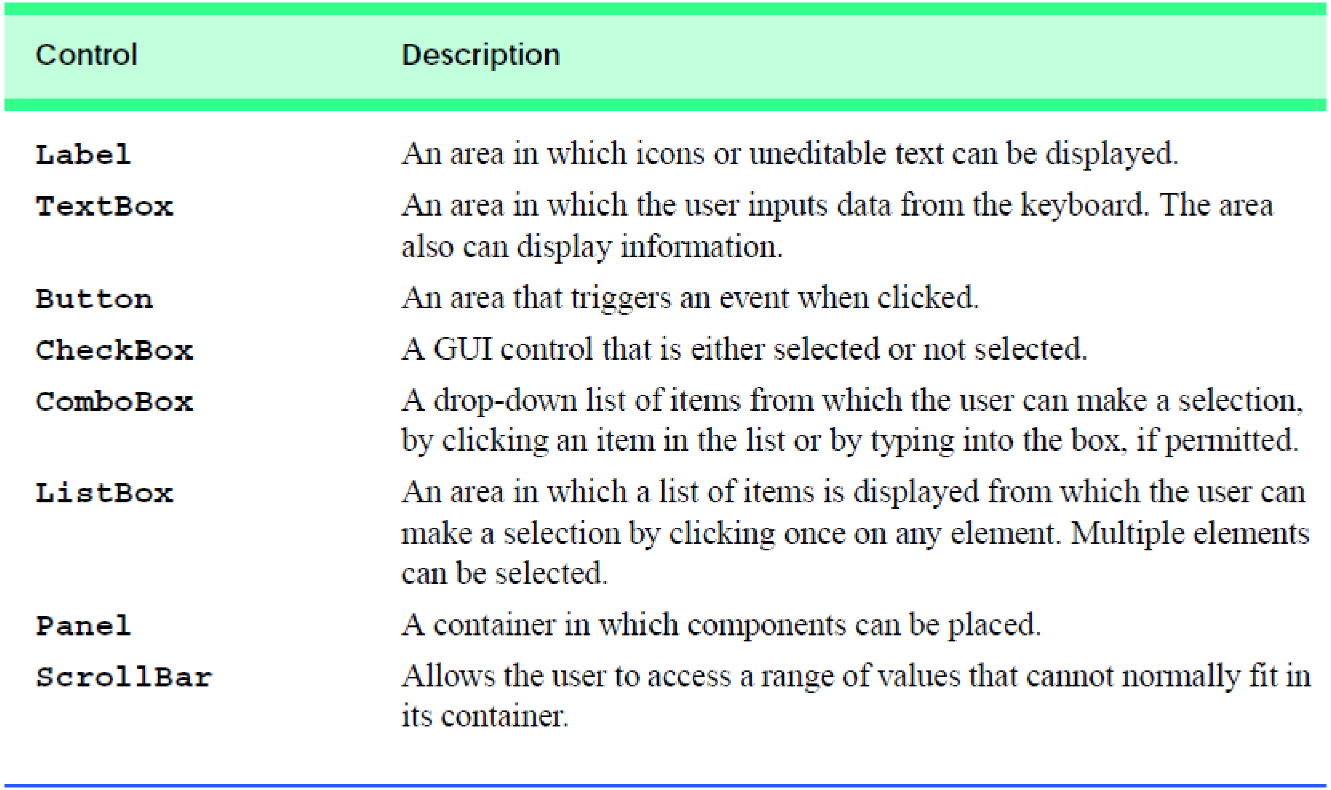
Ejemplo 6.1-OOP2

GUI PROGRAMMING USING WINDOWS FORMS

As forms are the base unit of your application, it is essential that you give some thought to their function and design. A form is ultimately a blank slate that you, as a developer, enhance with controls to create a user interface and with code to manipulate data. To that end, Visual Studio provides you with an integrated development environment (IDE) to aid in writing code, as well as a rich control set written with the .NET Framework. By complementing the functionality of these controls with your code, you can easily and quickly develop the solutions you need.

A GUI build using windows forms is event driven, that means that it wait until the user do something to trigger actions.

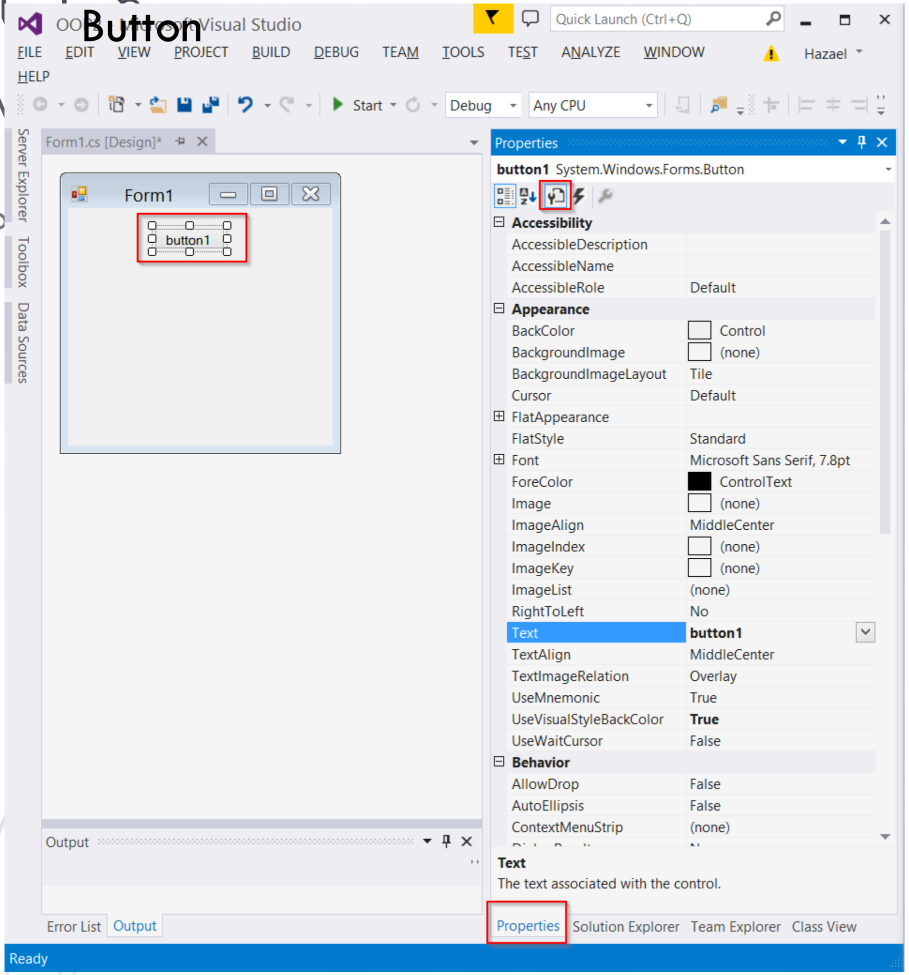
https://msdn.microsoft.com/en-us/library/dd30h2yb(v=vs.110).aspx



Control	Description
`Label`	An area in which icons or uneditable text can be displayed.
`TextBox`	An area in which the user inputs data from the keyboard. The area also can display information.
`Button`	An area that triggers an event when clicked.
`CheckBox`	A GUI control that is either selected or not selected.
`ComboBox`	A drop-down list of items from which the user can make a selection, by clicking an item in the list or by typing into the box, if permitted.
`ListBox`	An area in which a list of items is displayed from which the user can make a selection by clicking once on any element. Multiple elements can be selected.
`Panel`	A container in which components can be placed.
`ScrollBar`	Allows the user to access a range of values that cannot normally fit in its container.

Basic controls

Properties of a Windows Forms Button



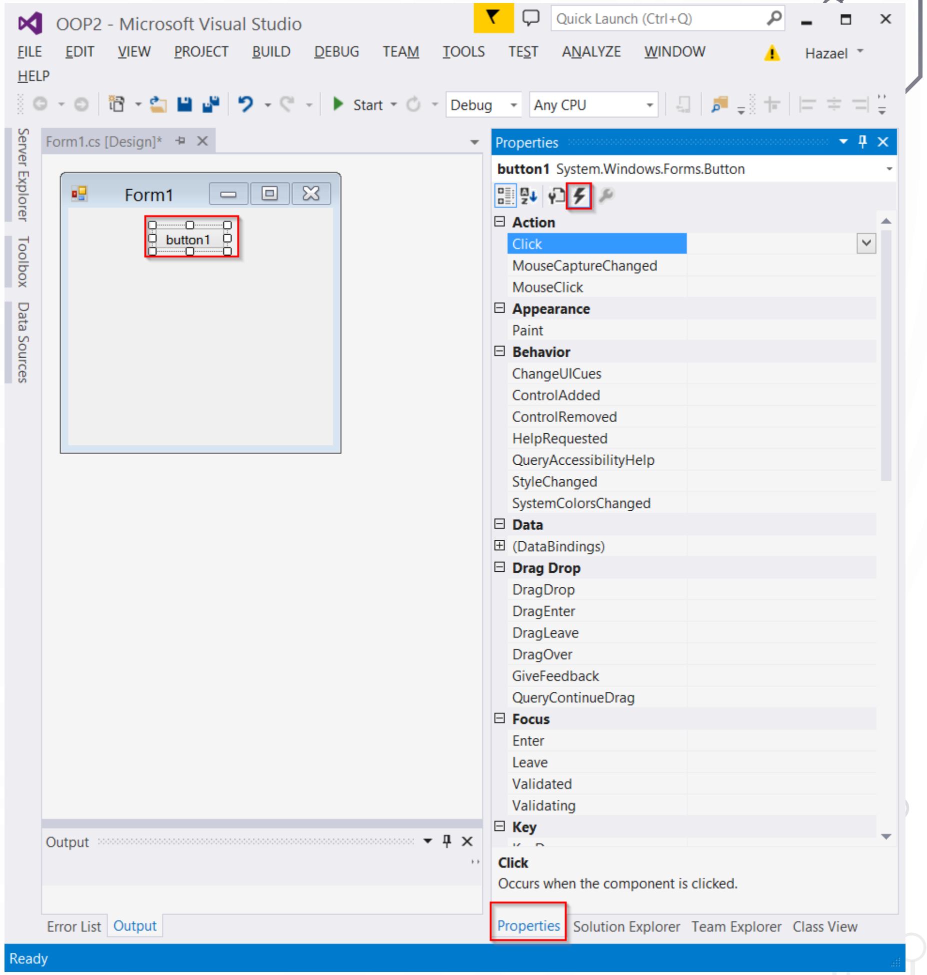
Properties

button1 System.Windows.Forms.Button

Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif, 7.8pt
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	**button1**
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackColor	**True**
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	(none)

Text
The text associated with the control.

Events of a Windows Forms Button



Properties

button1 System.Windows.Forms.Button

- **Action**
  - Click
  - MouseCaptureChanged
  - MouseClick
- **Appearance**
  - Paint
- **Behavior**
  - ChangeUICues
  - ControlAdded
  - ControlRemoved
  - HelpRequested
  - QueryAccessibilityHelp
  - StyleChanged
  - SystemColorsChanged
- **Data**
  - (DataBindings)
- **Drag Drop**
  - DragDrop
  - DragEnter
  - DragLeave
  - DragOver
  - GiveFeedback
  - QueryContinueDrag
- **Focus**
  - Enter
  - Leave
  - Validated
  - Validating
- **Key**

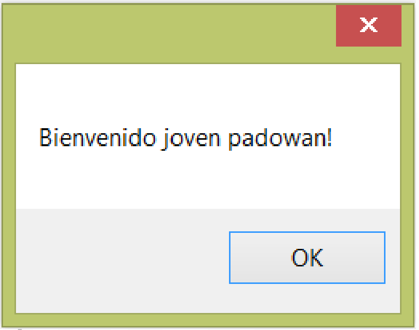
Click
Occurs when the component is clicked.

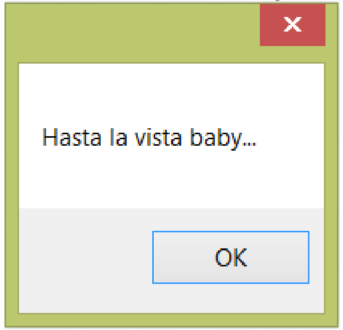
Ejemplo

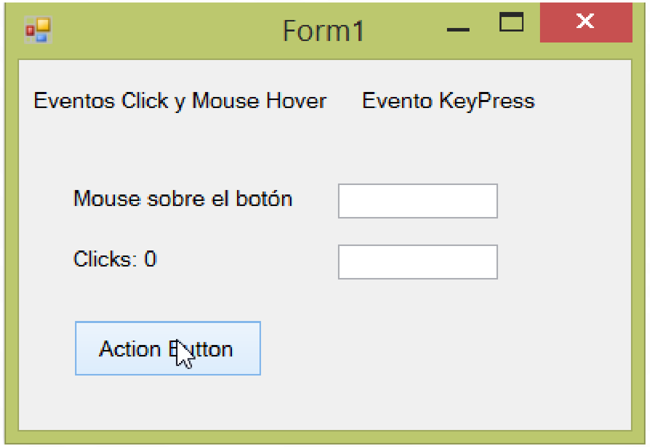
Mouse Events: Click, MouseHover

Keyboard Events: KeyPress

Forms Events: Load, FormClosing

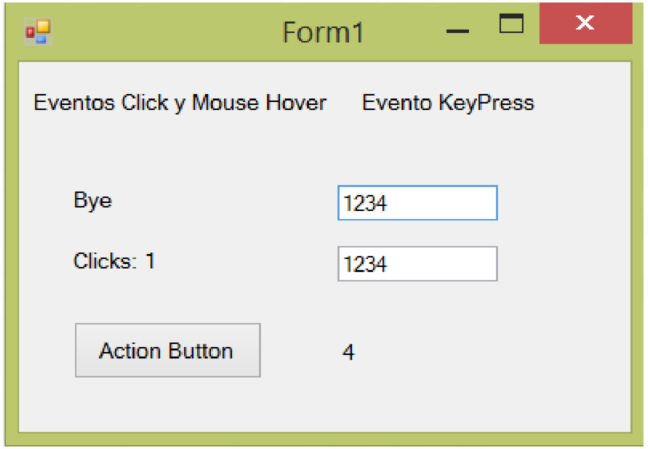

Bienvenido joven padowan!


Hasta la vista baby...



Mouse sobre el botón

Clicks: 0



Bye 1234

Clicks: 1 1234

4

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace OOP2
{
    public partial class Form1 : Form
    {
        int numberClicks = 0;

        public Form1()
        {
            InitializeComponent();
        }

        #region EVENTOS MOUSE
        private void buttonClick_Click(object sender, EventArgs e)
        {
            this.labelClick.Text = "Clicks: " + numberClicks;
            numberClicks++;
        }

        private void buttonClick_MouseHover(object sender, EventArgs e)
        {
            this.labelMouse.Text = "Mouse sobre el botón";
        }

        private void buttonClick_MouseLeave(object sender, EventArgs e)
        {
            this.labelMouse.Text = "Bye";
        }

        private void labelMouse_Click(object sender, EventArgs e)
        {
            MessageBox.Show("click!!");
        }

        private void labelClick_Click(object sender, EventArgs e)
        {
            MessageBox.Show("cluck!!");
        }

        #endregion
```

Mouse Events.
The method *buttonClick_Click* is executed when the user presses the button, the same as the methods *labelMouse_Click* and *labelClick_Click*.

The method *buttonClick_MouseHover* is being executed when the mouse pointer is over the button and the method *buttonClick_MouseLeave* is executed when the mouse pinter leaves he limits of the button.

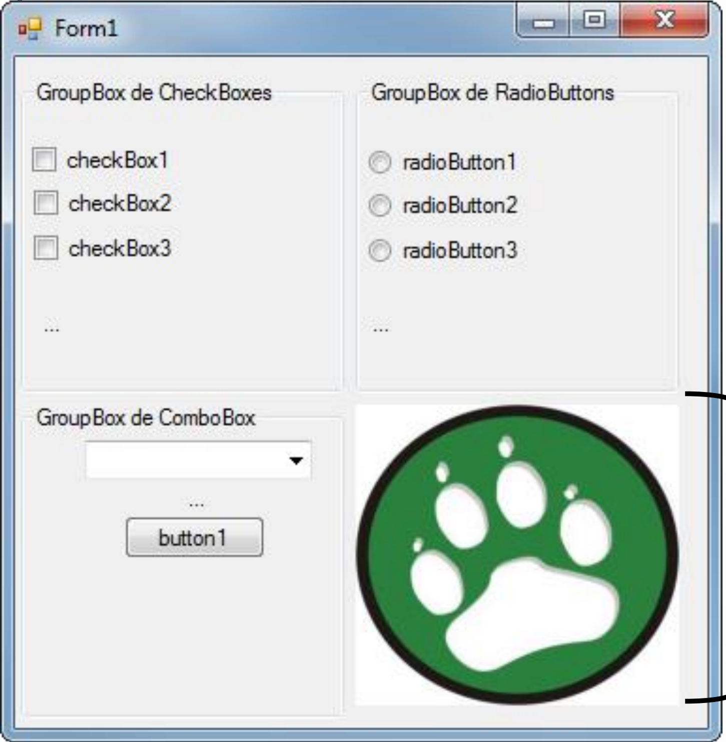
Form Events. The methods *Form1_Load* and *Form1_FormClosing* are executed when the Form is created or destroyed (when the window is just opened or closed)

Keyboard Events. The method *textBoxKeyPress_KeyPress* is being executed each time the TextBox has the focus and the user press a key on the keyboard.

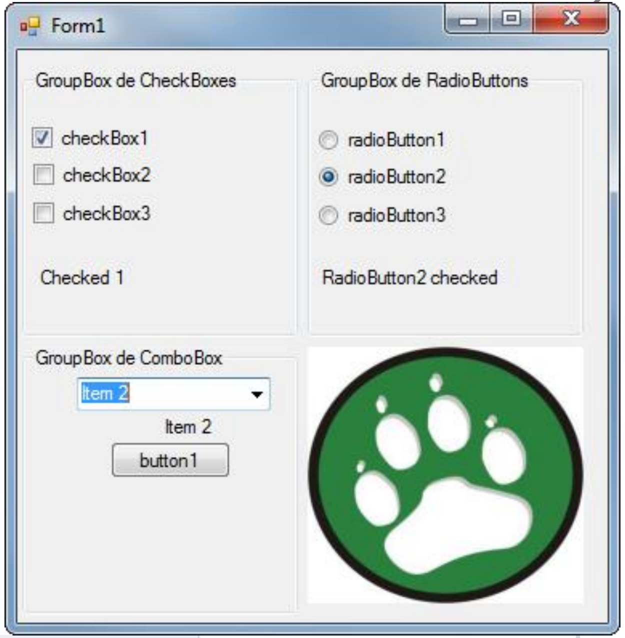
```
        #region EVENTOS DE FORMULARIO
        private void Form1_Load(object sender, EventArgs e)
        {
            MessageBox.Show("Bienvenido joven padowan!");
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            MessageBox.Show("Hasta la vista baby...");
        }
        #endregion

        #region EVENTOS DE TECLADO
        private void textBoxKeyPress_KeyPress(object sender, KeyPressEventArgs e)
        {
            this.textBoxKeyPress2.Text += e.KeyChar.ToString();
            this.labelKeyPress.Text = e.KeyChar.ToString();
        }
        #endregion
    }
}
```

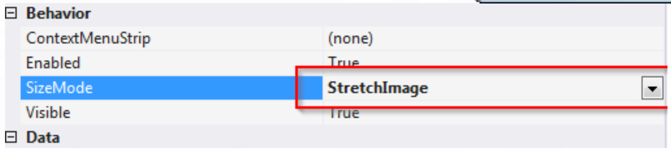
Example

PictureBox, CheckBox and RadioButton.

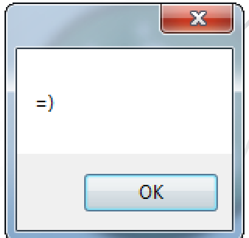


Source code in 6.3-OOP2





Behavior	
ContextMenuStrip	(none)
Enabled	True
SizeMode	**StretchImage**
Visible	True
Data	



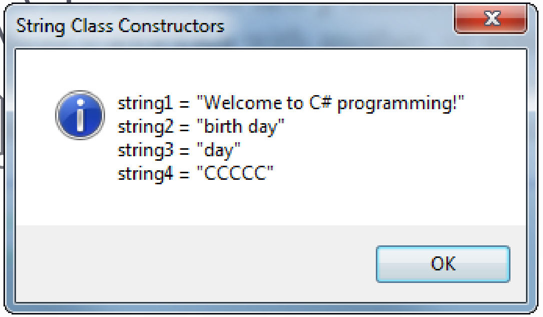
STRINGS

A **string** is an object of type **String** whose value is text. Internally, the text is stored as a sequential read-only collection of Char objects. There is no null-terminating character at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0'). The Length property of a string represents the number of Char objects it contains, not the number of Unicode characters.

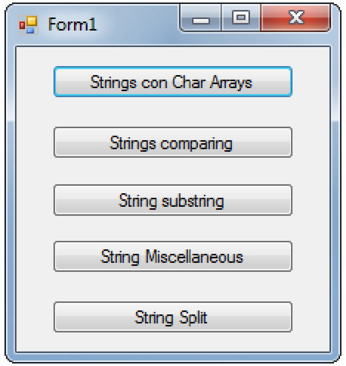
In C#, the ***string keyword is an alias for String***. Therefore, String and string are equivalent, and you can use whichever naming convention you prefer. The String class provides many methods for safely creating, manipulating, and comparing strings.

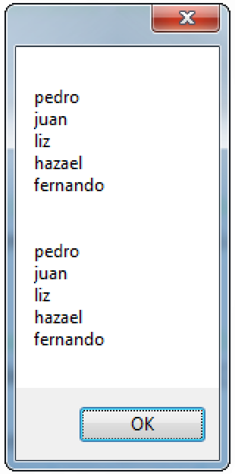
```
string color = "blue";
```

Example

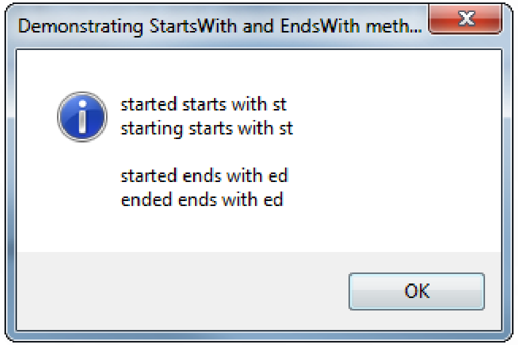

String Class Constructors

```
string1 = "Welcome to C# programming!"
string2 = "birth day"
string3 = "day"
string4 = "CCCCC"
```



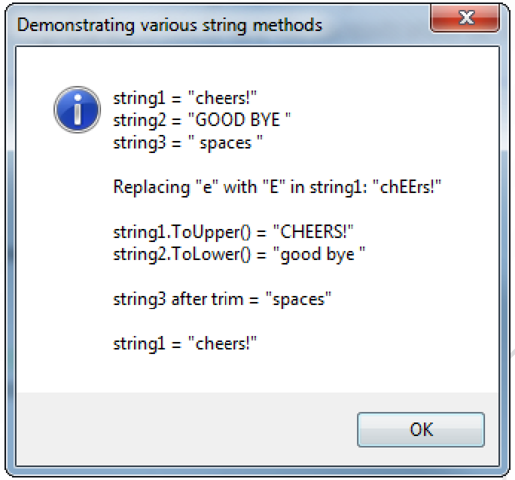


```
pedro
juan
liz
hazael
fernando


pedro
juan
liz
hazael
fernando
```

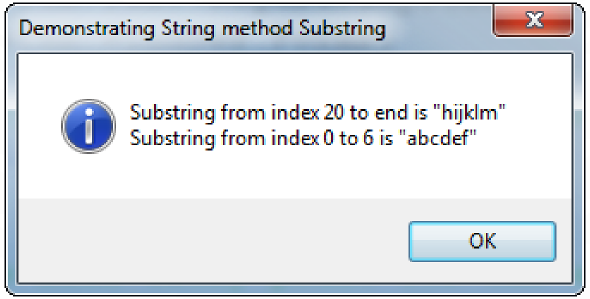

Demonstrating StartsWith and EndsWith meth...

```
started starts with st
starting starts with st

started ends with ed
ended ends with ed
```

Ver código fuente en 6.4-OOP2

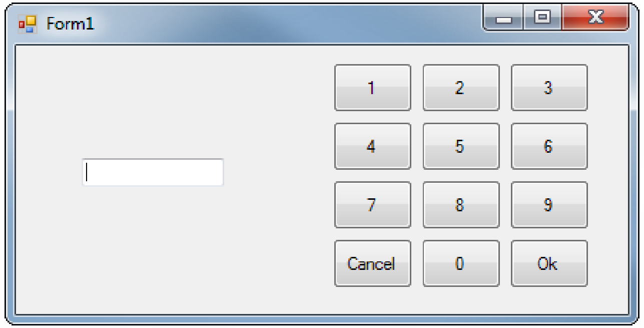

Demonstrating various string methods

```
string1 = "cheers!"
string2 = "GOOD BYE "
string3 = " spaces "

Replacing "e" with "E" in string1: "chEErs!"

string1.ToUpper() = "CHEERS!"
string2.ToLower() = "good bye "

string3 after trim = "spaces"

string1 = "cheers!"
```


Demonstrating String method Substring

```
Substring from index 20 to end is "hijklm"
Substring from index 0 to 6 is "abcdef"
```

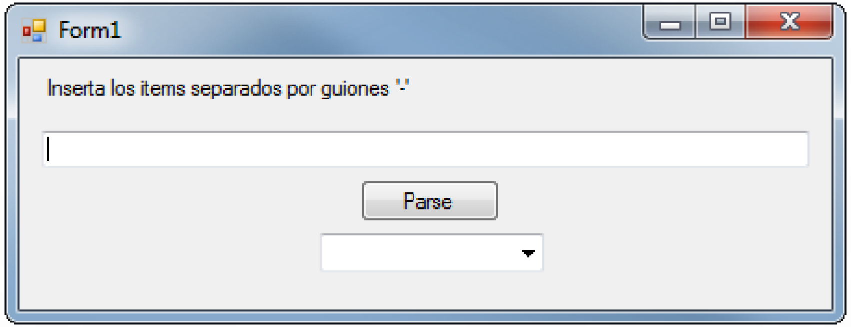
HOMEWORK 1



Build a small desktop application that simulates an ATM machine interface as you can see in the image above.

- The user should not be able to type anything in the TextBox, instead, he should click the button matrix in the right side of the screen.
- When the user clicks a "numeric button" only dots/asterisks (*) should appear in the TextBox control (simulating a password/NIP).
- The user should be able to enter up to 4 digits.
- When the user press the "ok button" the program will compare the number against a "hardcoded" value (example "1234") and message will pop up saying if the password is correct or not.
- Build the screen the way you like.

HOMEWORK 2



Create an application with a GUI like the above image.

The purpose of this application is to add items to the ComboBox using a text string typed in the TextBox.

- The user will type a "dash separated" text in the TextBox, like ***"hola-fime como-estas".***
- When the user clicks the button the application must **parse** that text, separate that text into items, each item is separated by a **dash**, then those items should be added to the ComboBox, ***{hola, fime, como, estas}***.
- The application must be robust enough to discern if the TextBox is empty or not and never end abruptly or have execution errors.