واجهات المستخدم المرئية
GRAPHICAL USER INTERFACE

ENG.HAMDO AL-HAMDO.

1

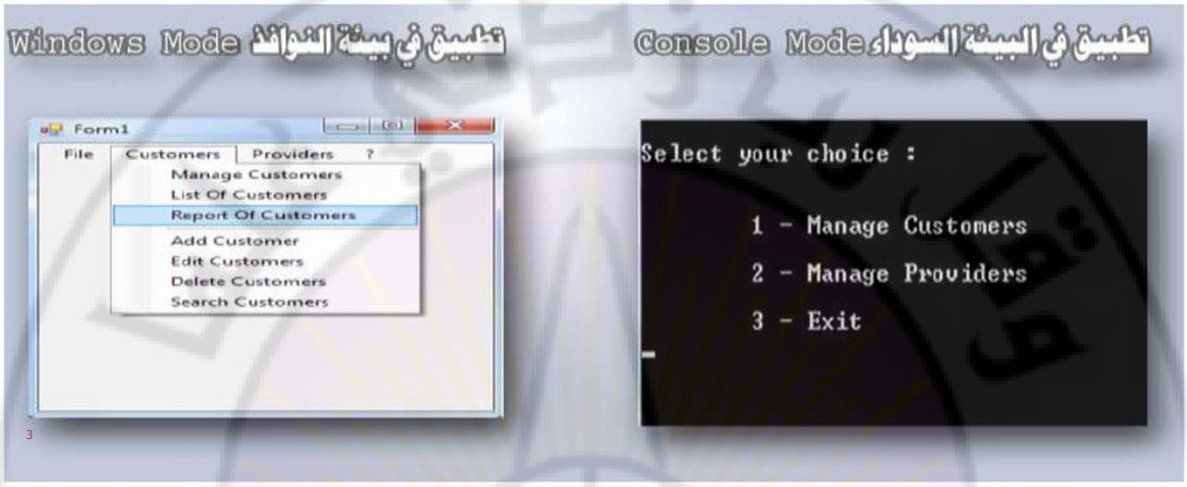
البرمجة غرضية التوجه OOP

واجهات المستخدم المرئية
Graphical User Interface With Windows Forms

- تسمح واجهة المستخدم المرئية GUI للمستخدم التفاعل بشكل رسومي مع البرامج ، ويمنح البرنامج مظهراً واحساساً مميزاً .
- يطلق على واجهة المستخدم التي تستخدم أسلوب الادخال والإخراج المحرفي اسم CHUI أي Character user Interface .
- **أما في تطبيقات windows فإننا نقوم بإنشاء واجهة مستخدم مرئية التي تقدم طريقة إدخال وإخراج المعطيات بواسطة واجهات مرئية .**
- نستخدم تطبيقات Windows Form في بيئة Visual Stduio.NET بدلاً من تطبيقات Console .

2

مدخل إلى برمجة الواجهات



3

مثال لبرنامج ال CONSOLE

سنقوم بتغيير لون خلفية الخط من الأسود إلى اللون الأصفر مثلا، وكذلك لون الخط من الأبيض إلى الأحمر.

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.BackgroundColor = ConsoleColor.Yellow;
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Write("Hi Brothers ! ");
            Console.ReadKey();
        }
    }
}
```

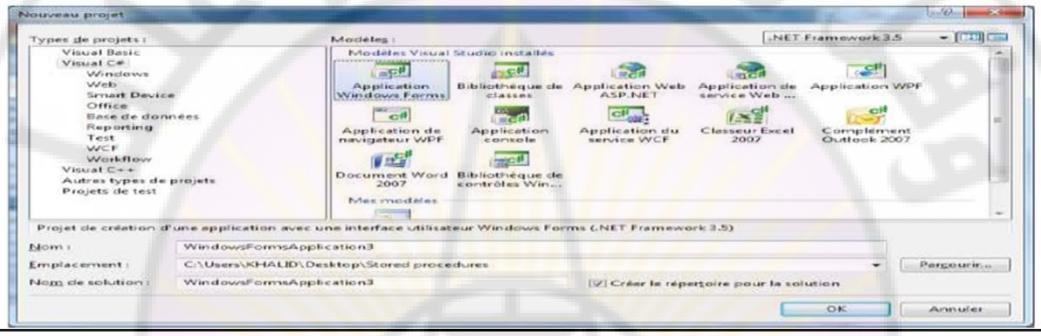
النتيجة كما يلي :

```
Hi Brothers !
```

4

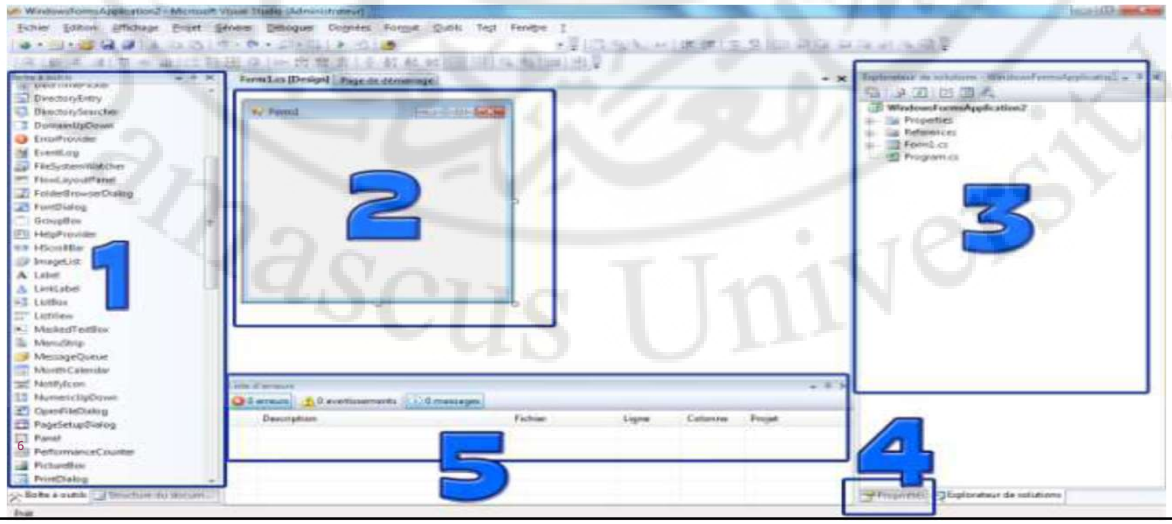
إنشاء مشروع جديد

للولوج إلى نافذة التصميم، سنقوم أولا بإنشاء مشروع جديد وذلك عن طريق فتح برنامج الفيجوال استوديو، ثم الذهاب إلى القائمة File واختيار New project، بعد ذلك ستظهر لنا هذه النافذة، أو بكل بساطة الضغط على Ctrl+Shift+N.



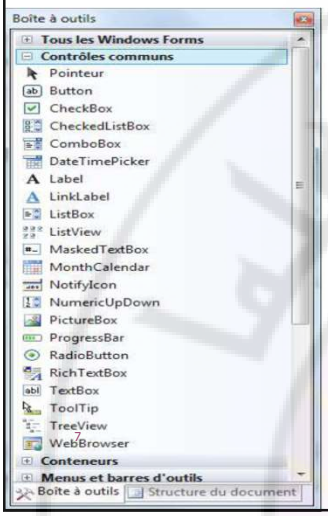
5

إنشاء مشروع جديد



6

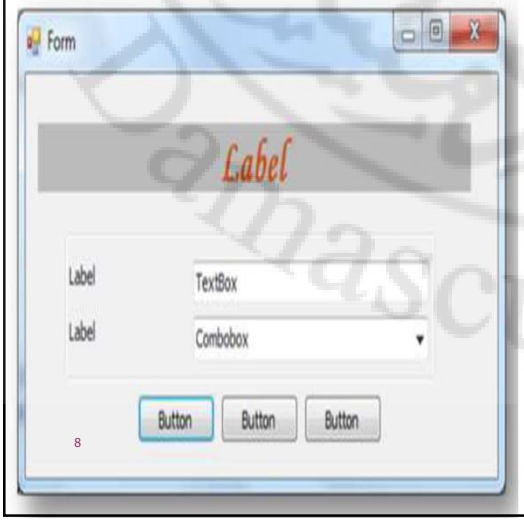
صندوق الأدوات



سنورد شرحا بسيطا لكل جزء من بيئة التصميم حسب الأرقام الواردة في الصورة:

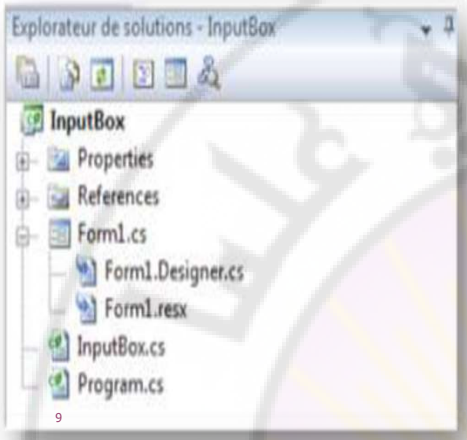
1. وتسمى هذه النافذه بعلبة الأدوات ToolBox، وهي تضم كل الأدوات التي قدم يحتاجها برنامجك (أزرار، قوائم، ...) . إن لم تكن ظاهرة عندك فاذهب إلى القائمة View ثم

FORMS



2. هذا هو الفورم Form الذي سنضع عليه الأدوات اللازمة لبناء المشروع ويمكنك إضافة العديد من الفورمات إلى مشروعك كما سنرى فيما بعد إن شاء الله، ويمكنك تغيير مقاسه عبر مسك الزوايا وجذبها أو من خلال نافذة الخصائص رقم 4، وهذه صورة لفورم وعليه بعض الأدوات:

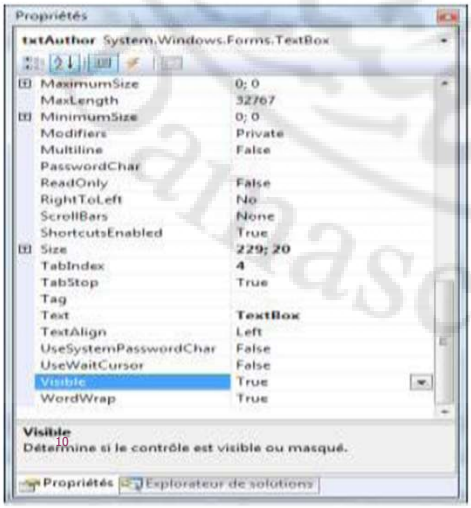
متصفح المشروع



9

3. ويسمى هذا الجزء متصفح المشروع Solution Explorer، وسمي كذلك لأنه يعرض كل الملفات التي يضمها المشروع حسب تبويبات خاصة بكل نوع، ويمكن إظهاره في حالة غيابه عن طريق الذهاب إلى القائمة View واختيار Solutions Explorer أو الإكتفاء بالضغط على الاختصار Ctrl+Alt+L

نافذة الخصائص

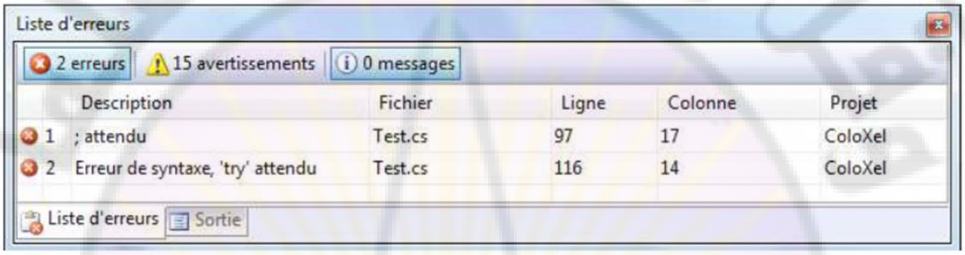


10

4. نافذة الخصائص Properties : وتحتوي على خصائص الأداة التي نحددها، ومن خلال هذه النافذة يمكننا تغيير اللون والخلفية و الخط وباقي الخصائص، لإظهارها في حالة اختفائها قم بتحديد الأداة المرغوب تغيير خصائصها والضغط عليها بيمين الماوس واختيار Properties أو يكفيك الضغط على زر لوحة المفاتيح F4

ERROR LIST

5. قائمة الأخطاء Error List وتعرض هذه النافذة الأخطاء المرتكبة قبل بدء عملية ترجمة الشفرة Compilation، من خلالها يمكنك معرفة مكان الخطأ ليتأتى لك تصحيحه.



11

ابسط برنامج WIN

- أبسط برنامج Windows يمكننا إنجازه هو عرض نموذج فارغ كما يلي :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication21
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }// end form_load
    } // end class
} // end namespace
```

12

ابسط برنامج WIN

- على الرغم من أن هذه النافذة لا تقدم شيئاً مفيداً ولكنها فعلياً تقدم الكثير .
- يمكن سحب النافذة إلى أي مكان من سطح المكتب ، ويمكن تصغيرها وتكبيرها من مربعات التحكم الموجودة في الزاوية اليمنى ، ويمكن إغلاقها أيضاً .
- هناك أيقونة في الزاوية اليسارية العليا يؤدي نقرها إلى عرض قائمة النظام ، ويمكن النقر على حافة النافذة ونسحبها لتغيير حجمها .
- كل هذه الوظائف تم بناءها مسبقاً في النافذة Form والحصول عليها مجاناً .
- تتألف النافذة الأساسية من شريط العنوان title bar والمحيط border والمنطقة التابعة للمستخدم client area من أجل رسم وعرض واجهة المستخدم الخاصة به .
- لإنشاء نافذة يتطلب التعامل مع الصف Form أو مع صف يرث الصف Form معرف في فضاء الحالة System.Windows.Forms ، ويجب تضمين فضاء الحالة في أعلى البرنامج .

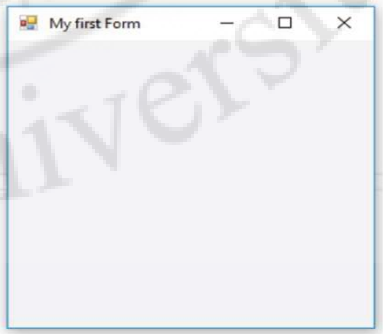
13

مثال بسيط

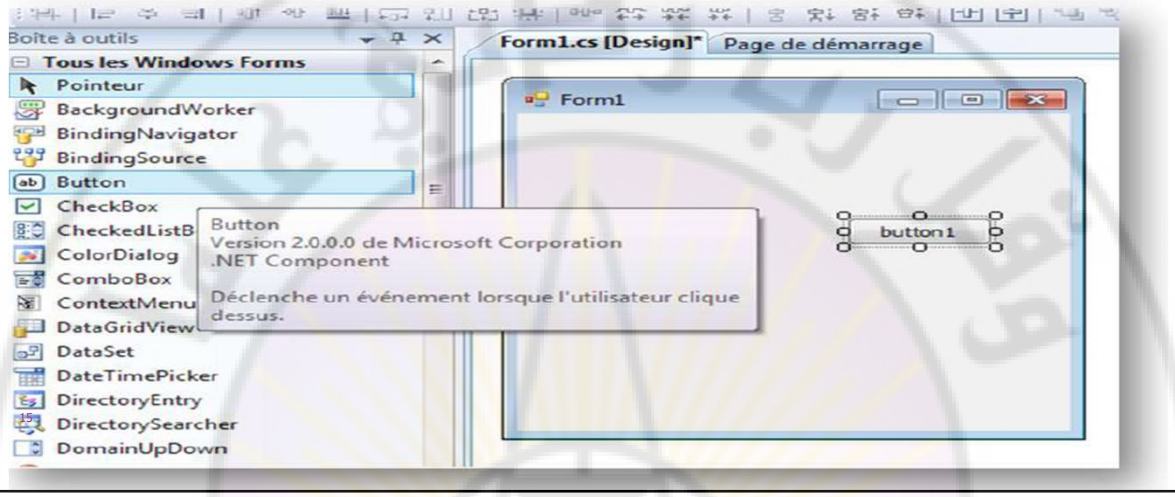
مثال (1)

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            Text = " My fierst Form ";
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }// end main
    }// end class
}// end nameclace
```

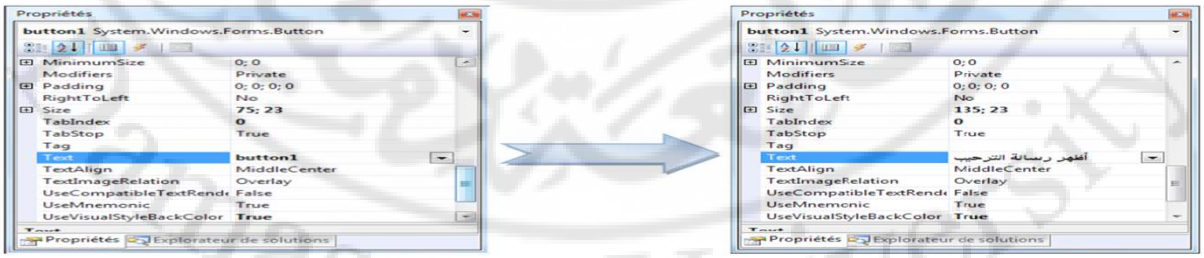
14



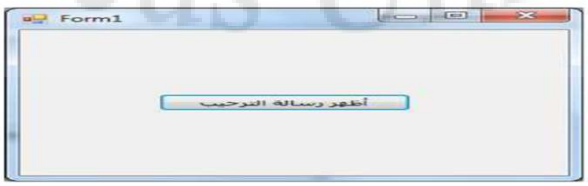
إضافة عناصر للفورم



إضافة عناصر للفورم



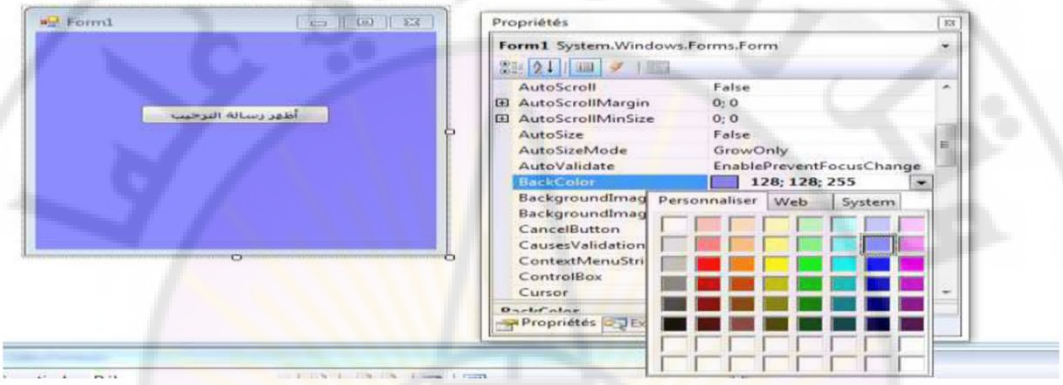
سيصبح الزر كما يلي :



16

إضافة عناصر للفورم

بنفس الطريقة سنغير لون الفورم، وذلك بتحديده اولا ثم الذهاب إلى الخصيصة BackColor ونختار اللون الذي نشاء.



17

مجالات الاسماء

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

وهي تلك الأسماء التي تكون مسبوقة بالموجهة using و هي مجالات تضم العديد من الفئات والأنواع، كما يمكن لمجال واحد أن يضم مجموعة من مجالات الأسماء كما هو الحال مع المجال System الذي نجد به فئات كثيرة مثل console و convert وكذلك يضم مجالات أسماء فرعية مثل IO و Collections وأول سطر يكون في البرنامج هو سطر التأشير إلى مجالات الأسماء ويكون باستعمال الأمر الموجه using متبوعا باسم مجال الأسماء ، واستعمال هذه الطريقة يوفر علينا أن نقوم كل مرة بكتابة مجال الأسم قبل فئاته، فلولا هذه الإمكانية لكتبنا System.Console.WriteLine() بدل Console.WriteLine().

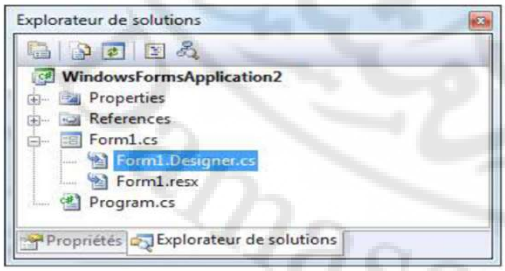
18

ملاحظة

عند الاشتغال على تطبيقات الويندوز لابد من جلب مجال الأسماء System.Windows.Forms، لأنه يضم كل الفئات اللازمة لهذا النوع من المشاريع، ولعل أهم فئة هي System.Windows.Forms.Control لأنها الفئة الأم التي ترث منها معظم الأدوات Controls

19

مفاهيم




```
namespace WindowsFormsApplication2
{
partial class Form1
namespace WindowsFormsApplication2
{
public partial class Form1 : Form
```

```
public partial class Form1 : Form
```

فيه نجد مفهومين أحدهما رأيناه في الجزء الأول وهو الوراثة Inheritance، حيث نقوم بوراثة Form1 مشتق من الفئة Form الرئيسية، أما المفهوم الثاني وهو الفئات الجزئية partial classes فإنه يسمح لنا بتقسيم الفئة الواحدة على العديد من الملفات حيث نستعمل الكلمة المحجوزة partial أمام اسم الكلاس إذا احتجنا إلى الاشتغال عليه في ملف آخر.

10

الحدث LOAD

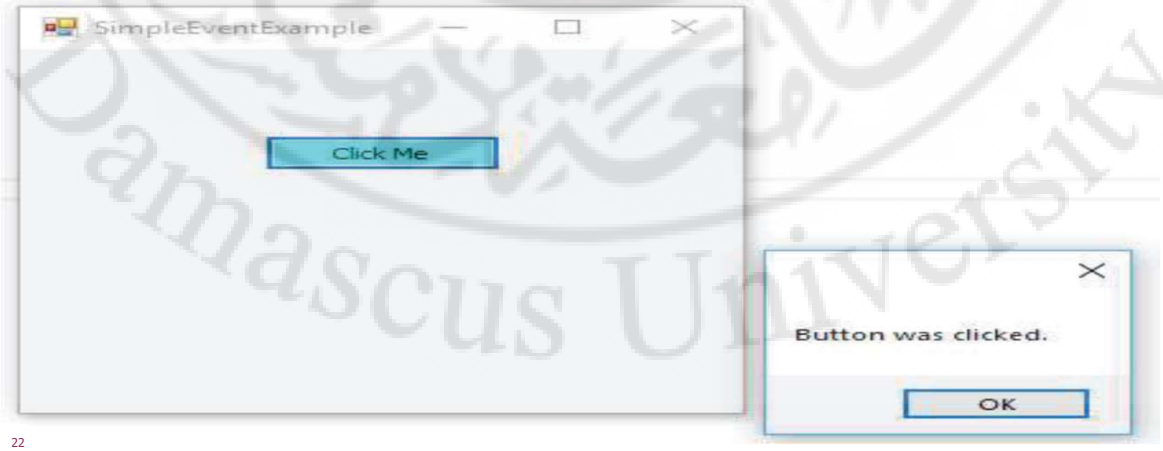
مثال (2)

ماذا يعني Form1_Load() :
تنفيذ الكود الموجود ضمن القوسين عندما الـ Form1 يحصل لها تحميل Load

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{// Form that shows a simple event handler
    public partial class Form1 : Form
    {  // default constructor
        public Form1()
        {
            InitializeComponent( );
        }// end constructor
        // handles click event of Button clickButton
        private void Form1_Load(object sender, EventArgs e)
        {
            Text = "SimpleEventExample ";
        }
        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Button was clicked.");
        }
    }//end form
}// end name class
```

21

مثال 2



22

خصائص الفورم

اسم الخصيصة	دورها
Text	لتغيير النص الظاهر على الأداة
BackColor	لتغيير لون خلفية الأداة
BackgroundImage	لتغيير صورة خلفية الفورم Form
BackgroundImageLayout	كيفية ظهور صورة خلفية الفورم، وتأخذ القيم التالية: None: ستظهر الخلفية في أعلى يسار الفورم. Tile: ستظهر الصورة مكررة طولا وعرضا. Stretch: ستظهر الصورة ممدة على كل الفورم. Zoom: ستظهر الصورة في أقصى مقاسها ممركزة في وسط الفورم.
ForeColor	لتغيير لون النص المكتوب على الأداة.

23

خصائص الفورم

Font	لتغيير الخط المكتوب على الأداة.
Name	لتغيير اسم الأداة، وهذا هو أهم خصيصة للأداة لأننا سنتعامل معه
Width	تغيير عرض الأداة.
Height	تغيير علو الأداة (طولها عموديا).
Visible	هذه الخصيصة تقبل إما صح True أو خطأ False، وهي تتحكم في ظهور الأداة أثناء تنفيذ البرنامج، بمعنى إذا كانت قيمة هذه الخصيصة False فلن تظهر الأداة عند التنفيذ.
Enabled	لإلغاء اشتغال الأداة أثناء التنفيذ إذا كانت قيمتها False، ولتفعيل الأداة نجعل قيمتها True
Dock	لتثبيت الأداة في إحدى زوايا الفورم.
Anchor	لجعل مقاس الأداة مرتبطا بمقاس الفورم، وهذا مهم حينما يكون

24

خصائص الفورم

Parent	لتحديد الفورم الأب للفورم الحالي (سنتعرف على هذا المفهوم فيما سيأتي إن شاء الله)
Cursor	لتغيير شكل مؤشر الماوس عند مروره فوق الأداة.
FormBorderStyle	لتحديد شكل إطار الفورم، من هنا يمكن إختيار القيمة التي تجعل الفور غير قابل لتغيير مقاسه وهي FixedSingle، القيمة FixedDialog تؤدي نفس الدور أيضا مع اختلاف شكل الإطار.
Icon	لتغيير أيقونة الفورم، على شرط أن يكون امتداد الأيقونة ico
Opacity	لتغيير شفافية الفورم، القيمة 0 تجعل الفورم مخفيا، والقيمة 1 تظهره بشكل عادي، وكلما غيرت القيمة من 0 إلى 1 تغيرت الشفافية.
MaximizeBox	لإخفاء زر التكبير الموجود على الشريط العلوي للفورم، وأيضا

25

خصائص الفورم

MinimizeBox	لإخفاء زر إنزال الفورم إلى شريط المهام، وأيضا لإظهاره.
WindowState	لتغيير حجم الفورم ويأخذ القيم التالية: Normal: القيمة الافتراضية حيث يظهر الفورم في المقاس الذي صممته عليه. Maximized: لإظهار الفورم بشكل مكبر يشغل شاشة الحاسوب. Minimized: لإنزال الفورم إلى شريط المهام.
Location	موضع الفورم حسب الإحداثيات X و Y
StartPosition	الموضع الذي ستظهر فيه النافذه عند التنفيذ، ويأخذ القيم التالية: CenterParent: ليظهر الفورم في وسط الفورم الأب. CenterScreen : ليظهر الفورم في وسط

26

خصائص الفورم

الشاشة.

Manual: لجعل الموضع مرتبطا بالخصيصة Location.

WindowsDefaultBoundS: لجعل المقاس والموضع مرتبطان بنظام التشغيل.

WindowsDefaultLocation: لجعل الموضع لوحده مرتبطا بنظام التشغيل، أما المقاس فيبقى كما حددته عند التصميم.

27

الصف MSGBOX

- لعرض أي نص ضمن صندوق حوار ، على سبيل المثال الرسالة (message dialog box) ، سنحتاج إلى استخدام التابع Show وظيفته إظهار رسالة ما للمستخدم ، هذا التابع موجود ضمن الصف MessageBox هو أحد صفوف البرمجة المرئية الموجود ضمن C#.
- هو تابع static، ويجب استدعاء هذا التابع باستخدام اسم الصف متبوعاً بنقطة (.) ثم التابع مع الوسائط (البارامترات) ، ويُستخدم لعرض أي نص في صندوق الحوار. حيث يأخذ هذا التابع أربعة(حالياً) وسطاء وهي محملة بشكل زائد .

❖**الوسيط الأول :** يمثل الرسالة التي ستظهر للمستخدم (Welcome to C#) ، ومن الممكن أن تكون سلسة محرفية تمثل النص الواجب إظهاره .

28

الصف MSGBOX

- **الوسيط الثاني:** يمثل الرسالة التي ستظهر على شريط العنوان لصندوق حوار الرسالة ، مثلاً (My First Example) .
- **الوسيط الثالث:** فهو يمثل أحد أنواع أزرار حوار الرسالة المبينة في الجدول (4) .
- **الوسيط الرابع:** فهو يمثل أحد أنواع أيقونات حوار الرسالة المبينة في الجدول (5) .

❑ أما الوسيطان الأخيران فهما ثوابت يؤثران على شكل مربع الرسالة ، حيث ان قيمة MessageBoxButton.OK تخبر MessageBox ان يقوم بإظهار الزر OK فقط ، بينما قيمة MessageBoxIcon.Information فتخبره ان يظهر أيقونة تبين أنها معلومات مفيدة .

29

ازرار صندوق الحوار

MessageBox Buttons	Description
`MessageBoxButton.OK`	Specifies that the dialog should include an **OK** button.
`MessageBoxButton.OKCancel`	Specifies that the dialog should include **OK** and **Cancel** buttons. Warns the user about some condition and allows the user to either continue or cancel an operation.
`MessageBoxButton.YesNo`	Specifies that the dialog should contain **Yes** and **No** buttons. Used to ask the user a question.
`MessageBoxButton.YesNoCancel`	Specifies that the dialog should contain **Yes**, **No** and **Cancel** buttons. Typically used to ask the user a question but still allows the user to cancel the operation.
`MessageBoxButton.RetryCancel`	Specifies that the dialog should contain **Retry** and **Cancel** buttons. Typically used to inform a user about a failed operation and allow the user to retry or cancel the operation.
`MessageBoxButton.AbortRetryIgnore`	Specifies that the dialog should contain **Abort**, **Retry** and **Ignore** buttons. Typically used to inform the user that one of a series of operations has failed and allow the user to abort the series of operations, retry the failed operation or ignore the failed operation and continue.

30

ايقونات صندوق الحوار

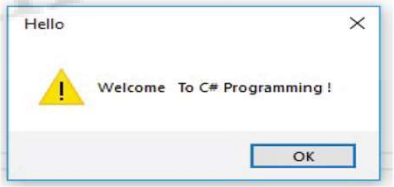
MessageBox Icons	Icon	Description
MessageBoxIcon.Exclamation		Specifies an exclamation point icon. Typically used to caution the user against potential problems.
MessageBoxIcon.Information		Specifies that the dialog contains an informational message for the user.
MessageBoxIcon.Question		Specifies a question mark icon. Typically used in dialogs that ask the user a question.
MessageBoxIcon.Error		Specifies a dialog with an × in a red circle. Alerts user of errors or important messages.

Fig. 5.6 Icons for message dialogs.

مثال 3

مثال (3) المثال التالي يوضح خصائص التابع Show من الصف MessageBox.

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication3
{
    public partial class SimpleEventExampleForm1 : Form
    {
        public SimpleEventExampleForm1()
        {
            InitializeComponent();
        }
        private void SimpleEventExampleForm1_Load(object sender, EventArgs e)
        {
            string str = " To C# Programming ! ";
           MessageBox.Show("Welcome  " + str ," Hello",
           MessageBoxButtons.OK,
           MessageBoxIcon. Exclamation);
        }// end form
    }//end class
}// end namespace
```



مثال 4: حساب مجموع الاعداد الزوجية من 2 لل100

مثال (4)

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication3
{
 public partial class SimpleEventExampleForm1 : Form
  {
   public SimpleEventExampleForm1()
   {
     InitializeComponent();
    }
   private void SimpleEventExampleForm1_Load(object sender, EventArgs e)
    {
          int sum = 0;
            for (int number = 2; number <= 100; number += 2)
                sum += number;
            MessageBox.Show("The sum is " + sum,
            "Sum Even Integers from 2 to 100",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
   }// end form
  }// end class
}// end namespace
```

33

النهاية ....

34

الأزرار و اللافتات و مربعات النص
BUTTONS , LABLE AND TEXT BOX

ENG.HAMDO AL-HAMDO.

الصف MSGBOX

بعد إضافة الوسيط الثالث الذي هو MessageBoxButtons الذي يدل على أنواع الأزرار التي يمكن أن تظهر على النافذة نريد إظهار اسم الزر الذي تم الضغط عليه وذلك كما يلي :

- ننشئ متحول من نوع DialogResult والذي يحتوي على القيمة المعادة من التابع MessageBox.Show().

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    DialogResult dr;
    dr = MessageBox.Show(" Hello ", " Title  ", MessageBoxButtons.AbortRetryIgnore);
    lbl.Text = dr.ToString();
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي :

- نجد ظهور اسم الزر الذي تم الضغط عليه على نافذة الخرج .

الصف MSGBOX

عند الضغط على الزر Retry

عند الضغط على الزر Abort

مثال

عند الضغط على الزر Ignore

عند الضغط على الزر Retry

عند الضغط على الزر Abort

مثال

نريد إظهار اسم الزر الذي تم الضغط عليه باللغة العربية وذلك كما يلي :

- نستخدم تعليمات if – else المتكررة كما هو موضح في البرنامج .

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    DialogResult dr;
    dr = MessageBox.Show(" Hello ", " Title  ", MessageBoxButtons.AbortRetryIgnore);
    lbl.Text = dr.ToString();
if (dr == DialogResult.Retry)
 {
   lbl.Text = "محاولة " ;
 }
else if (dr == DialogResult.Ignore)
  {
     lbl.Text = "تجاهل " ;
   }
else if (dr == DialogResult.Abort)
   {
      lbl.Text = "إلغاء " ;
   }
}// end  btn
```

الصف MSGBOX

5- عند إضافة الوسيط الخامس الذي هو MessageBoxDefaultButton يحدد الزر الافتراضي الفعال في الحالة الابتدائية عند تنفيذ البرنامج وجعل الزر Button3 هو الافتراضي الذي يظهر على النافذة.

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

MessageBox.Show(" Hello " , " Title  ", MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question  , MessageBoxDefaultButton.Button3 );
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي :

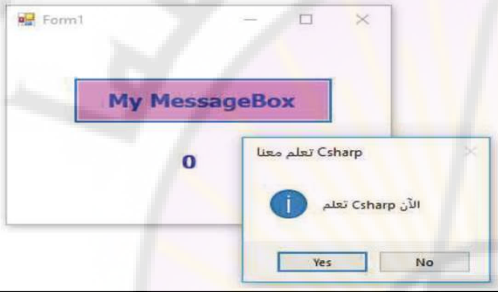
- نجد أن الزر Button3 هو الافتراضي على الشاشة .

الصف MSGBOX

نريد كتابة الوسيط الأول والثاني باللغة العربية كما يلي :

- الوسيط الأول " تعلم Csharp الآن " أما الوسيط الثاني " تعلم معنا Csharp ".

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
 MessageBox.Show("  تعلم Csharpالآن ,  " تعلم معنا Csharp    ", MessageBoxButtons.YesNo,
                 MessageBoxIcon.Information, MessageBoxDefaultButton.Button1);
}
```



عند تنفيذ البرنامج نحصل على الخرج التالي :

- نلاحظ أننا حصلنا على عبارات غير صحيحه .
- كيف يمكن تصحيحها وذلك بإضافة وسيط آخر من أجل الكتابة باللغة العربية .

الصف MSGBOX

6- عند إضافة الوسيط السادس الذي هو MessageBoxOptions
الذي يدل على بعض من Options منها تصحيح اللغة RtlReading

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
 MessageBox.Show("  تعلم Csharpالآن ,  " تعلم معنا Csharp    ", MessageBoxButtons.YesNo,
                 MessageBoxIcon.Information, MessageBoxDefaultButton.Button1 ,
                  MessageBoxOptions.RtlReading);
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي :

- نلاحظ أننا حصلنا على عبارات صحيحه .

BUTTONS

والأزرار
Buttons

- **الزر Button هو عنصر تحكم** يقوم المستخدم بالنقر فوقه لتشغيل إجراء محدد أو لتحديد خيار في أحد البرامج.
- كما سنرى ، **يمكن للبرنامج استخدام عدة أنواع من الأزرار** ، مثل مربعات الاختيار checkboxes وأزرار الاختيار radiobuttons .
- **جميع صفوف الأزرار تُشتق من الصف ButtonBase** namespace) System.Windows.Forms) ، التي تحدد ميزات الأزرار الشائعة.
- في هذا القسم ، **نناقش الصف Button**، الذي يمكّن للمستخدم عادةً من إصدار أمر إلى تطبيق ما.

الخصائص

Button properties and an event	Description
Common Properties	
`Text`	Specifies the text displayed on the `Button` face.
`FlatStyle`	Modifies a `Button`'s appearance—`Flat` (for the `Button` to display without a three-dimensional appearance), `Popup` (for the `Button` to appear flat until the user moves the mouse pointer over the `Button`), `Standard` (three-dimensional) and `System`, where the `Button`'s appearance is controlled by the operating system. The default value is `Standard`.
Common Event	
`Click`	Generated when the user clicks the `Button`. When you double click a `Button` in design view, an empty event handler for this event is created.

Fig. 14.19 | `Button` properties and an event.

الخصائص

- **btnOK** هو اسم الزر .
- **Click** هو أمر حدث عند النقر المضاعف عليه يتم إنشاء معالج حدث فارغ ، حتى يتم تنفيذ حدث ما نكتب العبارة التالية داخل تابع الحدث :

```
MessageBox.Show(" Hello  ") ;
```

- **استخدم الخاصة FlatStyle من أجل تعديل مظهر الزر منها :**
  - **Flat** مسطح (لعرض الزر بدون مظهر ثلاثي الأبعاد) .
  - **Popup** منبثق (يظهر الزر مسطحًا حتى يحرك المستخدم مؤشر الماوس فوق الزر) .
  - **Standard** القياسي (ثلاثي الأبعاد).
  - **System** النظام .

حيث يتم التحكم في مظهر الزر بواسطة نظام التشغيل ،حيث القيمة الافتراضية هي Standard .

مثال 2

مثال : انشاء زر وعند النقر عليه يقوم بحدث ما

- **المطلوب :** إنشاء زر ، ثم النقر المضاعف على الزر من أجل تغيير خصائصه كما يلي :
  - استخدم الخاصية Name من أجل تسمية التابع الذي يولد الزر (الاسم btnOk) .
  - استخدم الخاصية Text من أجل الكتابة على وجه الزر (عبارة OK) .
  - استخدام الخاصية BackColor من أجل تلوين الزر .
  - استخدام الخاصية ForeColor من أجل تغيير لون الخط للعبارة التي على وجه الزر .
- **التابع الذي يولد الزر له الشكل التالي :**

```
private void btnOK_Click(object sender, EventArgs e)
  {

   }
```

التنفيذ

البرنامج

```
using System;
using System.Windows.Forms;
namespace Button1 {
public partial class Form1 : Form
{
  public Form1()
  {
     InitializeComponent();
  }

private void Form1_Load(object sender, EventArgs e)
 {

 }
private void btnOK_Click(object sender, EventArgs e)
 {
    MessageBox.Show(" Hello C# Programming ");
  }
 }//end class
}// end namespace
```

مثال 3

- الزر الأول يقوم بتغيير لون خلفية الفورم إلى الأخضر.
- الزر الثاني يقوم بتغير الخط الافتراضي إلى Comic Sans MS
- الزر الثالث يقوم بإظهار رسالة

مثال 3

```
private void button1_Click(object sender, EventArgs e)
{
this.BackColor=System.Drawing.Color.Green;
}
private void button2_Click(object sender, EventArgs e)
{
this.Font = new Font("Comic Sans MS", 12);
}
private void button4_Click(object sender, EventArgs e)
{
this.Close();
}
```

LABEL

اللافتات
Labels

- توفر اللافتات Label معلومات نصية Text (بالإضافة إلى صور اختيارية) ويتم تعريفها باستخدام الصف Label (صف مشتق من الصف Control).
- تعرض اللافتة Label نصاً Text لا يمكن للمستخدم تعديله بشكل مباشر، ويمكن تغيير النص بشكل برمجي عن طريق تعديل خاصية النص Label's Text ، ويوضح الجدول(8) خصائص اللافتات Label الشائعة .
- من خواصه العامة Font ,Text , TextAlign .
- يمكن تغيير نوع الخط ولون الخط باستخدام الخصائص : Font و ForeColor و BackColor.
- يمكن كتابة نص به ، أو إظهار نص عليه باستخدام الخاصية Text .

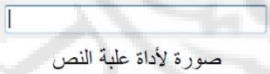
TEXT BOX

مربعات النص
TextBoxes

- مربع النص هو صف classBox وهو منطقة يمكن عرض أي نص فيها بواسطة برنامج أو يمكن للمستخدم كتابة النص text عبر لوحة المفاتيح.
- كلمة المرور password TextBox : هو TextBoxيخفي المعلومات التي تم إدخالها من قبل المستخدم ، وأثناء قيام المستخدم بكتابة المحارف ، تقوم كلمة المرور TextBox بوضع قناع mask لدخل المستخدم عن طريق عرض محرف كلمة المرور.
- إذا قمنا بتعيين الخاصية UseSystemPasswordChar إلى true، يصبح TextBox كلمة مرور مربع نص.
- غالبًا ما يواجه المستخدمون كلا النوعين من TextBoxes، عند تسجيل الدخول إلى جهاز كمبيوتر أو موقع ويب - يسمح اسم المستخدم TextBox للمستخدمين بإدخال أسماء المستخدمين الخاصة بهم.
- تسمح كلمة المرور TextBoxللمستخدمين بإدخال كلمات المرور الخاصة بهم. ويعرض الجدول(8) الخصائص العامة والحدث العام لـ TextBoxes .

الخصائص

يقدم الجدول التالي بعض خصائص هذه الأداة:



صورة لأداة علبة النص

الخصيصة	دورها
TextAlign	لتحديد موضع النص على الأداة (يمين، شمال، أو في الوسط).
MaxLength	لتحديد عدد الأحرف المسموح بكتابتها داخل علبة النص.

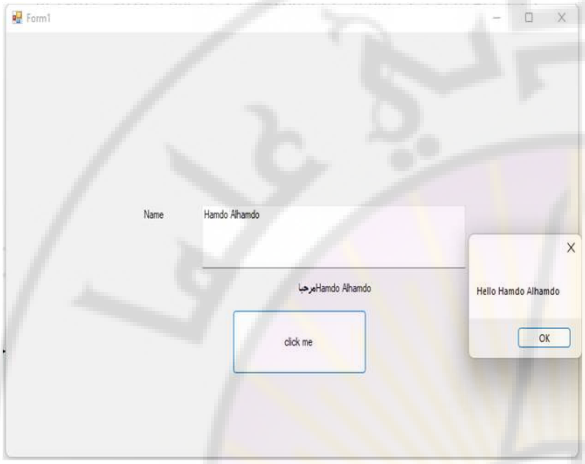
الخصائص

MultiLine	للسماح بتعدد الأسطر في الأداة، لأنها افتراضيا تسمح بسطر واحد فقط.
ReadOnly	لجعل الأداة خاصة بإظهار البيانات فقط، بمعنى تمنع الكتابة فيها وتسمح للمستخدم بالقراءة فقط.
PasswordChar	لتحويل شكل النص إلى حروف مرمزة، وتستعمل هذه الخاصية حينما نريد كتابة كلمة المرور أو أي معلومات سرية.
CharacterCasing	لتحديد حالة الأحرف (كبيرة Upper أو صغيرة Lower أو عادية Normal)
RighToLeft	لتغيير توجيه النص، ليسمح بالكتابة انطلاقا من اليمين، هذه الخاصية إذا أردنا كتابة نصوص عربية أو بأي لغة تبدأ من اليمين.

مثال 4

- **أنشاء زر له الخصائص التالية :**
  - اسم التابع الذي يولد الزر btnName .
  - لونه زهري لون الخط احمر ومظهره Pooup .
  - العبارة Click Here موجودة على وجه الزر .
- **انشاء لافته lable1** فيها العبارة address الخلفية cyan ، **أما اللافتة الثانية Lable2** نكتب بها نص هو Name لون الخط أزرق ، **والثالثة هي Lable3** نحذف النص الذي بداخلها ولون الخط أزرق .
- **أنشاء TextBox** مظهره Borderstyle نختار Fixt3D ، لون الخط Pink زهري ، ومن الخاصية TextAlign نختار Center .
- **معالج الحدث لمربع النص هو TextChanged** يتم توليده عندما يتغير النص في TextBox ، وعند النقر المضاعف عليه يتم إنشاء معالج حدث فارغ .

التنفيذ




```
private void btnName_Click(object sender, EventArgs e)
{
string strName = txtBxName.Text;
MessageBox.Show( "Hello "+ strName) ;
}
label3.Text = " مرحبا"+ strName ;
```

RICH TEXT BOX

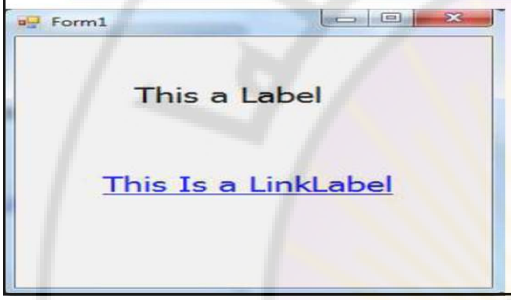
هذه الأداة شبيهة بالأداة السابقة TextBox، كونها تستطيع إظهار وإدخال النص، إلا أنها تتوفر على مزايا إضافية تنعدم في الأولى، ولعل أبرز هذه المميزات هي إمكانية احتوائها على نص متعدد الألوان والخطوط والأحجام، الشيء الذي تفتقد إليه أداة TextBox.

وتستعمل هذه الأداة في برامج معالجة النصوص كبرنامج الوورد Microsoft Word الشهير، وهذه صورة للأداة وهي تحتوي على ثلاثة أسطر وكل سطر يخالف الأسطر الأخرى في لونه وحجمه ونوع خطه:

LINKED LABEL

لتحديد رابط للأداة LinkLabel، اذهب إلى الحدث LinkClicked الخاص بها، أو اضغط عليها مرتين واكتب هذا السطر الذي يقوم بفتح أي رابط أنترنت تريده:

```
private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("www.google.com");
}
```




```
using System;
using System.Windows.Forms;
using System.Diagnostics;
namespace LinkLabelTest
private void linkLabel1_LinkClicked(object
sender,
LinkLabelLinkClickedEventArgs e)
{
Process.Start("www.google.com");
```

CHECK BOX

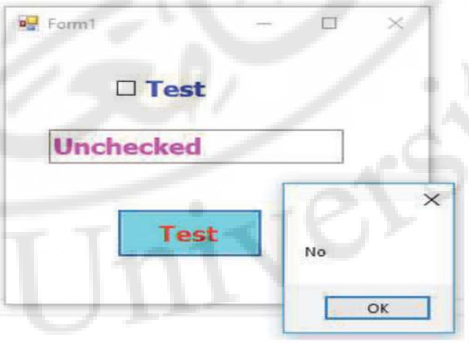
صندوق الاختيار CheckBox

- الـ CheckBox هو مربع صغير إما فارغ أو يحتوي على علامة اختيار، وعندما يقوم المستخدم بالنقر فوق CheckBox لتحديده ، تظهر علامة اختيار في المربع ، وإذا قام المستخدم بالنقر فوق مربع الاختيار مرة يتم إلغاء التحديد، وتتم إزالة علامة الاختيار.
- يمكننا أيضًا تكوين CheckBox للتبديل بين ثلاث حالات (محددة وغير محددة وما بينهما) عن طريق تغيير خاصية Three-State من false إلى true ، ويمكن تحديد أي عدد من CheckBoxes في كل مرة.

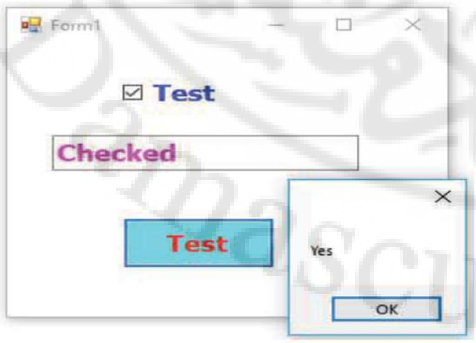
خصائص

- Appearance المظهر الخارجي : بشكل افتراضي ، يتم تعيين هذه الخاصية إلى عادي Normal، ويعرض CheckBox كصندوق اختيار تقليدي، ويكون له شكل زر عند الضغط عليه .
- Checked هذه الخاصية تشير إلى ما إذا كان CheckBox محددًا (يحتوي على علامة اختيار) أو غير محدد (فارغ)، وهذه الخاصية بإرجاع قيمة bool ، وبشكل افتراضي هو false غير محدد .
- CheckState هذه الخاصية تشير إلى ما إذا كان CheckBox محددًا أو غير محدد مع قيمة من تعداد CheckState هي :
  - ✓ Unchecked غير محدد .
  - ✓ Checked محدد .
  - ✓ Uneterminate أو ما بينهما .
- المثال التالي يوضح الـ CheckState في حالة false وفي حالة true .

مثال 5



الحالة الافتراضية لصندوق الاختيار



عند الضغط على صندوق الاختيار

مثال 5

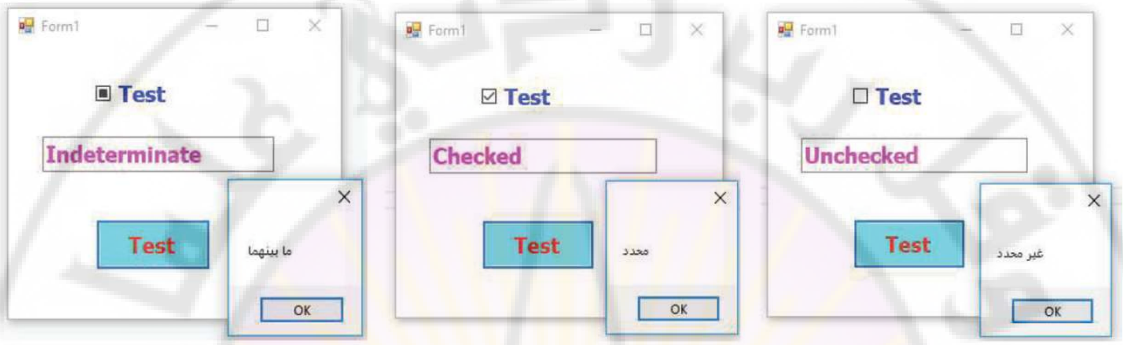
مثال : انشاء زر ومربع نص وصندوق اختيار (false)
ThreeState في الحالة false

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication5  {
    public partial class Form1 : Form {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = checkBox1.CheckState.ToString();

              if ( checkBox1.Checked )
               MessageBox.Show(" Yes ");
            else
                MessageBox.Show(" No ");
         }
```

THREE STATE

```
private void button1_Click(object sender, EventArgs e)
    {
            textBox1.Text = checkBox1.CheckState.ToString();

              if (checkBox1.CheckState.ToString() == " Unchecked ")
            {
                MessageBox.Show("  غير محدد  " );
            }
            else if (checkBox1.CheckState.ToString() == " Checked ")
            {
                MessageBox.Show("  محدد  " ) ;
            }
            else
            {
                MessageBox.Show("  ما بينهما  " ) ;
            }
        }
    }//end class
}// end namespace
```

التنفيذ



عند الضغط للمرة الثانية أي ما بينهما

عند الضغط أول مرة أي محددة

الحالة الافتراضية أي غير محددة

تمرين

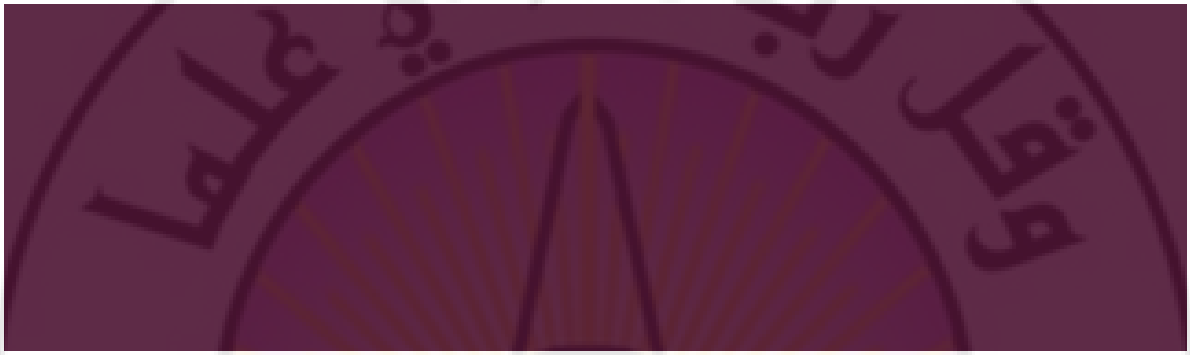
عند الضغط على Bold

الشكل الافتراضي

عند الضغط على Italic

عند الضغط على Bold و Italic

النهاية ....



RADIO BUTTONS , GRAPHICS AND COMBO BOX

ENG.HAMDO AL-HAMDO.

RADIO BUTTON

أزرار الاختيار RadioButton

- تشبه أزرار الاختيار (التي يتم تعريفها من خلال الصف RadioButton) صناديق الاختيار CheckBoxes في أنها تحتوي أيضًا على حالتين - محددة selected وغير محددة not selected ومع ذلك ، تظهر RadioButtons عادة كمجموعة group، حيث يمكن تحديد RadioButton واحد فقط في كل مرة.
- اختيار زر واحد RadioButton في المجموعة يجبر الآخرين على أن يتم إلغاء تحديدهم، لذلك ، يتم استخدام أزرار الاختيار RadioButtons لتمثيل مجموعة من الخيارات الحصرية المتبادلة (لا يمكن تحديد خيارات متعددة في نفس الوقت).
- جميع أزرار RadioButtons يجب أن تضاف إلى حاوية مثل : GroupBoxes أو Panels .

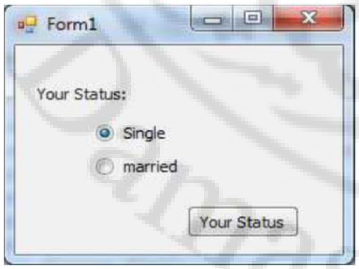
RADIO BUTTON

RadioButton properties and an event	Description
Common Properties	
Checked	Indicates whether the RadioButton is checked.
Text	Specifies the RadioButton's text.
Common Event	
CheckedChanged	Generated every time the RadioButton is checked or unchecked. When you double click a RadioButton control in design view, an empty event handler for this event is generated.

الجدول(11) | RadioButton properties and an event.

Checked يشير إلى ما إذا كان RadioButton محددًا أم لا
Text يحدد اسم زر الاختيار .

مثال




```
private void button1_Click(object sender,
EventArgs e)
{
if (RBsingle.Checked == true)
{
MessageBox.Show("You are Single ");
}
else
{
MessageBox.Show("You are Married ");
}
}
```

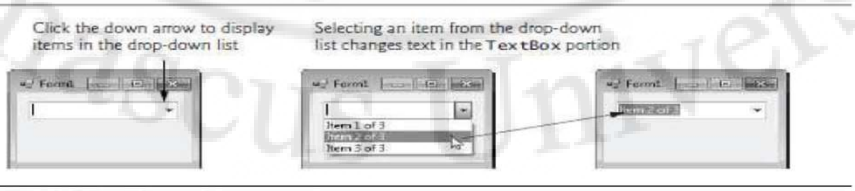
COMBOBOX

صندوق اختيار متاح
ComboBox

- يتمتع عنصر التحكم ComboBox بميزات TextBox مع قائمة منسدلة drop-down list الذي يحتوي على قائمة يمكن تحديد قيمة منها.
- يظهر ComboBox عادةً كـ TextBox مع سهم للأسفل إلى اليمين.
- افتراضيًا ، يمكن للمستخدم إدخال نص في مربع النص أو النقر على السهم لعرض قائمة بالعناصر المحددة مسبقًا ، وإذا اختار المستخدم عنصرًا من هذه القائمة ، فسيتم عرض هذا العنصر في مربع النص.
- إذا كانت القائمة تحتوي على عناصر أكثر مما يمكن عرضه في القائمة المنسدلة ، فسيظهر شريط تمرير *scrollbar*.

COMBOBOX

- يتم تعيين الحد الأقصى لعدد العناصر التي يمكن عرض قائمة منسدلة في وقت واحد بواسطة الخاصية MaxDrop- DownItems ، و يعرض الشكل (1) نموذج ComboBox في ثلاث حالات مختلفة.

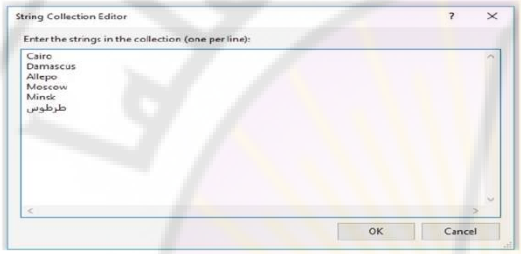


الشكل(1) | ComboBox demonstration.

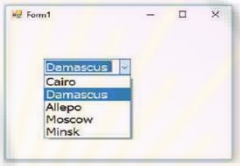
ADD ITEMS

مثال : صندوق اختيار متاح ComboBox

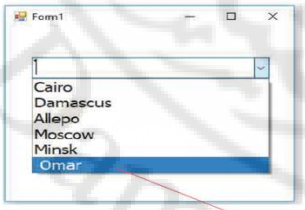
- تجميع عدد من العناصر على شكل قائمة مخفية ، وعند الضغط عليها تظهر جميع العناصر التي بداخلها ، ويمكن اختيار أي عنصر منها بالضغط عليه .
- ننشئ ComboBox ونسميه cbxCity ، نختار الخاصية Items ونختار Callaction فتظهر لنا نافذة (محرر نصوص) ونكتب العناصر كما يلي :



- نضغط على Ok يتم حفظ العناصر بداخلها .
- ننفذ البرنامج نحصل على الواجهة التالية .



ADD ITEM



- الطريقة الثانية لإدخال العناصر إلى ComboBox كما يلي :
  - انشاء زر اسمه btnAddItem ونكتب عليه Add Item .
  - نكتب داخل التابع للزر البرنامج التالي للإدخال العناصر عن طريق الضغط على الزر .
  - كلما نضغط يتم ادخال عنصر ، نضغط مرتين يتم ادخال نفس العنصر مرتين وهكذا .

```
private void btnAddItem_Click(object sender, EventArgs e)
        {
            cbxCity.Items.Add(" Omar");
        }
```

ادخل العنصر Omar إلى عناصر الـ ComboBox

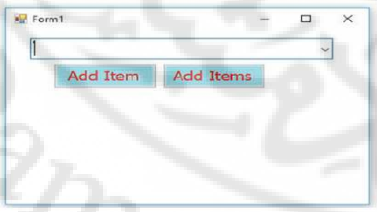
ADD ITEMS

- الطريقة الثالثة لإدخال العناصر إلى الـ ComboBox كما يلي :
  - انشاء ثاني ونسميه btnAddItems ونكتب عليه Add Items .
  - نكتب داخل التابع للزر btnAddItems البرنامج التالي :

```
private void btnAddItems_Click(object sender, EventArgs e)
        {
            String []str = { "طرطوس", "القنيطرة" , "حماة", "حمص"  "} ;
             cbxCity.Items.AddRange(str);
        }
```

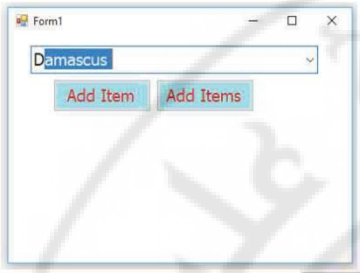
  - تم انشاء مصفوفة محرفية .
  - العبارة cbxCity.Items.AddRange إضافة جميع عناصر المصفوفة إلى الـ ComboBox ، عند الضغط على الزر Add Items .
  - عند تنفيذ البرنامج تظهر الوجهة التالية .

الخصائص

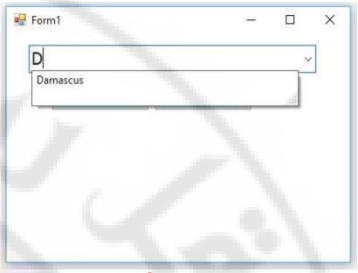


- إذا أردنا البحث عن كلمة موجودة في الـ ComboBox نختار الخاصيتين :
  - الخاصية الأولى AutoCompleteMod وفيها : Suggest , Append , SuggestAppent ، وتعني اقتراح الكلمة ، تكملة الكلمة المقترحة ، اقتراح بالإضافة إلى تكملة الكلمة المقترحة .
  - الخاصية الثانية AutoCompleteSource نختار منها الـ ListItems ، أي اقتراح الكلمة هي من القائمة الموجودة في الـ ComboBox وهي مصدر الكلمات .

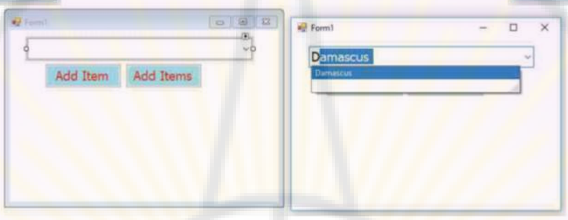
الخصائص



Append



Suggest



SuggestAppend

TOOL TIPS

أدوات التلميح
ToolTips

- إن تلميحات الأدوات المعروضة في Visual Studio تساعدنا على التعرف على ميزات IDE ، وتكون بمثابة تذكير مفيد لوظيفة أي رمز من شريط الأدوات، وتستخدم العديد من البرامج نصائح الأدوات لتذكير المستخدمين بكل هدف من عناصر التحكم.
- على سبيل المثال ، يحتوي Microsoft Word على نصائح الأدوات التي تساعد المستخدمين على تحديد الغرض من رموز التطبيق.
- يوضح هذا القسم كيفية استخدام التلميحات لعناصر ToolTip لإضافة معلومات للأدوات في التطبيق .

TOOL TIPS

- يمكن تغيير شكل التلميح من مستطيل إلى بالون Balloon ، وذلك بتغيير الخاصة IsBalloon من False إلى True .
- يمكن كتابة عبارة ما (عنوان الرسالة) على التلميح من الخاصية ToolTip Title .
- يمكن إضافة أيقونة Icon على التلميح (Info , Warning , Error) من الخاصية ToolTipIcon .
- حتى يكون ظهور واختفاء التلميح بشكل مريح نغير الخاصية UseFading من False إلى True .

GRAPHICS

الصف Graphics

- تزودنا C# من خلال الصف Graphics بعدد من التوابع التي تسمح لنا برسم صورة معينة أو شكل هندسي ما (خط ، مستطيل ، قطع ناقص ، متعدد الأضلاع , . . الخ) على مكون ما .
- **يوجد نوعان من التوابع :**
  - الأول يسمح لنا برسم حدود الشكل الهندسي فقط (شكل غير مصمت)
  - الثاني يسمح لنا برسم الشكل الهندسي مصمت .
- ومن الجدير بالذكر أن مبدأ الإحداثيات هو الزاوية اليسارية العليا لمنطقة الرسم ، مع العلم بأن اتجاه محوري الرسم موضحان في الشكل التالي :

DRAW STRING

رسم نص (سلسلة محرفية)
Draw String

- يمكن استخدام توابع الرسم المختلفة والمعرفة في الصف Graphics من أجل رسم النصوص والأشكال الهندسية .
- نستخدم التابع DrawString من أجل رسم نص ويحتوي على عدد من الوسائط وله الشكل التالي :

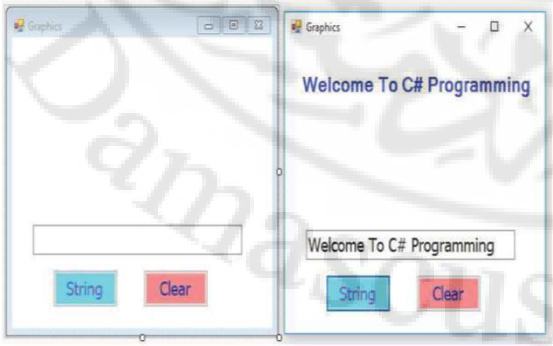
```
DrawString(string str,Font font, Brush brush, Point point);
```

- str : هي السلسلة الواجب رسمها .
- font : هو غرض من الصف Font ، الذي له الشكل التالي :

```
Font font = new Font(" Tahoma ", 16, FontStyle.Bold);
```

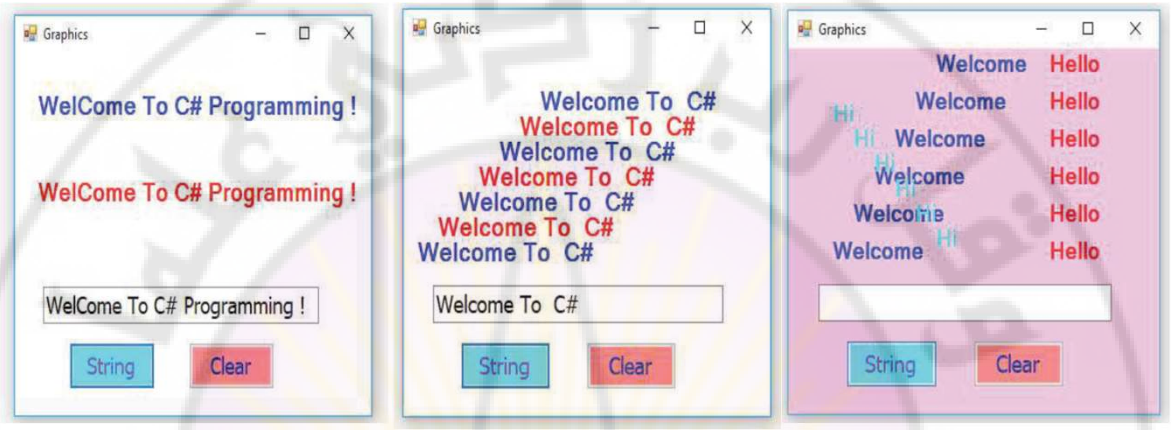
- brush : هو غرض من الصف Brush الذي يحدد لون الرسم للسلسلة .
- point : هو غرض من الصف Point الذي يحدد بداية رسم السلسلة .

EXAMPLE




```
private void btnString_Click(object sender, EventArgs e)
    {
      string str = txtString.Text;
      Font myFont = new Font(" Tahoma ", 16, FontStyle.Bold);
      Point point = new Point(20, 30);
    this.CreateGraphics().DrawString(str, myFont, Brushes.Yellow, point);
  }

 private void btncClear_Click(object sender, EventArgs e)
       {
           this.CreateGraphics().Clear(Color.White);
       }
  }//end class
}// end namespace
```

تمرين



النهاية ....

THREAD+ CREATELE MENT

ENG.HAMDO AL-HAMDO.

THREAD:

- أولاً كل عملية في نظام التشغيل تقوم بإنشاء proccess خاص بها.

THREAD:

- هي عبارة عن proccess خاصة بالنظام تقوم بتنفيذ الكود الخاص بالبرنامج الخاص بنا ويمكن ان تكون one thread او multi thread في نفس الوقت .

التعامل مع THREAD:

- للتعامل مع thread نحن بحاجة الى استدعاء المكتبة using.system.threading;
- برنامج عملي للتوضيح يقوم بالطباعة باستخدام thread .

:ONE THREAD

```
class Program
    {
        public static void Test()
        {Thread.Sleep(5000); }
        static void Main(string[] args)
        { Thread Tr = new Thread(Test);
            Console.WriteLine("Hello 1");
            // Test();
            Tr.Start();
            Console.WriteLine("Hello 2");
            Console.ReadKey();
        }}}
```

:MULTI THREADING

```
private void Button1_Click(object sender, EventArgs e)
        { Thread Tr = new Thread(CreateElement);
            Tr.Start();
            TextBox t = new TextBox();
            this.Controls.Add(t);
            //CreateElement();  }
        private void CreateElement()
        {
            for (int i = 0; i < 100; i++)
            {
                for (int j = 0; j < 100; j++)
                {
                    Button b = new Button()
                    {
                        Width = 30,
                        Height = 30,
                        Margin= new Padding(0),
                        FlatStyle = FlatStyle.Flat,
                        BackColor = Color.Red};
                    flowLayoutPanel1.Invoke(new Action(() => flowLayoutPanel1.Controls.Add(b)));}}}}}
```

النهاية ....

EVENTS

ENG.HAMDO AL-HAMDO.

للتذكير:

وسائط معالج الحدث Event Handler Parameters

يتلقى معالج الحدث وسيطين عند استدعائه :

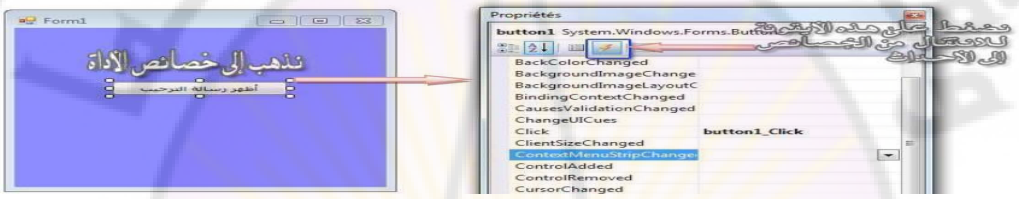
- ✓ اول وسيط (بارامتر) هو مرجع غرض يدعى المرسل sender هو مرجع إلى غرض الذي يولد الحدث event .
- ✓ الثاني هو مرجع إلى غرض من الصف المشتق EventArgs ويسمى عادة e ويحتوي هذا الغرض على معلومات إضافية حول الحدث الذي ظهر .
- ✓ EventArgs : هو صف أساسي لكافة الصفوف التي تمثل معلومات الحدث .

الحدث:

تذكير (الأحداث):

حينما تضغط على زر الماوس، أو على زر من أزرار لوحة المفاتيح مثلا، فإن ذلك يرسل رسالة إلى دالة مرتبطة بهذا الحدث لتقوم بتنفيذها، كما رأينا في المثال العملي الأول، عندما نضغط بالماوس على الزر Button فإن ذلك ينفذ الدالة التي تظهر رسالة ترحيبية.

تتوفر كل أداة على عدد كبير من الأحداث والتي يمكنك رؤيتها في نافذة الخصائص بعد تحديد الأداة المراد مشاهدة أحداثها، كما تظهر الصورة التالية:



أنواع الاحداث:

الحدث	دوره
Click	ينفذ هذا الحدث حينما نضغط على الأداة بزر الماوس الأيسر مرة واحدة.
DoubleClick	ينفذ هذا الحدث حينما نضغط على الأداة بزر الماوس الأيسر مرتين متتاليتين.

احداث لوحة المفاتيح:

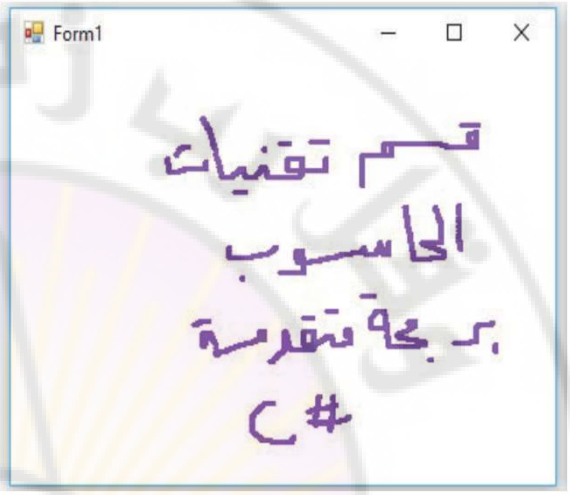
FormClosed	ينفذ هذا الحدث بعد إغلاق الفورم.
Load	ينفذ هذا الحدث عند بداية تحميل الفورم.
KeyDown	هذا الحدث يتعلق بلوحة المفاتيح، ويتولد حينما نضغط على مفتاح ما، وهو يأتي قبل الحدثين KeyPress و KeyUp
KeyPress	هذا الحدث يتعلق بلوحة المفاتيح أيضا، ويتولد حينما نضغط على مفتاح ما وهو ينفذ بعد KeyDown و قبل KeyUp
KeyUp	هذا الحدث يتعلق بلوحة المفاتيح أيضا، ويتولد حينما نرفع إصبعنا عن المفتاح وهو طبعا يتولد بعد الحدثين KeyDown و KeyPress

احداث الفأرة:

MouseDown	هذا الحدث شبيه بالحدث Click ولكنه لا يتطلب منك أن تترك زر الماوس بعد الضغط، فقط بمجرد الضغط على الأداة يتولد هذا الحدث.
MouseUp	هذا على عكس الحدث السابق، ينفذ بعد أن تترك الضغط بزر الماوس، كما لو أن مجموع هذين الحدثين يشكل الحدث Click
MouseMove	يتولد هذا الحدث عند مرور مؤشر الماوس فوق الأداة.

احداث الفأرة: مثال عملي للتوضيح:

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApplication14
{
public partial class Form1 : Form
{
bool shouldPaint = false; // 8 //determines whether to paint
public Form1()
{
InitializeComponent();
}
private void Form1_Load(object sender, EventArgs e)
{
}
```



احداث الفأرة: مثال عملي للتوضيح:

يحتوي الصف MouseEventArgs على معلومات متعلقة بحدث الفأرة ، مثل إحداثيات Right, Left or Middle للمؤشر الخاص بالفارة ، وضغط زر الفأرة x-and y وعدد مرات النقر بالفأرة.

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{shouldPaint = true; }
private void Form1_MouseUp(object sender, MouseEventArgs e)
{shouldPaint = false; }
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
if (shouldPaint) //check if mouse button is being pressed
{
// draw a circle where the mouse pointer is present
using (Graphics graphics = CreateGraphics())
{ graphics.FillEllipse(new SolidBrush(Color.BlueViolet), e.X, e.Y, 4, 4);
} // end using; calls graphics.Dispose()
} // end if
}//end class
}// end namespace
```

أحداث لوحة المفاتيح:

تحدث الأحداث الرئيسية للمفتاح Key عند الضغط على مفاتيح لوحة المفاتيح وتحريرها.

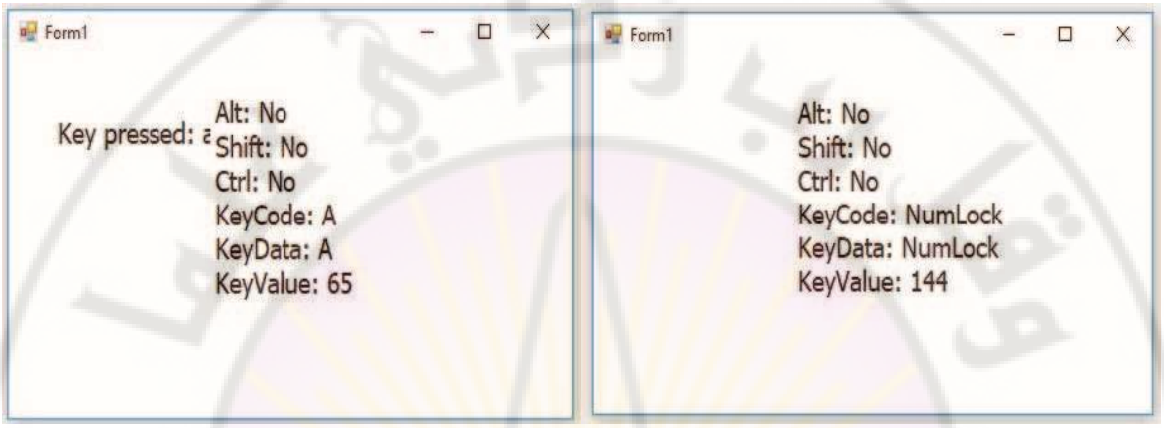
KeyPress
KeyUp
KeyDown

ملاحظة: يحدث الحدث KeyPress عندما يقوم المستخدم بالضغط على مفتاح يمثل حرف ASCII ويمكن تحديد المفتاح المحدد باستخدام الخاصية KeyChar عن طريق غرض من الصف KeypressEventArgs الخاصة بمعالج الحدث.
لايشير الحدث KeyPress إلى ما اذا كانت المفاتيح المعدلة مثل ShiftوCtrlوAlt قد تم الضغط عليها عند حدوث حدث رئيسي.

احداث لوحة المفاتيح: مثال عملي للتوضيح:

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{ charLabel.Text = "Key pressed: " + e.KeyChar; }
 private void Form1_KeyDown(object sender, KeyEventArgs e)
 { keyInfoLabel.Text =
"Alt: " + ( e.Alt ? "Yes" : "No" ) + '\n' +
"Shift: " + ( e.Shift ? "Yes" : "No" ) + '\n' +
"Ctrl: " + ( e.Control ? "Yes" : "No" ) + '\n' +
 "KeyCode: " + e.KeyCode + '\n' +
 "KeyData: " + e.KeyData + '\n' +
"KeyValue: " + e.KeyValue; }
private void Form1_KeyUp(object sender, KeyEventArgs e)
 { charLabel.Text = "";
 keyInfoLabel.Text = ""; }
}//end class}// end namespace
```

احداث لوحة المفاتيح: مثال عملي للتوضيح:



شرح بعض الخصائص :

- ترجع الخاصية KeyCode قائمة بقيمة المفاتيح Keys (السطر 24) ، أما الخاصية KeyCode تقوم بإرجاع المفتاح المضغوط ، ولكنها لا توفر أي معلومات حول مفاتيح التعديل modifier keys ، ويتم تمثيل كل من الحرف الصغير والكبير مثل (a) كمفتاح A .
- ترجع الخاصية KeyData (السطر 25) أيضاً قائمة بقيمة المفاتيح ولكن هذه الخاصية تتضمن بيانات حول مفاتيح التعديل ، وبالتالي ، إذا كان " A " هو الدخل ، فستعرض KeyData أنه تم الضغط على كل من المفتاح A ومفتاح Shift ، وأخيراً ، يُرجع KeyValue (السطر 26) قيمة int تمثل مفتاحًا مضغوطًا، وهذا int هو رمز المفتاح ، ورمز المفتاح مفيد عند اختبار مفاتيح غير ASCII مثل F12 .

النهاية ....

التعامل مع الملفات والمجلدات و الأقراص

ENG.HAMDO AL-HAMDO.

اولاً:

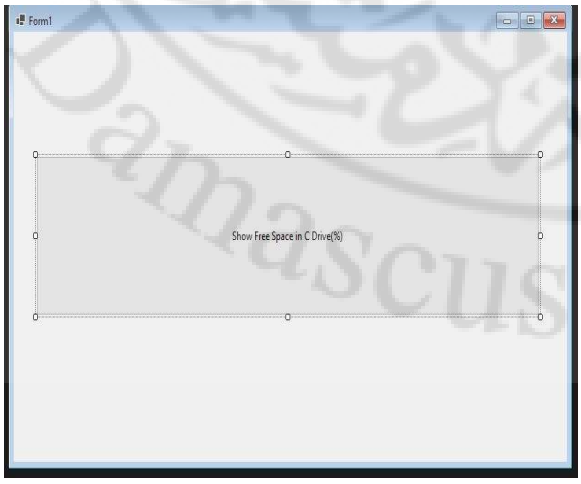
- للولوج الى عالم الملفات والمجلدات وحتى نستطيع التعامل معها يجب استعمال المجال او المكتبة USING SYSTEM.IO اختصاراً ل input/output حيث تحتوي على كل لفئات التي تحتاجها للتعامل مع الملفات Files والمجلدات Folders والتعامل مع الوحدات Disks .

الفئة الأولى:

- Drive Info:حيث تستخدم للحصول على معلومات حول الوحدات الصلبة مثل القرص :C وغير ذلك وهذه المعلومات هي :

1. المساحة الفارغة
2. الحجم الكلي
3. الخ.....

مثال عملي بسيط:



- برنامج يقوم بحساب المساحة الفارغة بالنسبة المئوية لوحدة معينة مثل القرص :C

مثال عملي بسيط:

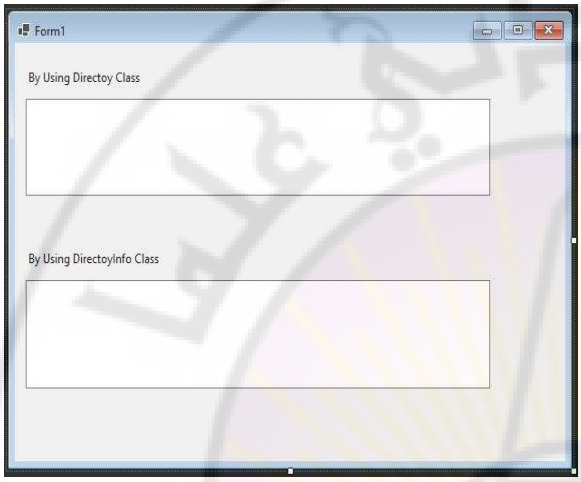
```
private void button1_Click(object sender, EventArgs e)
    {

        DriveInfo d = new DriveInfo("C:");
        double freeSpace = d.AvailableFreeSpace;
                double total = d.TotalSize;

        double result = freeSpace / total * 100;
        MessageBox.Show("your Free Space in C:" + Math.Floor( result)+"%");
    }
```

- إعلان كائن من الفئة DriveInfo وإضافة اسم القرص المراد
- حيث تعيد الخاصية AFS المساحة الفارغة من القرص
- حيث الخاصية TotalSize تعيد المساحة الكلية
- إعلان متغير لتخزين قيمة المساحة المتبقية بشكل مئوي

الفئة الثانية:

- Directory and DirectoryInfo:تمكننا من الحصول على معلومات المجلدات وإجراء بعض العمليات على المجلدات مثل انشاء المجلدات الفرعية
- الفرق: الأول نستطيع الولوج للمجلدات مباشرتاً عبر مسارها بينما الثاني نحتاج الي تعريف متغير للحصول على وظائفها

مثال عملي بسيط:



- برنامج يقوم بتعبئة ListBox بملفات مجلد معين

:DRIVE

```
public FORM1()
{
foreach(string file in Directory.GetFiles("C:\\user"));
listBox1.Items.Add(file);
}
```

- استخدمنا foreach لعرض التكرار وGetFiles لعرض محتويات مجلد معين

DRIVEINFO:

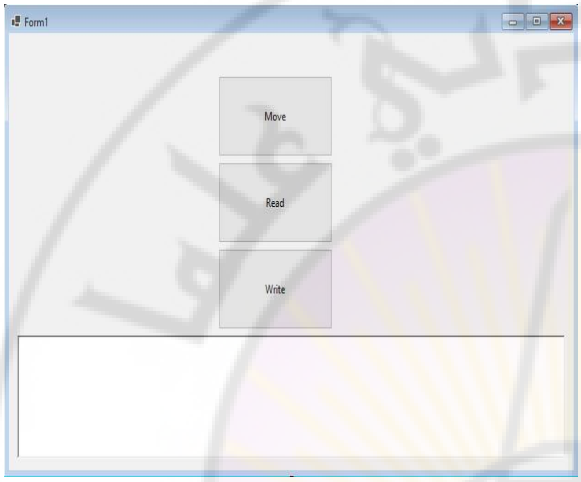
```
public FORM1()
{
DirectoryInfo d = new DirectoryInfo( "C:\\user");
foreach(FileInfo file in d.GetFiles());
listBox1.Items.Add(File.Name);
}
```

- استخدمنا FileInfo لان هذا السطر يعيد معلومات عن كل ملف
- لإظهار أسماء الملفات
- استخدمنا foreach لعرض التكرار و GetFiles لعرض محتويات مجلد معين

الفئة الثالثة:

- File and FileInfo:للتعامل مع الملفات مثل عمليات النقل والنسخ والحذف والتحقق من وجود ملف معين ضمن مسار محدد

مثال عملي بسيط:



- برنامج يقوم بنقل ملف من مسار معين الى مسار اخر ويقوم بقراءة ملف نصي والكتابة على ملف نصي

:FILE

```
private void button1_Click(object sender, EventArgs e)
    {
File.Move("C:\\1.png","D:\\mypic.png");}
private void button2_Click(object sender, EventArgs e)
    {
if(File.Exists("C:\\mytext"))
```

- اذا كان الملف موجود ضمن المسار

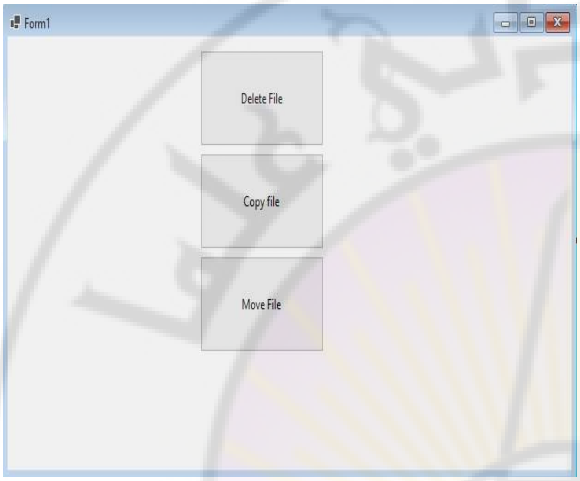
```
richtextBox1.Text=File.ReadAllText("C:\\mytext.txt");}
```

- اقرأ محتوى الملف ويتم طباعتها كسطور ضمن RichTextBox

```
private void button3_Click(object sender, EventArgs e){
File.WriteAllText("C:\\mytext.txt",richTextBox1);}
```

- تحديث محتوى ملف معين

مثال عملي بسيط:



- برنامج يقوم بنقل ملف من مسار معين الى مسار اخر او يقوم بعملية نسخ ويقوم بعملية حذف لملفات معينة

:FILEINFO

- FileInfo fi = new FileInfo("C:\\mytext.txt");
- نصرح عن متغير من نوع FileInfo
- private void button1_Click(object sender, EventArgs e)
- {
- Fi.MoveTo("D:\\myFile2.txt");}
- تم نقل الملف وإعادة تسميته ب myFile2
- private void button2_Click(object sender, EventArgs e)
- {
- Fi.CopyTo("D:\\myFile2.txt",True);}
- في حال وجود نفس الملف نضيف True للكتابة فوق الملف و False لإلغاء النسخ
- private void button3_Click(object sender, EventArgs e){
- Fi.Delete();}

النهاية ....