

C# Programming Language

ان لغة السى شارب تتبع مجموعة لغات Net Frame Work. وهذه البيئه 

تحتوي على عدة لغات لكن كل منهم له الكود الخاص به وتوجد بها لغه وسيطيه  وبالتالي هي لغه لا تعتمد Microsoft Intermediate Language (MSIL)

على الاله فهى تعمل على اي جهاز

ان C# تضم مميزات VB - C++ بما لها من سهولة التعامل فى الواجهات الرسوميه بالإضافة الى قوة البرمجه . 

ان الفارق الجوهرى بين C# و C++ هو كالتالى : 

<i>C++</i>	<i>C#</i>
هي لغة تستطيع بها ان تستخدم مفهوم البرمجه كائنية التوجه	هي لغه مبنيه بالاساس على مفهوم البرمجه كائنية التوجه او Object Oriented

الهيكل التنظيمى لبرنامج C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
    class Program ← class
        ← الرئيسي للبرنامج ويوجد بداخله الدالة الرئيسية
    {
        static void Main(string[] args)
        {
            ← يكتب هنا داخل الدالة الرئيسية الكود الذى سوف ينفذ فهو يبدع التنفيذ من الدالة
            Main()
        }
    }
}

```

الأساسية والتي تستدعي عند كتابة اي برنامج

namespace

الأساسية والتي تستدعي عند كتابة اي برنامج

class

الرئيسي للبرنامج ويوجد بداخله الدالة الرئيسية

يكتب هنا داخل الدالة الرئيسية الكود الذى سوف ينفذ فهو يبدع التنفيذ من الدالة

Main()

و نستطيع انه نمثله بالشكل التالي :

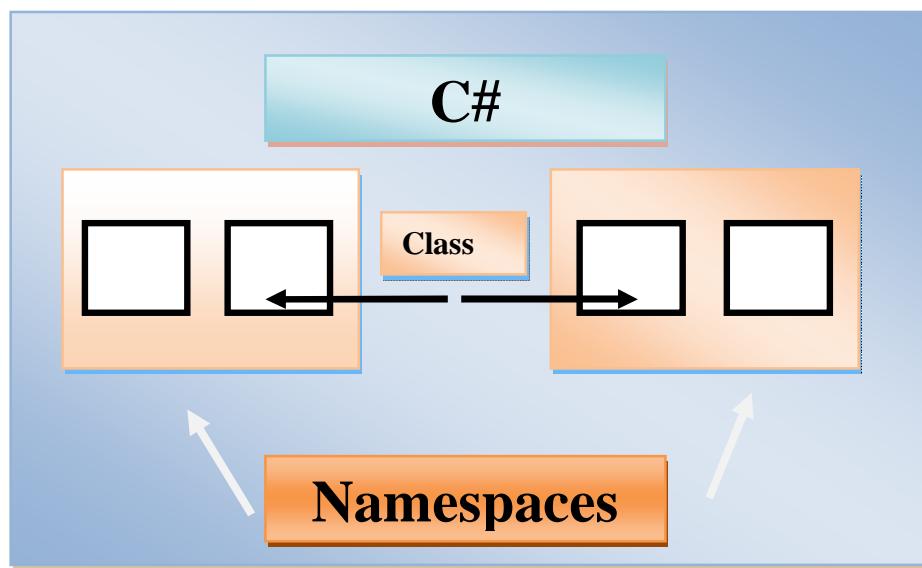
نأتى لتفسير التكوين الرئيسي لبرنامج C#

Using هى كلمة محفوظة داخل اللغة وتستخدم للنداء على اسم معين ليصبح

جزء من البرنامج

ويعرف مجازا بفضاء الاسماء وهو عباره عن مجلد يحتوى

بداخله على Classes جاهزه للعمل عليها ولها اسماء محددة



Classes هى وحدات محووظة داخل اللغة وتحتوى على جزئين رئيسيين:

وهي المتغيرات والتى تحتوى على بيانات **Data Member -1**

Function Member -2 دوال تقوم بتنفيذ عمليات على الدوال المخزنة

ان اى برنامج داخل C# قد يحتوى على اكثر من **namespace** على الاقل

لابد من وجود واحد فقط وقد يحتوى على اكثر من Class لكن على الاقل

يجب ان يحتوى على **Class** واحد

وبعد **System** هو **namespace** الرئيسي بصفته يحتوى على **classes**

تحتوى على دوال التوصيف للدخل والخرج اى برنامج

 ان وظيفة الدالة الرئيسية Compiler Main() يبدء دائمًا بتنفيذ خطوات

البرنامج من عندها أيا كان موضعها ولا بد أن تكون موجودة داخل class

الرئيسي للبرنامج والذي يعطى اسم الافتراضي Program ومن الممكن

وضع تلك الدالة داخل أي class وسعنتره أنه الرئيسي الذي يحتوى على

تلك الدالة

 ان لغة C# حساسة لحالة الاحرف بمعنى انه يجب ان يراعى فيها capital

او small

 شرح الاوامر البرمجية داخل لغة C# :-

 اوامر الطباعة :

هناك امران اساسيات داخل C# وهم كيفية التعامل مع الدخل والخرج والذي

لا غنى عنهم داخل أي برنامج وهم ممثلان في دالتين يتبعان نفس class

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string myname;
            myname = Console.ReadLine();
            Console.WriteLine("hello", myname);
            Console.ReadLine();
        }
    }
}

```



دوال الدخول والخرج والتى تتبع جميعها التصنيف Console	
هذه الدالة تقوم بطباعة الخرج ويظل المؤشر فى مكانه	Write()
هذه الدالة تقوم بطباعة الخرج وينتقل المؤشر الى سطر جديد اسفل الخرج	WriteLine()
تقوم هذه الدالة بقراءة المدخلات ودائما ما تعتبر لغة السى شارب ان جميع المدخلات على انها نصوص لذلك يجب تحويل المدخلات بعد قرأتها	ReadLine()

Replacement Operator

```
string myname;
string hellomessage;
myname = Console.ReadLine();
hellomessage = "hello mr";
Console.WriteLine("hello{0},{1}", myname,hellomessage);
Console.ReadLine();
```

Replacement Operator

أنواع البيانات داخل لغة C# :

أنواع البيانات داخل السى شارب وسعة كلا منها		
النوع	فيما يستخدم	سعته
Byte	لأرقام	bits 0 الى 8
Short	لأرقام	2byte الى bits16
Int	لأرقام	32byte to 4byte
Char	للحروف	8byte for one char
Float	للكسور	4byte
Boolean	لقيم المنطقية	True or false
String	لنصوص	Stream open of char
Double	لأرقام التي تحتوى على علامات عشرية	8byte

القاعدہ العامہ لتعريف المتغيرات داخل C# :

```

1 Variable decleration
2 syntax : Var_type var_name = initial_value;
3
4 example : int x = 5;           قيمه ابتدائيه
5     string y = "ali";          نوع المتغير
6     char c = "m";
7     double s = 34.23;
8     float t = 2.3;
9     and so on

```

داخل لغة السى شارب لابد من الاعلان عن المتغيرات صراحة وتعريفها قبل استخدامها

او اعطاؤها اي قيمة

```

using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string myname;
            string hellomessage;
            myname = Console.ReadLine();
            hellomessage = "hello mr";
            Console.WriteLine("hello{0}, {1}", myname, hellomessage);
            Console.ReadLine();
        }
    }
}

```

تم تعریف المتغيرات قبل استخدامها او اعطاؤها قيم
 يأخذ المتغير القيمة من المستخدم اثناء تنفيذ البرنامج
 اعطيه قيمة ابتدائية

دوال التحويل داخل لغة السى شارب :

قلنا سابقا ان السى شارب تعتبر جميع المدخلات على انها نصوص فكيف اذا ما تم

ادخال ارقام او انواع اخرى فهناك دوال للتحويل بين انواع البيانات

دوال التحويل داخل السى شارب

Parse

Convert

ولدينا الامثله التاليه :

```

string myname;
string hellomessage;
int myage;
myname = Console.ReadLine();
myage = Int32.Parse(Console.ReadLine(myage));
myage = Convert.ToInt32(Console.ReadLine(myage));

```

تحويل القيم قبل استخدامها بالطريقتين

تستطيع الان استخدام نفس الداللين لتحويل باقى انواع البيانات حيث انك مضطر

لاستخدامها فى جميع الاحوال

انواع المعاملات داخل السى شارب :

Arithmetic operators

المعاملات الرياضيه

Multiplication	*
Addition	+
Subtraction	-
Module	%
Dividing	/
Increment by 1	++
Decrement by 1	--

Competitive Operators

معاملات المقارنه

Smaller than

<

Greater than	>
Equal	==
Smaller than or equal	<=
Greater than or equal	=>
Not equal	!=

Logical operator	المعاملات المنطقية
And	&&
Or	
not	!

معاملات المساواه	
Assignment	
Assignment plus	+=
Assignment minus	-=

ان جميع المعاملات التي تم سردها في الجداول السابقة لها الاستخدام الاوسع

داخل السى شارب وتدخل في معظم العمليات الحسابيه والمنطقية

Flow Control Statements

استخدام الدوال الشرطية داخل اللغة : Flow Control – statements

الدوال الشرطية من الدوال الواسعة الانتشار داخل لغات البرمجة بصفه عامه

وذلك لاهميتها الكبيره في التحكم في سير البرنامج ولدينا العديد منها داخل

السى شارب .

القاعدہ الشرطیہ IF

القاعدہ العامہ لها :

```

1 IF Statement : syntax : if (condition)
2           |          |
3           |          |
4           |          |
5           |          |

```

معنی لو تحقق الشرط سوف يتم تنفيذ الجمل البرمجية التي تتبع القاعدہ الشرطیہ

```

static void Main(string[] args)
{
    int mynumber;
    mynumber = int.Parse(Console.ReadLine());
    if (mynumber == 10) ← الشرط المطلوب تحقيقه
    {
        Console.WriteLine("we are win the race");
    }
}

```

الجمله التي ستتفذ اذا تحقق الشرط

القاعده الشرطيه : IF ELSE

القاعده العامه لها :

```

1  IF Elase Statement : syntax : if (condition)
2
3
4
5
6
7
8
  {
    statements ;
  }
  Else
  {
    statements;
  }

```

بمعنى انه اذا تحقق الشرط سيتم تنفيذ الجمل التابعه IF و اذا لم يتم تحقيق الشرط سيتم

تنفيذ الجمل التابعه ELSE

```

static void Main(string[] args)
{
    int mynumber;
    mynumber = int.Parse(Console.ReadLine());
    if (mynumber == 10)
    {
        Console.WriteLine("we are win the race");
    }
    else
    {
        Console.WriteLine("we are lose");
    }
}

```

الجمله الاخرى فى حالة عدم تحقيق الشرط

القاعدہ الشرطیہ NESTEAD IF

القاعدہ الشرطیہ المتداخلہ والصورہ العامہ لہا :

```

1 Nested IF Statement : syntax
2
3 If (condition1)
4     If (condition2)
5     {
6         Statements1;
7     }
8 Else if (condition3)
9     If (condition4)
10    {
11        Statements2;
12    }
13 Else
14     Statements3;
15 Else
16     Statements4;

```

لاحظ دائمًا في حالة if المتداخلة فإن else تتبع اقرب if لها

```

{
    static void Main(string[] args)
    {
        int mynumber;
        mynumber = int.Parse(Console.ReadLine());
        if (mynumber <= 10)
            if (mynumber >= 5)   فى حالة تحقق الشرطين تطبع الجمله الاولى
            {
                Console.WriteLine("we win the race");
            }
            else      فى حالة عدم تطبيق الشرط الثاني بعد الاول تطبع
                      الجمله الثانية
            {
                Console.WriteLine("we are lose");
            }
        else      فى حالة عدم تطبيق اي من الشرطين تطبع الجمله الثالثه
        {
            Console.WriteLine("wrong value");
        }
        Console.ReadLine();
    }
}

```

القاعده الشرطيه : SWITCH CASE

القاعده العامه لها :

```

1  Switch Case statement : syntax
2  Switch (variable or expression)
3  {
4      Case 1: { statements };
5      Break;
6      .
7      .
8      Case n : { statements };
9      Break;
10
11     Default : { statements };
12     Break;
13 }
```

يتم تعريف المتغير ثم في كل مره اعطيه قيمة معين مع كل جملة Case وتنفيذ الجمله المطلوبه اذا ما تساوت هذه القيمه مع اي حاله من تلك الحالات ولا حظ انه بعد كل حاله لابد من وضع الكلمه المحجوزه Break لتفصل كل حاله عن الاخرى ثم في النهايه وضع الحاله Default والتى سوف تنفذ جملتها اذا لم تتحقق اي من الحالات السابقة

```

string thename;
thename = Console.ReadLine();
switch (thename)
{
    case "ali":
        Console.WriteLine("this is my name");
        break;
    case "mohamed":
        Console.WriteLine("this is my brother name");
        break;
    case "basem":
        Console.WriteLine ("this is my frend name");
        break ;
    default :
        Console.WriteLine("i dont know that name");
        break;
}
Console.ReadLine();

```

كل حاله منفصله عن الاخرى

ستنفذ هذه الجمله اذا لم يتتوفر اي شرط من الشرط
السابقه

Loops statements

: الجمل التكراريه

تستخدم هذه الجمل فى تكرار جزء معين من البرنامج اذا ما تحقق شروط معينة

حيث يكون التكرار محدود وله ضوابط وليس الى مالانهايه ولها انواع داخل لغات

البرمجه وسنوضح الان كل جمله وطريقة عمله

: For Loop الجمله التكراريه

القاعدہ العامہ لها :

```

1 For Statement :syntax
2 For (initialize; condition; update);
3 {Statements} ;

```

```

using System.Text;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++) ;
            int k;           الحاله الاخرى تعريف المتغير مسبقا ثم بناء الجمله عليه
            for (k = 0; k < 10; k++) ;

            Console.ReadLine();
        }
    }
}

```

الحاله الاخرى تعريف المتغير مسبقا ثم بناء الجمله عليه

اولا يوضع القيمه الابتدائيه ثم الشرط ثم التحديث بحيث لا يزيد التحديث عن الشرط المحدد ولا حط انه يمكن تعريف المتغير داخل جملة التكرار

```

static void Main(string[] args)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine(i);
    }
    Console.ReadLine();
}
}

```

الجمله التى ستتند فى كل مره يتتحقق فيها الشرط و هي طباعة قيمة المتغير

The screenshot shows the Visual Studio IDE with the output window open. The title bar of the window reads: file:///C:/Users/Sofyany/Documents/Visual Studio 2008/Projects/ConsoleApplication2/ConsoleApp... . The output window displays the following text:

```

0
1
2
3
4
5
6
7
8
9
-
```

القاعده الشرطيه While Loop

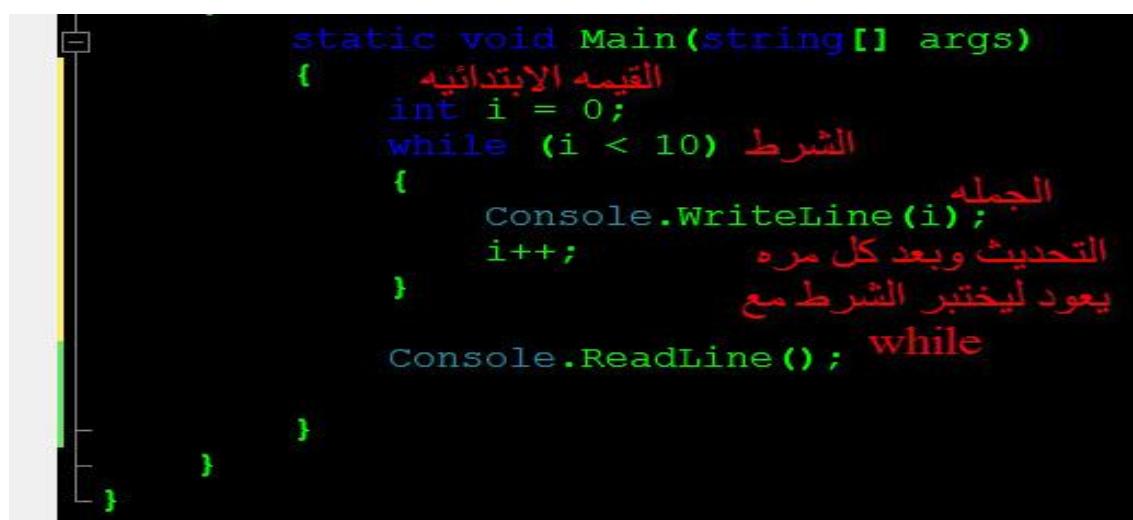
وهذه القاعده تؤكى على استمرار تنفيذ جزء معين من الكود طالما ان الشرط مازال

متحققا والقاعده العامه لها كالتالى

```

1   while Loop statement : syntax
2
3       Initialize;
4   While (condition)
5   {
6       Statements;
7       Update;
8   }

```



The screenshot shows a C# code editor with the following code:

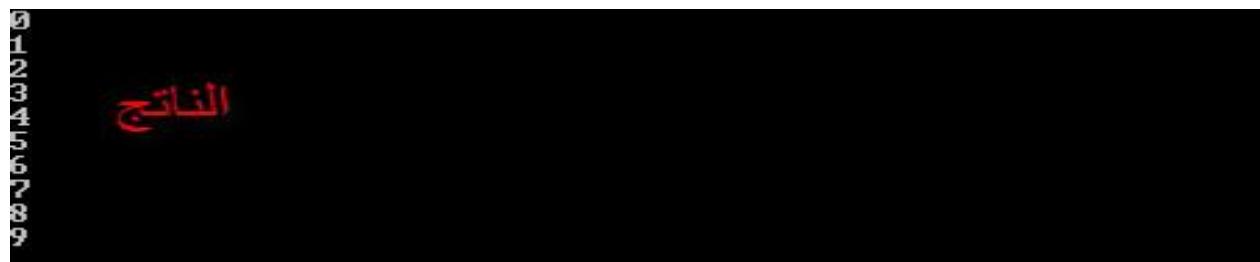
```

static void Main(string[] args)
{
    القيمه الابتدائيه
    int i = 0;
    الشرط
    while (i < 10)
    {
        الجمله
        Console.WriteLine(i);
        التحديث وبعد كل مرره
        i++;
    }
    يعود ليختبر الشرط مع
    Console.ReadLine();
}

```

Annotations in Arabic explain the code:

- القيمه الابتدائيه (Initial value)
- الشرط (Condition)
- الجمله (Statement)
- التحديث وبعد كل مرره (Update after each iteration)
- يعود ليختبر الشرط مع (Returns to check the condition again)



The terminal window displays the following output:

```

0
1
2
3
4
5
6
7
8
9

```

On the left, there is a vertical column of numbers from 0 to 9, likely representing the line numbers or a cursor position.

الناتج

القاعده الشرطيه : DO While

و هذه القاعده لها خصوصيه حيث انه يتم تنفيذ الكود اول مره قبل اختبار الشرط
بمعنى انه اول مره سيتم التنفيذ حتى لو يتحقق الشرط وبعد ذلك يختبر الشرط كل
مره . القاعده العامه لها :

```

1 do while statement : syntax
2 Initialize;
3 do
4 {
5 Statements;
6 Update;
7 }
8 while (condition);
```

لاحظ ان الشرط وضع في نهاية الجمله حتى يتم التنفيذ في المره الاولى اذا لم يتحقق
لكن بعد ذلك فإنه لن يتم التنفيذ الا اذا تحقق الشرط

```

static void Main(string[] args)
{
    int i = 0;           المثال الاول سوف يطبع
    do                  القيمه الاولى ثم يختبر
    {                  الشرط بعدها
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);
    Console.ReadLine();
}
```

The screenshot shows a C# code editor with the following code:

```

static void Main(string[] args)
{
    int i = 0;           المثال الاول سوف يطبع
    do                  القيمه الاولى ثم يختبر
    {                  الشرط بعدها
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);
    Console.ReadLine();
}
```

Arabic annotations are overlaid on the code:

- المثال الاول سوف يطبع
- القيمه الاولى ثم يختبر
- الشرط بعدها

0
1
2
3
4
5
6
7
8
9

الخرج

```
static void Main(string[] args)
{
    int i = 11; لاحظ هنا ان القيمه الابتدائيه
    do تخالف الشرط
    {
        Console.WriteLine(i);
        i++;
    }
    while (i < 10);

    Console.ReadLine();
}
```

11

ومع ذلك قام بطباعة
القيمه الاولى وتوقف عند
ذلك لانه اختبر الشرط
فى المره الثانية ووجده

جملة الخروج عن الحلقة التكراريه عند الوصول الى نقطه معينه : break

والتي تقوم بانهاء الحلقة عند الوصول الى نقطه معينه او تحقيق شرط معين
القاعدہ العامہ لها : تستخدم داخل اي من الجمل التكراریه السابقہ ودائماً توضع
بعد اختبار الشرط كل مرہ ولدينا المثال التالي :

```

1 static void Main(string[] args)
2 {
3     int i;
4     for (i = 0; i < 10; i++)
5     {
6         if (i == 5) break;
7         Console.WriteLine(i);
8     }
9     Console.ReadLine();
10 }

```

سيتم كسر الحلقة عند الوصول الى العدد 5

تم الخروج من الحلقة عند
الوصول الى الرقم 5 حيث قام
بالعد حتى الرقم 4 فقط حسب
شرط الخروج من الحلقة

أمر الاستمرار داخل الحلقة عند تحقيق شرط معين : Continue

هذا الامر يخرج من الحلقة ثم يعود اليها مره اخرى عند شرط معين بمعنى انه

يتخطى تنفيذ جزء معين من الحلقة ثم يعود لينفذ الباقي

ولدينا المثال السابق حيث سيتجاهل البرنامج طباعة الرقم 5 ثم يطبع باقى الارقام

القاعده العامه لها : انها توضع فى نفس مكان break

```

static void Main(string[] args)
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (i == 5)           عند الوصول الى 5 لن ينفذ
            continue;         الامر ويقوم بتنفيذ الباقي
        Console.WriteLine(i);
    }

    Console.ReadLine();
}

```

```

0
1
2
3
4
5
6
7
8
9
ابحث عن الرقم 5 لن تجده

```

Compound Data: Arrays

: 1 المصفوفات او Array

يجب ان يكون كل عناصر المصفوفه لها نفس نوع البيانات وممكن ان تختلف فى القيم
ويوجد لها نوعان المصفوفه احادية البعد او one-D والمصفوفه الثنائية البعد او

Tow - D

القاعدة العامه لتعريف المصفوفه الاحديه :

- 1 Array Definition : syntax
- 2 Data type [] reference name = new data type [size];

```

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int [] onearray = new int [5];
            Console.ReadKey();
        }
    }
}

```

The diagram shows a C# code snippet with several annotations in Arabic:

- نوع بيانات المصفوفه** (Type of the array) points to the declaration `int [] onearray`.
- اسم المصفوفه** (Name of the array) points to the identifier `onearray`.
- حجم المصفوفه** (Size of the array) points to the dimension `[5]`.

ومن الممكن ان تعطى قيمها ابتدائياً عند التعريف لكن يجب الاتزad القيمة عن حجم المصفوفة

```

class Program
{
    static void Main(string[] args)
    {
        int [] onearray = new int [5]{10,15,20,25,30};
        Console.ReadKey();
    }
}

```

القيم الابتدائية من نفس نوع
البيانات ولا تزيد عن حجم
المصفوفة

ومن الممكن ايضا تحديد حجم المصفوفة عن طريق القيم الابتدائية لكن ليست مستحبة
ومن الممكن اعطاء قيمها ابتدائياً للمصفوفة من المستخدم وقت تنفيذ البرنامج

 طريقة الاتصال مع المصفوفات الاحادية :

```

1 Write data to element in array :
2 Direct in code: array name [index of element] = value;
3 X [2] = 10;
4
5 write data from user on runtime:
6 X [2] = int.parse (Console.ReadLine ());
7
8 print element data :
9 Console.WriteLine(x [2]);
10
11 modifi on element data:
12 X [3] = x [3]*3;

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] x = new int[10]; يزيد الحلقه التكراريه عن ذلك سعه المصفوفه 10 لذلك لن
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine("enter no :{0}", i);
                x[i] = int.Parse(Console.ReadLine());
سوف يتم استلام بيانات المصفوفه من المستخدم وقت تنفيذ البرنامج
            }
            Console.ReadKey();
        }
    }
}

```

ومن المثال السابق سوف بتكميلة البرنامج لحساب اكبر قيمة واقل قيمة تم ادخالها من

المستخدم كما يلى :

```

    {
        static void Main(string[] args)
        {
            int i;
            int[] x = new int[10];
            for (i = 0; i < 10; i++)
            {
                Console.WriteLine("enter no :{0}", i);
                x[i] = int.Parse(Console.ReadLine());
            }
            int max = x[0]; هنا تم تعريف متغيرن بالقيمة الصغرى والعظمى
            int min = x[0]; واعطائهم القيمة المواجهة بالعنصر الاول من
            for (int j = 0; j < 10; j++) المصفوفه
            {
                if (x[j] < min) تم المقارنه العنصر الاول من المصفوفه بجميع العناصر الاخرى
                    min = x[j]; ونرى من فيهم الاصغر من الاخر حتى يتم ايجاد الاصغر فيهم
                if (x[j] > max) جميعاً ثم بعدها نفس العملية تكرر من فيهم الاكبر من الاخر حتى
                    max = x[j]; نجد الاكبر فيهم ثم نقوم بطباعة القيمتين على الشاشه
            }
            Console.WriteLine("the max no :{0}", max);
            Console.WriteLine("the min no :{0}", min);
            Console.ReadKey();
        }
    }

```

هذا مثال يوضح كيفية إيجاد العنصر الأكبر والعنصر الأصغر في مصفوفة.

الخطوات:

- تعريف مصفوفة `x` من نوع `int` بحجم 10.
- استخدام دالة `Console.WriteLine` لطلب إدخال العدد `i` (الindeks).
- استخدام دالة `int.Parse` لتحويل إدخال المستخدم إلى عدد صحيح وتخزينه في المصفوفة `x`.
- بيان متغير `max` بقيمة `x[0]` (العنصر الأول).
- بيان متغير `min` بقيمة `x[0]` (العنصر الأول).
- استخدام دالة `for` لـ `j` من 0 إلى 9.
- في كل دورة، التأكد من أن العنصر `x[j]` ليس أصغر من `min`، وإلا، تحديث `min` بـ `x[j]`.
- في كل دورة، التأكد من أن العنصر `x[j]` ليس أكبر من `max`، وإلا، تحديث `max` بـ `x[j]`.
- بعد انتهاء الدورات، طباعة `max` وـ `min`.
- إغلاق المصفوفة `x`.

: For EachLoop 

هناك نوع اخر مهم جدا من الحلقات التكراريه وغالبا ما يستخدم مع المصفوفات

حيث يستخدم في قراءة بيانات المصفوفه ولا يمكن استخدامه في الكتابه اليها او

تغير قيمها

القاعده العامه لها :

```

1  Foreach statement : syntax
2  For each <data type> <identifier> in array>
3  {Stmts} ;
4

```

```

static void Main(string[] args)
{
    int i;
    int[] x = new int[10];
    for (i = 0; i < 10; i++)
    {
        Console.WriteLine("enter no :{0}", i);
        x[i] = int.Parse(Console.ReadLine());
    }
    foreach (int m in x) استخدام الحلقة التكراريه
        Console.WriteLine(m); foreach
    في قراءة محتويات المصفوفه وطباعتها
    Console.ReadKey();
}

```

: Tow – D Array 

المصفوفات ثنائية الابعاد هى عباره عن نوع من المصفوفات يشبه الجداول

حيث يتكون من صفوف واعمده ويكون الخلايا وهذه الخلايا هى عناصر

المصفوفه

التصنيف العام لها :

```
1 2-D array : syntax
2 Data type [,] arrayreference = new data type [rows_no, columns_no];
3
```

For example:

```
1 int [,]A = new int [4,5] \\ decleration
2 initialize array
3
4
5 int [,] A = new int [1,2]
6 A[0,0] = 1
7 A[0,1] = 2;
8
9 int [,] A = new int [3,4]
10 A[0,0] = 1;
11 A[0,1] = 2;
12 A[0,2] = 3;
13 A[0,3] = 4;
14 ..
15 ..
16 ..
17 ..
18 A[2,3] = 12;
```

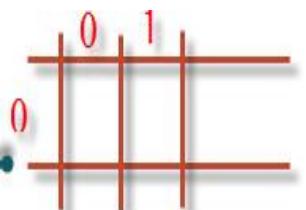
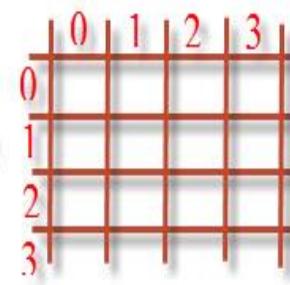
تعريف المصفوفه

رقم الصف

رقم العمود

اعطائها قيم ابتدائيه

عدد الاعمده X اذا تم ضرب عدد الصفوف تحصل على عدد عناصر المصفوفه

اعطاء بيانات للمصفوفه من المستخدم اثناء تنفيذ البرنامج فى هذا المثال :

```

static void Main(string[] args)
{
    int[,] x = new int[4, 5];
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            Console.WriteLine("enter value of element:(0,{1})", i, j);
            x[i, j] = int.Parse(Console.ReadLine());
        }
    }
    Console.ReadKey();
}

```

```

enter value of element:<0,0>
1
enter value of element:<0,1>
2
enter value of element:<0,2>
3
enter value of element:<0,3>
4
enter value of element:<0,4>
5
enter value of element:<1,0>
6
enter value of element:<1,1>
7
enter value of element:<1,2>
8
enter value of element:<1,3>
9
enter value of element:<1,4>
10
enter value of element:<2,0>
11
enter value of element:<2,1>
12
enter value of element:<2,2>
13
enter value of element:<2,3>
14
enter value of element:<2,4>
15
enter value of element:<3,0>
16
enter value of element:<3,1>
17
enter value of element:<3,2>
18
enter value of element:<3,3>
19

```

ولقراءة عناصر المصفوفه :

```
1 Read 2d array values
2
3 Console.WriteLine(A[2,3]);
```

```
20
<0,0> value is : 1
<0,1> value is : 2
<0,2> value is : 3
<0,3> value is : 4
<0,4> value is : 5
<1,0> value is : 6
<1,1> value is : 7
<1,2> value is : 8
<1,3> value is : 9
<1,4> value is : 10
<2,0> value is : 11
<2,1> value is : 12
<2,2> value is : 13
<2,3> value is : 14
<2,4> value is : 15
<3,0> value is : 16
<3,1> value is : 17
<3,2> value is : 18
<3,3> value is : 19
<3,4> value is : 20
```

طباعة عناصر المصفوفه
الثانويه

وللتعديل على بيانات المصفوفه الثنائيه :

```
1 Modify 2-d array
2
3 A[0,0] = A[0,0] * 2
```

: Array Class Properties And Methods

الخصائص والدوال التي يتضمنها الفصيل **Array**

Array Length – 1

ومنه تعرف طول المصفوفه تقديرك في حالة اذا كان عدد العناصر مجهولا لك

بمعنى انه عند تعريف المصفوفه فلن يتم تحديد طول معين بل سيقوم المستخدم

بأدخاله وقت التنفيذ

```

1 syntax : Arrayname.length
2 example : A.length
3
4 example in programming : تعريف متغير بطول المصفوفه
5 int arraylength; جعل المستخدم هو من
6 arraylength = int.parse(console.ReadLine()); يحدد الطول المناسب
7 int[] A = new int[arraylength];
8 for(i=0;i<A.length;i++)
9 {
10     A[i] = int.Parse(Console.ReadLine()); تحديد طول المصفوفه بالطول
11 } الذى ادخله المستخدم
12 Console.ReadKey();
13
استدعاء خاصية الطول من
الفصيل الذى يتبعه وهو
المصفوفه وبالتالي سوف
تأخذ القيمه التى ادخلها
ادخال بيانات الى العناصر

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int arraylength;
            arraylength = int.Parse(Console.ReadLine());
            int[] A = new int[arraylength];
            for (int i = 0; i < A.Length; i++)
            {
                A[i] = int.Parse(Console.ReadLine());
            }
            Console.ReadKey();
        }
    }
}

```

الدخل الطول = 4

ثم كتابة بيانات
إلى العناصر

: الخاصية Rank 2

والتي منها تعرف نوع المصفوفه اذا كانت ثنائية الابعاد او احادية البعد

واستدعائها يكون كالتالى :

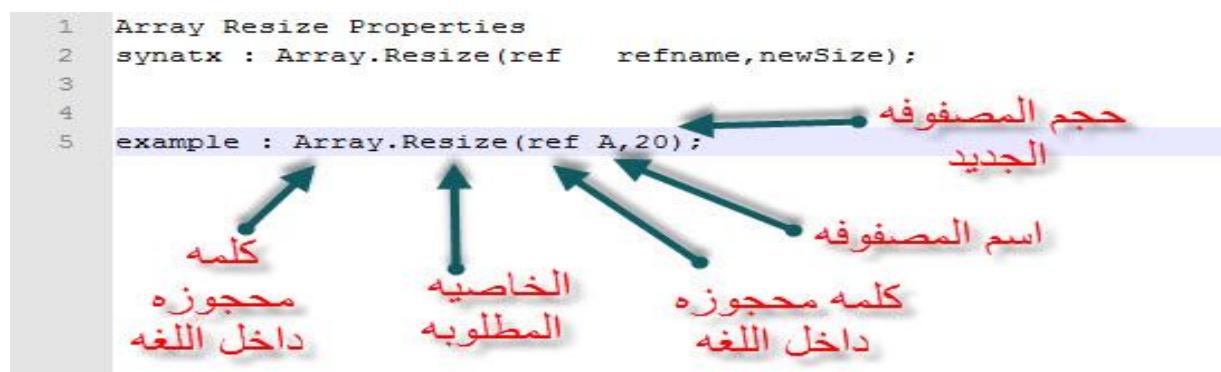
```

1 Rank Properties
2
3 syntax : arrayname . Rank

```

3 - الخاصية التي تسمح بـ تغيير حجم المصفوفة : Resize

تغيير الحجم مع الاحتفاظ بالقيم القديمة لها والقاعدہ العامہ لاستدعاھا

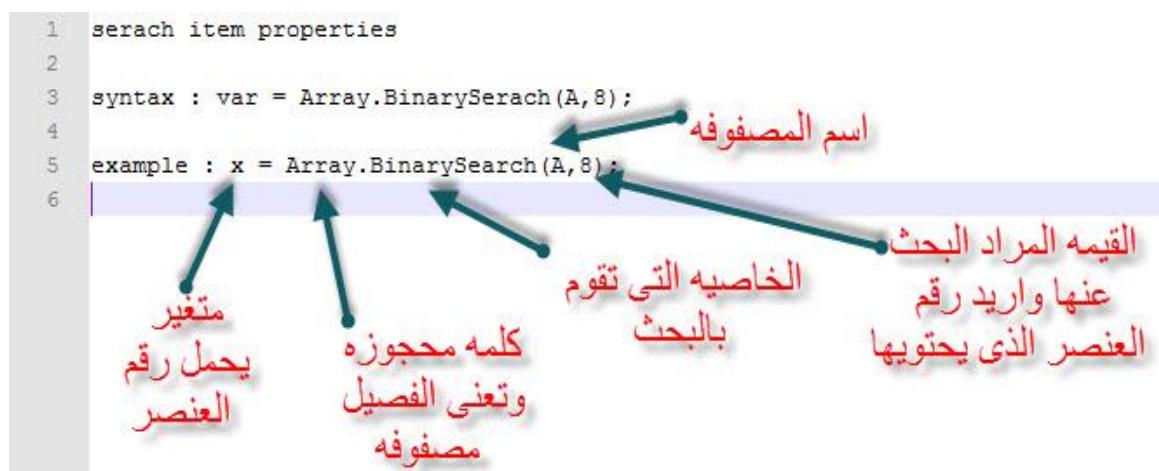


4 - خاصية البحث داخل عناصر المصفوفة : Binary Search

وتقوم هذه الخاصية بالبحث داخل عناصر المصفوفة عن عنصر يحتوى على قيمه معينه

لاستخدامه ولكن شرط ان تكون عناصر هذه المصفوفة مرتبه ترتيبا تناظريا او تصاعديا

القاعدہ العامہ لاستخدام الخاصية :



5 – الخاصيه التي تسمح بترتيب عناصر المصفوفه : Sort

القاعد़ه العامه لاستخدامها :

```

1 Sorting array elements
2
3 syntax : Array.Sort (refname) ;
4
5 example : Array.Sort (A) ;
6


```

الخاصيه
ترتيب
عناصر
اسم المصفوفه المراد
ترتيبها

6 – الداله Getting لاحضار قيمة العناصر والداله Setting لوضع قيم لعناصر

المصفوفه :

```

1 Getting and Setting method
2 Getting method : syntax : var = refname.GetValue(i) ;
3 example : z = A.GetValue(3);
4 here if you set the value in avriable
5 you must convert it like this : z = int.Parse(A.GetValue(3));
6


```

المتغير يحمل قيمة
العنصر
 تحويل القيمه
اسم المصفوفه
 استخدام الداله مع رقم العنصر


```

10 Setting method :syntax : refname.SetValue(data,index);
11 A.SetValue(20,2);


```

استخدام الداله لوضع قيمه
 القيمه المراد
 وضعها
 رقم العنصر

7 – دوال نسخ المصفوفات الى اخرى وهم نوعان

Copy – 1 وتقوم بنسخ جزء من مصفوفه الى اخرى

Copy To – 2 تقوم بنسخ كامل مصفوفه الى اخرى

قاعدة استخدامهم :

```

1 Copy and Copy to
2
3 Copy syntax : Array.Copy(Arrayref1,start1,Arrayref2,start2,no of element);
4
5 example : Array.Copy (A,5,B,6,3);  

6                               ↓          ↓          ↓          ↓          ↓  

7   الكلمة محووظة      الدالة      اسم المصفوفة     بداية النسخ  

8                               ↓          ↓          ↓          ↓          ↓  

9 Copy To syntax : Arrayref1.CopyTo(Arrayref2,start);
10
11 example : A.CopyTo(B,0);  

12                               ↓          ↓          ↓          ↓          ↓  

13   اسم                 الدالة      اسم المصفوفة     المراد نسخ اليها  

14   المصفوفة           ↓          ↓          ↓          ↓          ↓  

15   المراد نسخها       بالكامل
  
```

من اليسار اسم المصفوفه المراد نسخ العناصر منها - بداية النسخ - اسم المصفوفه المراد نسخ العناصر اليها - بداية وضع العناصر الاتيه - عدد العناصر المراد نسخها

8 – الداله Reverse

والتي تقوم بعكس عناصر المصفوفه اي عكس وضعية العناصر

طريقة استخدامها :

```

1 Reverse Function
2
3 syntax : Array . Reverse (A);
4                               ↓          ↓          ↓          ↓          ↓  

5   الكلمة محووظة      الدالة      اسم المصفوفة     المراد تنفيذ الامر  

6                               ↓          ↓          ↓          ↓          ↓  

7   الماء                 الماء      الماء             عليها
  
```

C# Methods

الدوال واستخدامها داخل السى شارب :

القاعدہ العامہ لاستخدام الدوال داخل لغۃ C# :

```

1 Method declaration in c#
2 syntax :
3 [Access_modifier] Return_type method_name (arguments)
4 {
5 //method body
6 }
```

هي عباره عن
معاملات
تستخدم لتحديد
سماحية
استخدام الدالة
داخل البرنامج
والنوع
الافتراضي لها
هو خاص

اهم ما يميز الدالة عن المتغيرات العاديہ
وهو تحديد بارمترات تستخدم لتحديد
نوع الدخل للدالة والتي تؤدى عليه
اسم يميز الدالة
البيانات
والافضل ان
التي تعود
يكون مرتبط
بها الدالة
بوظيفة الدالة

أنواع المعاملات او Access Modifiers

```

1 Private : ← لها فقط class يسمح هذا النوع بالتعامل مع الدالة او المتغير داخل
2
3
4 Public : ← يسمح هذا النوع بالتعامل مع الدالة او المتغير خارج class
5
6
7 Protected : ← يسمح هذا النوع بالتعامل مع لاداله او المتغير داخل المشروع
8   فقط namespace لديه او

```

- بعض الامثله على بناء الدوال وتعريفها داخل البرنامج :

```

1 examples :
2
3 int fact (int number); ← دالة لكن لها بارامتر واحد فقط
4 {method code}
5
6
7 int bar (); ← دالة ليس لها اي بارامترات
8 {method code}
9
10 void sum (int x, int y); ← دالة لها بارمتران لكن ليس لها قيمة عائده منها
11 {method code} ← بسبب المعرف
12

```

void حيث انه لا يسمح باي قيمة عائده من الدالة

هذه هي الثلاثة احتمالات التي تبني عليها الدالة داخل السى شارب وهناك احتمال رابع وهو نادر وهو ان الدالة لا تعود بقيمه وليس لها بارامترات ومن امثلة ذلك الدالة التي تستخدم في طباعة الخرج على الشاشه WriteLine()

بعض الامثله :

```

class Program
{
    void showdate()
    {
        DateTime thedate;
        thedate = DateTime.Now;
        Console.WriteLine(thedate);
    }

    static void Main(string[] args)
    {
        Program pro = new Program();
        pro.showdate();
        Console.ReadKey();
    }
}

```

داله لاظهار تاريخ اليوم ليس لها
بارامترات او قيمه عائده

ولاستدعى هنا يجب تخلص كائن جديد من
الفصيل التابع له ثم استدعائها بهذا الكائن

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program    داله لها بارامتران وتعود بقيمة المساحه
    {
        static int getarea(int height, int width)
        {
            return height * width;
        }

        static void Main(string[] args)    استدعاء الداله واعطائها قيمة
        {
            Console.WriteLine(getarea(4, 4));
            Console.ReadKey();
        }
    }
}

```

الخرج

ومن المثالين السابقين نريد ان نسلط الضوء على بعض الاساسيات الهامه جدا في

استخدام الدوال :

1 – ان استدعاء الدوال يكون على حسب المعرف لها او Access Modifier

وقد قمنا بشرح انواعه .

2 – ان كتابة الداله داخل الفصيل او Class قبل الداله الرئيسيه Main او بعدها ويكون

استدعائها من داخل الداله Main او من اي داله اخرى .

3 – ان استدعاء اي داله لا يكون الا عن طريق انشاء كائن جديد من الفصيل او Class

والذى تتبعه الداله او الذى تم انشائها بداخل كواحدة من عناصره الاساسيه ولا يكون

غير ذلك الا في حالة واحده وهى ان تعطى المعرف Static وهو الوحيد الذى يسمح

باستدعاء الداله بشكل منفرد دون الحاجه الى كائن المخلق من الفصيل ويفسر هذا وضع

هذا المعرف دائمأ قبل الداله الرئيسيه Main داخل اي برنامج فى السى شارب حتى

يتسى استدعائها من داخل البرنامج مباشرتا لان التنفيذ يبداء من عندها

4 – ان الداله او قد يكون لها اكثرب من بارامتر او لا يكون لديها اي بارامترات وقد تعود

بقيمه واحده او اكثرب من قيمه او لاتعود بقيم على الاطلاق لكن يجب ان تلاحظ الفرق

بين الحالتين في المثال التالي :

```

namespace ConsoleApplication1
{
    class Program
    {
        void showdate()
        {
            DateTime thedate;
            thedate = DateTime.Now;
            Console.WriteLine(thedate);
        }

        static int getarea(int height, int width) return
        {
            return height * width;
        }
    }
}

```

ان استخدام الكلمة التعریف void تمنعك من ان تكون الدالة لها قيمة عائده وتلاحظ هذا في عدم وجود الكلمة المحجوزه return

ان وجود نوع من انواع البيانات للدالة اذا لابد وان يكون لها قيمة عائده اى لابد من استخدام الكلمة المحجوزه return

ان الكلمتان void و return لا يجتمعان في تركيب دالة واحدة

5 – من الممكن اعطاء القيم التي تعود بها الدوال لمتغيرات لكن من نفس نوع البيانات

التي تعود به الدالة

6 – لابد من اعطاء كافة القيم للبارامترات التي تحملها الدالة داخل الكود ولا بد ان تكون

القيم من نفس نوع تلك البارامترات

- فى المثال التالي نقوم بعمل برنامج يقوم ببعض العمليات الرياضيه لكن

باستخدام الدوال :

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  namespace ConsoleApplication1
6  {
7      class Program
8      {
9          static double sum(int x, int y)
10         {
11             return x + y;
12         }
13         static double sub(int x, int y)
14         {
15             return x - y;
16         }
17         static double mul(int x, int y)
18         {
19             return x * y;
20         }
21         static void Main(string[] args)
22         {
23             int x, y;
24             Console.WriteLine("enter value of x and y");
25             x = int.Parse(Console.ReadLine());
26             y = int.Parse(Console.ReadLine());
27             Console.WriteLine(sum(x, y));
28             Console.WriteLine(sub(x, y));
29             Console.WriteLine(mul(x, y));
30             Console.ReadKey();
31         }
32     }
```



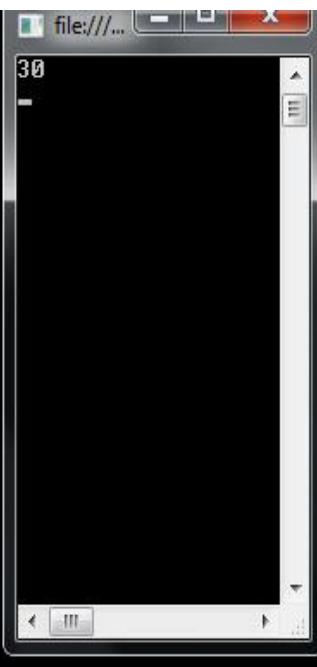
طرق اعطاء البارامترات الى الدوال : Passing Parameters to method

by value – 1

وهي ارسال القيمه الى البارامتر مباشرتا وتكون عباره عن نسخ الاساسيه

ونقلها الى مكان فى الذاكره وبالتالي هذا استغلا سيئ للذاكره لانه يأخذ مساحه

اكبر بدون داعى ومثال عليه :



```

using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static int sum(int x, int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10;
            int b = 20;
            Console.WriteLine(sum(a, b));
            Console.ReadLine();
        }
    }
}

```

by reference – 2

الطريقه الثانيه وهى اعطاء البارامترات عن طريق الاشاره الى مكان معين فى الذاكره

واليك المثال التالي :

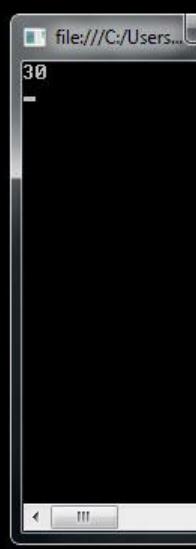
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static int sum(ref int x, ref int y)
        {
            int z = x + y;
            return z;
        }
        static void Main(string[] args)
        {
            int a = 10; مره اخرى عن استدعاء الداله واعطاء
            int b = 20; قيم للبارامترات
            int z = sum(ref a, ref b);
            Console.WriteLine(z);
            Console.ReadLine();
        }
    }
}

```

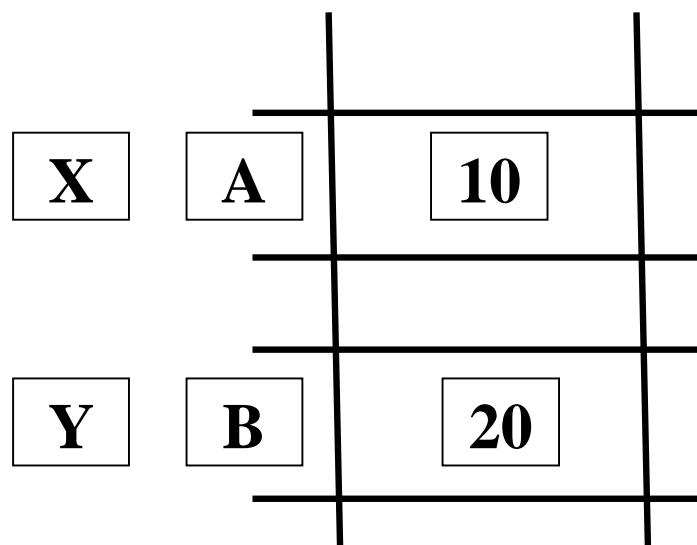
استخدام الكلمه الحجوزه قبل البارامترات
مره عند التعريف



ان استخدام هذه الطريقة هو الاستخدام الامثل للذاكره لانها كلا من البارامترات

والمتغيرات التي تمثلها تشير الى مكان واحد في الذاكره

لاحظ ان استخدام
REF
لابد من اعطاء قيم للبارامترات قبل
استدعاء الداله اي انك تدخل البيانات ثم
تقوم باستدعاء الداله لتجمعهم



by out – 3

هى نفس استخدام ref لكن الاختلاف فى انك تستدعي الداله اولا ثم تعطىها قيمة

ابتدائيه ولنى فى المثال التالى :

```
static int GetArea(int width, int height, out int prem)
{
    prem = (width * 2) + (height * 2);
    return width * height;
}

static void Main(string[] args)
{
    int area;
    int prem;
    area = GetArea(5, 10, out prem);
    Console.WriteLine("the area is :{0}&the prem is :{1}", area, prem);
    Console.ReadLine();
}
```

الطول والعرض أخذوا قيمة مباشرة لأنهم
وتم حساب المساحة بهم byvalue معرفين
لكن المحيط لم يأخذ قيم مباشرة بل تم حسابه
عن طريق القيم التي أعطيت له من الطول
والعرض

: C# Variable Scopes

النطاق الذى تستخدم فيه المتغيرات داخل اللغة :

وله ثلاثة انواع :

class level – 1

متغيرات يتم تعريفها داخل class ويتم استخدامها مع اى method اخرى

method level – 2

متغيرات خاصه ب method ولا تستطيع استخدامها مع اى method اخرى

nested level – 3

متغيرات تعرف داخل اى block من data ولا تستخدم خارج هذا block مثل الحلقات

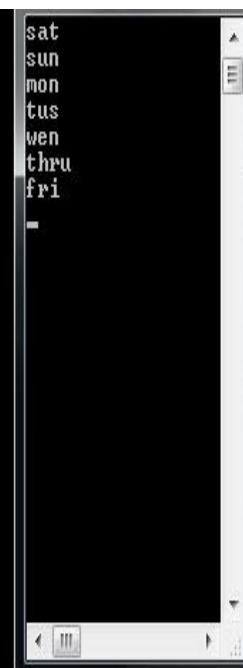
مثلا او الجمل الشرطية

: Passing Arrays as Parameters

Passing by value – 1

برنامج عمل array ويضع قيمه ابتدائيه ويعطيها الى method لكى تطبعها

على الشاشه وهنا سوف نعطي المصفوفه للداله كبار امتر



```

using System.Text;

namespace ConsoleApplication3
{
    class Program
    {
        static void printarray(string[] w)
        {
            for (int i = 0; i < w.Length; i++)
            {
                Console.Write(w[i]);
                Console.WriteLine();
            }
        }

        static void Main(string[] args)
        {
            string[] weekdays = new string[]
            { "sat", "sun", "mon", "tus", "wen", "thru", "fri" };
            printarray(weekdays);
            Console.ReadLine();
        }
    }
}

```

: Method overloading 

من الممكن بناء اكثربن نفس الاسم لعمل عدة وظائف تحت مسمى واحد
بشرط ان تختلف هذه الدوال فى عدد البارامترات او ان البارامترات تختلف نوع
البيانات المعروفة بها وتسمى هذه العملية Overloading او التحميل الذائد

```

class Program
{
    public int add(int a, int b)
    {
        return a + b;
    }
    public float add(float a, float b)
    {
        return a + b;
    }
    public int add(int a, int b, int c)
    {
        return a + b + c;
    }
    static void Main(string[] args)
    {
        Console.WriteLine(add(2, 3));
        Console.WriteLine(add(2.2, 3.3));
        Console.WriteLine(add(2, 3,4));
        Console.ReadLine();
    }
}

```

جميع الدوال بنفس الاسم لكنها
تختلف من ناحية تعريف
البارامترات ونوع بياناتها

Object Oriented Programming

- رؤيه فلسفية حول البرمجه الكائنية التوجه :

ان مفهوم البرمجه الكائنية التوجه لھو بالاساس يعتمد على النظره المجرده للحياة التي نعيشها ذلك بأن كل شئ في هذه الحياة عباره عن مجموعه من الكائنات ولھا صفاتها الخاصه ومميزاتها والشكل التي تتصرف به وتجد ان الطبيعه قد ربطت بين كل الاشكال المختلفه للحياة كل منھم له اسلوبه الخاص بالتعامل واسلوبه الخاص في الطريقة التي يتعامل بها مع الآخر وذلك الربط ادى في النهايه الى تكوين فضاء يحتوى هذه الكائنات جميعا لکى توجه الى فكرة العالم المتوازيه مبرمجه طبيعيا لأن تتعامل مع بعضها في تعاون برمجي كامل وبالتالي ارادوا تطبيق تلك الفكره والسير على خططاها في مفهوم البرمجه الكائنية التوجه فهو يعطى فكره مجرد للنظام ككل ثم يقسمه الى مجموعه من الكائنات التي لها خصائص تميزها عن غيرها ويوضع مع كل كائن الاساليب التي يتعامل بها مع غيره .

 مزايا البرمجة كائنية التوجه :

1 – سهولة متابعة وصيانة السوفت وير

2 – سرعة تطوير البرمجيات ومتانتها

Object Oriented Programming Concepts

Encapsulation – 1

ان مجموعه من العناصر لها نفس الخصائص ولها نفس السمات من الممكن ان تجتمع

داخل فصيل واحد ويسمى Class وهذا الفصيل هو الذى يمثلها بوجه عام لكن لا يتعامل

معها عن طريقه بل عن طريق كائن مخلق منه وهو Object

- اذا ما اردنا ايجاد امثاله من الحياه لدينا Class يمثل البشريه وهذا الفصيل له

جميع صفات البشريه واسلوب تعاملها مع غيرها والذى هو فصيل رئيسي من

منظومه اشمل وسوف يطلق عليها namespace وهذه المنظومه ممثله فى الكون

اذا اردنا انشاء كائن من فصيل البشريه ليكن الانسان الذى يحيا على الارض واى

انسان اذا يحمل جميع صفات وسلوكيات هذا الفصيل

ومن الممكن ان ننشئ فصيل اخر من فضاء الكون ولتكن فصيل للحيوان

ومن الممكن ان يكون هناك فصيل للنباتات واخر للجماد وهكذا

: Class Builder Features 

مالذى توفره لك Classes من مزايا داخل البرمجه كائنية التوجه ؟

1 – استطعت بذلك ان تقوم بتمثيل كل Object او عنصر داخل System

بمجموعه من Classes والتى تحتوى على كل التعاريف لهذا Object من

سمات وتصرفات

2 – استطعت حماية بياناتك الاساسيه لهذا Object من محاولة العبث بها او

حتى تغييرها والتى تتمثل فى المفاهيم الاتيه :

زايا التغليف : Encapsulation Features – 1

```

1 Encapsulation Builder :
2 يبني الفضيل بواسطة الكلمه الممحوزه التى تعرفه ثم يعطي اسم ثابت له
3 class class_name
4 {
5 //attrbuites// تكتب هنا المتغيرات او
6 int var1; ما يسمى بالمعلومات
7 int var2; التي تخص هذا الفضيل
8 string var3;
9
10 public string function_name(parameters)
11 {
12 //method body// توضع هنا الدوال التى تعبر عن تصرفات الفضيل والتى تكون ثابته خلال حياة النظام
13 }
14
15 public data_type property_name
16 {
17 get
18 { توضع هنا الخواص التى يعتمد عليها
19 return attrbuites_value; الفضيل والتى تعطى قيمًا متغيره وهذه
20 } الخواص هي التى تعطيك التحكم فى
21 set قيم لمعلومات الفضيل عند بعد
22 {
23 attrbuites = value;
24 }
25 }
26 }
```

مزايا التغليف :

State Retention – 1

الابقاء على حالة وتصرفات العناصر او Object داخل System وذلك يمثل فى الجزء الذى يعطى له وهو Methods فقط انا اعطى لها القيم وهى تقوم بالتصرف الثابت لها

Data Hiding – 2

يتم عزل البيانات والتى تكون ممثلاً فى Attributes للعنصر Object داخل النظام بعيداً عن أي تغيير

ويبقى هنا السؤال كيف لى ان استخدام هذا الفصيل الضخم والمليء بعده

عناصر وكيف لى ان استخدام تلك العناصر من خارج هذا الفصيل ؟

وتاتى الاجابة بأن بناء الفصيل شيء واستخدام محتوياته شيء آخر بمعنى ان الفصيل الذى يسمى البشرية هو فصيل عام يطلق على بنى البشر لكن يجب ان يجسد فى عنصر معين لكي تستطيع ان تتعامل معه وهذا العنصر الذى يمثله هو الانسان وبالتالي سوف يتم انشاء عنصر من كل فصيل لكي تستطيع ان تستخد كافية عناصره

انتهت النظره الفلسفيه لناتى الى تدعيمها بأمثله عمليه :

سماحية استخدام عناصر Class داخل البرنامج :

تحتوي لغة C# على خمسة أنواع منهم :

```

1 private : وتعطى سماحية استخدام عناصر الفصيل داخل الفصيل فقط
2
3
4 protected: داخل الفصيل او فصيل مثني منه
5
6 يستخدم في اي مكان داخل المشروع عامة
7 internal:
8
9
10 protected internal : استخدامه داخل المشروع او مشروع مثني منه
11
12
13 public: يتعامل مع عناصر الفصيل من اي مكان داخل نفس المشروع او داخل
      فضاء الاسماء المضمن بداخله

```

بعض الامثله على بناء Classes داخل C# :

```

using System.Text;

namespace ConsoleApplication3
{
    class EX1
    {
        //تعريف الفصيل باسم معين
        //تعريف المتغيرات من نوع خاص اي لا يمكن استخدامها الا داخل الفصيل فقط
        private int x;
        private int y;

        public void setdata(int a, int b)
        {
            x = a; //تعريف دالة من نوع عام والتي يمكن استدعائهما
            y = b; //خارج الفصيل والتي تعطى قيم للمتغيرات
        }

        public int sum()
        {
            return x + y; //تعريف دالة للجمع
        }

        public float avg()
        {
            return sum() / 2; //تعريف دالة للمتوسط الحسابي
        }
    }

    class Program
    {
    }
}

```

والآن كيف لنا ان نستخدم عناصر هذا الفصيل داخل البرمجه ؟

والجواب هو ان نقوم بانشاء كائن يتبع هذا الفصيل وبالتالي فهو يستطيع استدعاء كافة

عناصره كالتالى : القاعده البرمجيه

```

1 create object from class
2
3 class_name obj_name = new class_name; //اسم الفصيل مره اخرى
4

```

اسم الفصيل مره اخرى

اسم الفصيل الذي تريد انشاء كائن منه

كلمه محجوزه تدل على انشاء الكائن

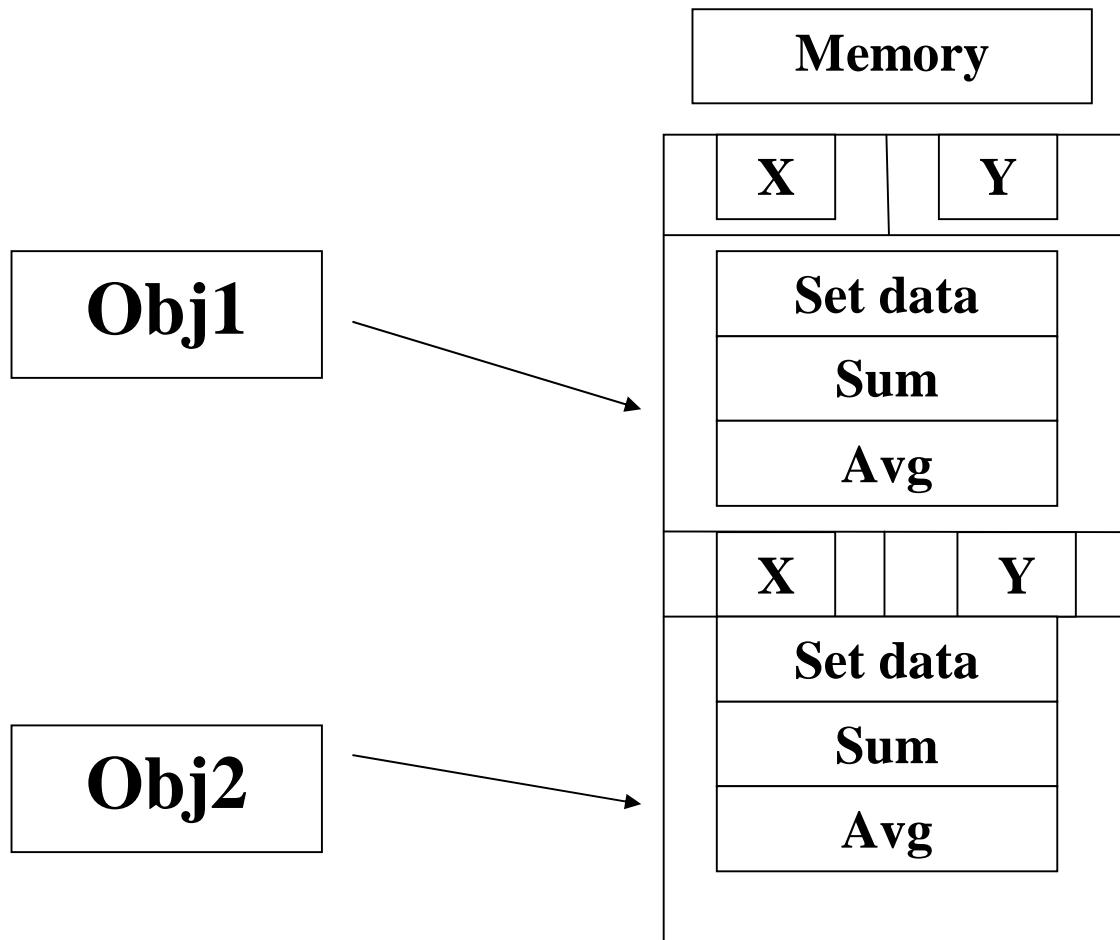
اسم الكائن الذي تريد انشاؤه

```

class Program
{
    static void Main(string[] args)
    {
        EX1 obj1 = new EX1();
        EX1 obj2 = new EX1();
        Console.ReadLine();
    }
}

```

انشاء الكائن وتعريفه يكون داخل الفضيل الرئيسي داخل الدالة الرئيسية ومن الممكن تعريف اكثر من كائن من نفس الفضيل ولهذا ميزة كبيرة وهي ان حناصر الكائنين لا يتأثروا ببعضهم البعض رغم انهم ينتمون لفضيل واحد لكن لكل منهم مكانه في الذاكرة



توصيات هامة : يفضل ان تكون المتغيرات داخل الفضيل تعطى المعرف Private

حتى لا يتم العبث بها او تغييرها .

يفضل ان تعطى الدوال داخل الفضيل المعرف Public حتى تستطيع استخدامها خارج الفضيل .

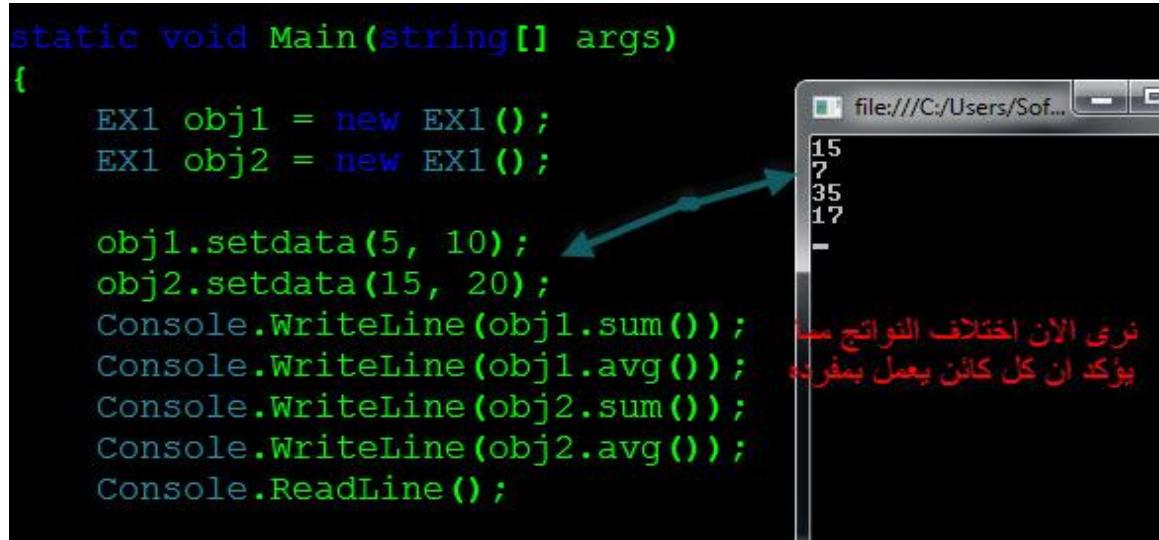
 كيفية الاتصال باعضاء class باستخدام الكائن الجديد:

القاعدہ البرمجیہ :

1 Accessing object members:	2	3 object_name . member_name;	
		اسم الكائن	اسم العضو

سنقوم في المثال السابق باعطاء قيم لعناصر الكائنين الذي تم تعريفهم من نفس الفضيل

ونرى النتيجه لدالتين الجمع والمتوسط الحسابي :



The screenshot shows a C# console application. On the left, the code defines two objects of type EX1 and calls their setdata methods with different parameters. On the right, the console window displays the results of calling sum and avg on each object separately.

```

static void Main(string[] args)
{
    EX1 obj1 = new EX1();
    EX1 obj2 = new EX1();

    obj1.setdata(5, 10);
    obj2.setdata(15, 20);

    Console.WriteLine(obj1.sum());
    Console.WriteLine(obj1.avg());
    Console.WriteLine(obj2.sum());
    Console.WriteLine(obj2.avg());
    Console.ReadLine();
}

```

النتيجة في консоль:

```

file:///C:/Users/Sof...
15
7
35
17

```

نرى الان اختلاف التواليج سـ
يؤكـد ان كل كـائن يـعمل بمفرده

هناك دالتين رئيستين تعطى قيمًا ثابتة Private members وهذا يحدث مع

اي لغه تستخدم البرمجه كائنية التوجه حيث تستخدم دالتين :

class : لاعطاء القيم الابتدائية لاعضاء Setter – 1

: لقراءة البيانات واظهارها على الشاشه Getter – 2

بعض الامثله على ذلك :

```

1 private string name;
2
3 //getter function :
4 public string getname()
5 {
6     return name;
7 }
8 //setter function
9 public void setname(string thename)
10 {
11     name = thename;
12 }
```

هذه الدوال من الممكن استخدامها داخل السبي
شارب لكن اللげ لها اسلوب متتطور وهو استخدام
الخواص او
properties
والتي تقوم بنفس المهمه بشكل اكثر كفائه ويتوافق
مع النظريه الكائنية التوجه

: Properties

Read Only – 1

Write only – 2

Read / Write – 3

القاعدہ العامہ لتعريفہا داخل Classes :

syntax of properties:

```

اسم الخاصية نوع البيانات التي تعود بها نوع الاتصال بالخاصية
<access modifiers><return data type><name of property>

{
    كلمه ثابتة معرفه داخل اللغه وتحل محل الدالة
    get           getter
    {
        اسماء المتغيرات التي سوف تعود
        return <some private field>;   الخاصية بالقيمة التي تعطى لها
    }
    كلمه محجوزه وتحل محل الدالة
    set           setter
    <some private field> = value;
    اسماء المتغيرات التي تريد
    {
        اعطاء القيمة لها عن طريق
        الخاصية وهي نفسها الموجودة
        في الجزء الخاص ب
        get
}

```

بعض التوصيات :

الخواص التي تبني داخل classes ليس لها اى بارامترات
 اسم الخاصية لابد ان يكون هو نفسه اسم المتغير التي تتعامل معه مع الفارق بان يكون
 اول حرف من اسم الخاصية capital حتى يتميز عن المتغير

ولنرى بعض الامثله على بناء الخواص :

```

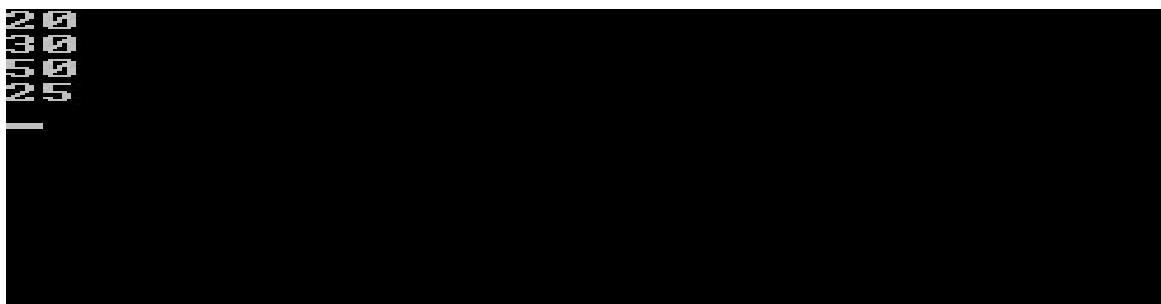
class example
{
    int num1, num2;
    public int Num1
    {get{ return num1; }
     set{ num1 = value; } }
    public int Num2
    {
        get { return num2; }
        set { num2 = value; }
    }
    public int sum()
    { return num1 + num2; }
    public float avg()
    { return sum() / 2; }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        example obj1 = new example();
        obj1.Num1 = 20; اعطاء القيم عن طريق الخواص
        obj1.Num2 = 30; اظهار القيم عن طريق الخواص
        Console.WriteLine(obj1.Num1);
        Console.WriteLine(obj1.Num2);
        Console.WriteLine(obj1.sum()); اظهار نتائج الدوال
        Console.WriteLine(obj1.avg());
        Console.ReadLine();
    }
}

```



لاحظ ان القيم اعطيت للمتغيرات عن طريق الكلمه set داخل الخاصيه وتم اظهار

البيانات عن طريق الكلمه get

: Static Member

ان هذا المعرف له ميزه خاصه جدا داخل السى شارب حيث انه يقوم بمشاركة العضو

الذى يعرف بواسطته بين جميع الكائنات التى تعرف من نفس الفصيل ويلزم لتعريف



ولنرى مثال عام :

```

{
    class student
    {
       تعريف العضو المشارك عليه من جميع
        الكائنات
        public static int phonenumber;
        public int idnumber;
        public string name;
    }
    class Program
    {
        static void Main(string[] args)
        {
            student st1 = new student();
            student st2 = new student();
            st1.idnumber = 3;
            st1.name = "ahmed";   استدعاء جميع الاعضاء بالفأعدة الاساسية
            st2.idnumber = 4;      وهي باستخدام الكائنات
            st2.name = "kamel";
            student.phonenumber = 492067;
            عد ارقم التليفون فلقد تم استدعاءه عن طريق الفصيل
            مباشرتنا
        }
    }
}

```

ان اى static member ينتمي الى الفصيل مباشرتا وليس الكائن وهذه ميزة

هامه جدا داخل السى شارب

دوال البناء والهدم داخل السى شارب :

1 – دالة البناء : Constructor

هي دالة خاصه باى Class وتستخدم لوضع القيم الابتدائيه لعناصره

خصائصها :

1 – تأخذ نفس اسم Class

2 – ليس لها قيم مسترجعه

3 – يتم تنفيذ دالة البناء تلقائيا مع كل تعريف لكائن جديد من Class

4 – الكود بداخله يستخدم لوضع Data لاى عضو داخل Class

امثله على ذلك :

```
class example
{
    private int x;
    private int y;
    public example()
    {
        x = 50;
        y = 100;
    }
    public double sum()
    {
        return x + y;
    }
}
class Program
{
    static void Main(string[] args)
    {
        example ex = new example();
        Console.WriteLine(ex.sum());
        Console.ReadLine();
    }
}
```

تعريف دالة البناء والتى تعطى قيمها ابتدائيه
للمتغيرات داخل الفصل ويبكون اسمها نفس اسم الفصل

بناء دالة اخرى لاظهار محتوى القيم
الابتدائيه

150

نتائج الجمع 150 مما يدل على أن
القيمة الابتدائية المعطاة عن طريق
دالة البناء صحيحة

في حالة عدم بناء هذه الدالة داخل اي فصيل فسيتم وضع قيمه ابتدائيه لاعضائه

كالتالى :

String = null

Boolean = false

Int = 0

Float = 0.0

وذلك لأن دالة البناء يتم بنائها تلقائيا داخل اي فصيل وهذه ميزة هامة جدا داخل

السى شارب حيث انه يتم تخزين هذه القيم داخل الذاكرة منعا لتخزين قيم

عشوانئه وبالتالي فانت تعرف مسبقا ما الذى يوجد داخل عنوانين الذاكرة بالنسبة

للمتغيرات حتى ولو لم تعطيها قيم ابتدائيه ونستطيع التأكيد بالمثال التالي :

```

class example
{
    private int x;
    private float y;
    private double z;
    private string name;
    private Boolean c;
    public example()
    {
        Console.WriteLine(x);
        Console.WriteLine(y);
        Console.WriteLine(z);
        Console.WriteLine(name);
        Console.WriteLine(c);
    }
}

class Program
{
    static void Main(string[] args)
    {
        example ex = new example();
        Console.ReadLine();
    }
}

```

فيما يلى انواع المتغيرات والتى لن يتم اعطاء
قيمة ابتدائية لها عن طريق دالة البناء فقط
سنقوم باظهار القيم الافتراضية لها

هذا فقط ننشئ كائن جديد وسيقوم
باستدعاء دالة البناء تلقائياً

E
E
E

False

هذه هي القيم الافتراضية
لكل نوع من المتغيرات في
حالة عدم وضع اي قيمة لها

انواع دالة البناء داخل السى شارب :

: Without Parameter – 1

وتستخدم فى وضع قيم ابتدائيه ثابته داخل اى كائن وتعرف بنفس اسم Class وفي

هذا النوع يكون لجميع عناصر الفصيل نفس القيم مع تعريف كائن جديد من نفس

الفصيل

: With Parameter – 2

وتستخدم لوضع قيم ابتدائيه مختلفه داخل الكائن وفي هذا النوع يكون هناك قيم

مختلفه لعناصر الفصيل على حسب البارامترات

```

class example
{
    private int num1;
    private int num2;
    //constructor with parameter
    public example(int a,int b)
    {
        num1 = a;      دالة البناء يوجد بارامترات لها
        num2 = b;
    }
    //constructor without parameter
    public example()
    {
        num1 = 50;      دالة البناء في حالة عدم وجود بارامترات
        num2 = 100;    واستخدام القيم الابتدائية
    }
}
class Program
{
    static void Main(string[] args)   الان الدالة التي لها احقيه في التنفيذ تكون على حسب تعريف الكائن
    {
        example ex1 = new example();   النوع الثاني بدون
        example ex2 = new example(20, 30);  بارامترات
        Console.ReadLine();
    }
}

```

النوع الاول مع وجود البارامترات
النوع الثاني بدون بارامترات

مثال شامل على استخدام Classes في البرمجة :

نريد عمل مثال باستخدام Classes وذلك لحساب المساحات لثلاثة اشكال مختلفه

المربع - المستطيل - المثلث على ان يتم الاتي

1 - تعريف المتغيرات اللازمه لايجاد المساحات

2 - بناء دالة لاعطاء القيم لهذه المتغيرات

3 - بناء دالة لحساب المساحات

```
class square
{int lenght;
    public void setvalue(int a)
    { lenght = a; }
    public int area()
    { return lenght * lenght; }}
class rectangle
{ int wedith, lenght;
    public void setvalue(int a, int b)
    { wedith = a;
        lenght = b; }
    public double area()
    { return wedith * lenght; }}
class triangle
{ int lenght,tbase;
    public void setvalue(int a,int b)
    { lenght = a;
        tbase = b; }
    public double area()
    { return (lenght * tbase) / 2; }}
```

```

class Program
{
    static void Main(string[] args)
    {
        square sqr = new square();
        triangle tr = new triangle();
        rectangle rect = new rectangle();
        sqr.setvalue(5);
        tr.setvalue(3, 4);
        rect.setvalue(3, 4);
        Console.WriteLine(sqr.area());
        Console.WriteLine(tr.area());
        Console.WriteLine(rect.area());
        Console.ReadLine();
    }
}

```



لكن هناك طرق اكثراً احترافية لكتابة مثل هذه البرامج وذلك لأنه تم تكرار نفس المتغيرات ونفس الدوال في جميع Classes ولتفادي ذلك يجب أن تتعرف على شيء هام جداً يسمى الوراثة داخل البرمجة كائنية التوجه .

Inheritance

ان الوراثه هى عباره عن اشتقاق فصيل من اخر تحمل نفس خصائصها وهذه ميزة
هامه جدا داخل البرمجه كائنية التوجه وسبق لنا ان شرحت هذا الجزء فى كتاب هندسة
البرمجيات الخاص بيالجزء الثالث منه
يسمى Class الرئيسي بعده اسماء :

Original class – base class – parent class – super class
ويسمى المشتق بعده اسماء :

New class – derived class – child class – sub class
ان الوراثه لها ميزتين :

1 – انك تستطيع اشتقاق فصيل من اخر يرث جميع العناصر الموجوده فيه وبالتالي
توفر على نفسك عناي اعادة بعض الاكواد يتم تكرارها فى البرنامج كمثال المساحات
السابق

2 – انك تستطيع الاضافه داخل الفصيل المشتق الى عناصر الفصيل الرئيسي

القاعدہ البرمجیہ للاشتقاء داخل السی شارب :

```

1 Creating derived class from another
2
3 Class derived_classname : base_class name
4 {
5
6 }
```

كلمه معرفه داخل اللغة
وتعنى بناء الفصيل

اسم الفصيل المشتقة

اسم الفصيل الاساسي

يوجد نوعان من الاشتقاء :

- 1 – النوع الاول انك تشقق فصيل جديد يحمل جميع عناصر الفصيل الرئیسی دون اى اضافات
- 2 – النوع الثاني انك تستطیع ان تضيف خصائص اخری للفصيل المشقق تطبيق الوراثه على مثل المساحات سوف نبني فصيل رئیسی يسمی الشکل او ونشتق منه جميع Classes Polygon الاخری

```

class polygon
{protected int w, l;
 public void setvalue(int a, int b)
 { l = a;
   w = b; }

class square : polygon
{public int area()
 { return l * w; }}

class rectangle : polygon
{ public double area()
 { return l * w; }}

class triangle : polygon
{ public double area()
 { return (l * w) / 2; }}

```

لقد بناء الفصيل الاساسي والذى يسمى الشكل ويشتغل جميع الفصائل منه مع تغير الدالة التي تحسب المساحة في كل شكل على حدى

لاحظ فى هذا المثال استخدمنا فى تعريف المتغيرات المعرف Protected وليس Private والسبب لكى تستطيع استخدام المتغير من فصيل مشتق وذكرنا هذا من قبل

تعديل على بعض الخصائص داخل الفضيل الرئيسي واستخدام قاعدة :

Overriding Base Class Functionality

تستطيع التعديل على بعض الخصائص التي كانت موجودة داخل الفضيل الاب وذلك باستخدام الكلمة الثابتة داخل السى شارب Overriding واستخدام اسم الدالة التي تريد التغيير فيها او ما يسمى Match Signature

وحتى تكتمل قاعدة التغيير وتكون صحيحة فعند بناء الدالة في الفضيل الاساسى والتى تنوى التغيير فيها مستقبليا عند استيقاف فضائل اخرى من هذا الفضيل فيجب ان تعطى المعرف

Virtual

مثال على ذلك في برنامج المساحات :

```

class polygon
{protected int w, l;
 public virtual void setvalue(int a, int b)
 { l = a;
   w = b; } }
class squar : polygon
{
 public override void setvalue(int a, int b)
 {
   l = a;
   w = a;
 }
 public int area()
 { return l * w; }}

```

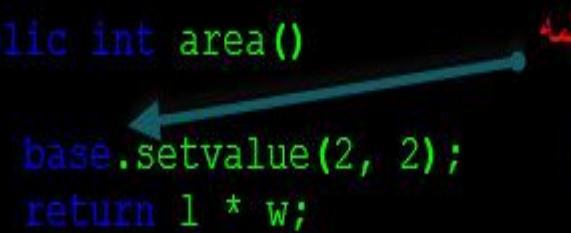
اعطيناها المعرف Virtual
عند تعريفها داخل الفضيل الاساسى

و عند التعديل اعطيناها المعرف Override

هناك ميزة أخرى داخل الاشتراق وهي أنك تستطيع استدعاء الدوال في الفصيل 

الأساسى داخل الفصيل المشتق وذلك باستخدام الكلمة المعرفه داخل اللغة Base

```
class polygon
{protected int w, l;
 public void setvalue(int a, int b)
 { l = a;
   w = b; }
class squar : polygon
{
    public int area()
    {
        base.setvalue(2, 2);
        return l * w;
    }
}
```



استدعاء الدالة بواسطة الكلمة
Base

داخل الفصيل المشتق

: Multi – level Hierarchies 

عملية الاشتراق المتابع لفصيل من اخر بمعنى ان عملية اشتراق تمت من فصيل

رئيسي تلتها عملية اشتراق من الفصيل المشتق وهكذا مما يعطيها شكل هرمى

ولاحظ انك فى كل مره تشقق فيها فصيل من الآخر تستطيع ان تعدل او تضيف

خصائص جديدة ويجب ان تلاحظ ان التعديلات لها الاولويه فى التنفيذ عن

الخصائص الأساسية

```

class polygon
{protected int w, l;
    public void setvalue(int a, int b)
    { l = a;
        w = b; } }
class squar : polygon
{ public int area()
    { base.setvalue(2, 2);
        return l * w; } }
class asquar : squar
{ public int area()
    { base.setvalue(3, 3);
        return l * w; } }
class bsquar : asquar
{public int area()
    { base.setvalue(4, 4);
        return l * w;} }

```

تابع علیات الاشتراق المترج

: Preventing Inheritance

منع الوراثه او الاشتراق داخل البرمجه كائنية التوجه حفاظا على فصيل معين من الاشتراق وتحدد كثيرا مع Classes المبنيه داخل السى شارب اساسا والتى قامت شركة مايكروسوفت ببرمجه حفاظا على حقوق ملكية الكود وتسمح باشتراق بعض الفصائل الاخرى

القاعد البرمجية لمنع اشتغال Class :

```
1 preventing inheritance
```

```
2
```

```
3 Sealed Class Class_name
```

```
4 {
```

```
5 }
```

```
6 }  
الكلمة المعرفة داخل اللغة  
والتي تمنع وراثة الفصيل
```

اسم الفصيل الذي تريده منعه من الوراثة
الكلمة المعرفة والتي تقوم ببناء
الفصيل

ولاحظ ان هذه العملية تتم عند بناء الفصيل الرئيسي قبل ان يشتق

```
{
    sealed class polygon
    {
        protected int w, l;
        public void setvalue(int a, int b)
        {
            l = a;
            w = b;
        }
    }

    class square:po
    class Program
    {
        static void Main()
        {
            System.EntryPointNotFoundException ex = new System.EntryPointNotFoundException();
            ex.Message
        }
    }
}
```

لم يعطيك الفرصة لاستئثار الفصيل الاساسي

Polymorphism

ان مفهوم Polymorphism داخلى السى شارب هو تماماً كمفهوم التعديل على بعض خصائص الفضيل الاساسى وذلك باستخدام المعرف Override ومن الممكن ان تقوم ببناء Class الاساسى دون عمل تكويده ثم بعد ذلك تعديل و تقوم بعمل الاكتوار فى الفضائل المشتقة

Abstract Class

و معناه Class بدون كود وهو مجرد Class عادى لكن دون سطر كدو واحد والهدف

- 1 – تعریف نسخه اصلیه من Class على انها Abstract وهي کلمه محجوزه واستعمال کلمة Partial معها داخلى السى شارب
- 2 – جميع الدوال والخواص التي تكتب بداخله يجب ان تعطى نفس المعرف وهو Abstract ولا يجوز تعریف اي متغيرات بداخله او انشاء كائن منه فقط تعریف الدوال والخواص

لدينا المثال التالي :

```
abstract class definition

abstract partial class car
{
    public abstract int calc(int fuel);
```



من الملاحظ ان الدالة لم يكن لها كود لكن
عند اشتقاق فصيل اخر منه فان المبرمج
سيكون مجبوا على عمل تكويده لهذه الدالة

```
abstract partial class House
{
    public abstract int roomnumber { set; get; }
    public abstract string label { set; get; }
    public abstract double budgetinmonth { set; get; }
    public abstract int roomnumber(int roonmuber);
    public abstract string houselabel(string label);
    public abstract double housebudget(double budget);
}
```

من الملاحظ ان جميع اعضاء الفصيل اخزو نفس المعرف ولا يوجد اي متغيرات
بل خواص ودوال فقط ولا يوجد اي تكويده لهم فقط تعريف لكل منهم

```
class Program
{
    static void Main(string[] args)
    {
    }
}
```

لترى الان اشتراق class جديد من السابق ونرى ماذا يكون شكله :

```

}
class myhouse : House
{
    int roomnumber;
    string label;
    double budgetinmonth;
    public override int Roomnumber
    {
        get{ return roomnumber; }
        set{ roomnumber = value; }
    }
    public override string Label
    {
        get { return label; }
        set { label = value; }
    }
}

```

من الملاحظ في الفصل السابق انه تم تعریف المتغيرات رقم كتبة کود للخواص وبالامكان ايضا كتابة الكود للدوال ولاحظ ايضا ان استخدمنا المعرف `override`

وناك للتعديل على اعضاء الفصل الاساسى مما يدعم نظرية `polymorphism`

التي تحدثنا عنها سابقا يمكن الان انشاء كائن من هذا الفصل السابق والتصرف به داخل البرنامج كما تعلمنا

Interface

تصميم الواجهات داخل الكود :

وهي بديل عن تعدد الوراثه الهرميه كما انه يتميز بعمل نسخه منه بالوراثه لكن

الاضافه على الفصائل المشتقه تكون بالتعديل على اعضاء Interface

كما انه يمكن عمل تكوييد لاكثر من Interface معا والشكل العام له يكون كالتالى :

interface definition

```
Access_modifier interface interface_name
{
    functions; المعرف
    properties; كلمه محجوزه
    without any implementation
}
اسم الواجهه
```

ثم تعريف جميع اعضائه لكن دون
تكوييد لها ولاحظ ايضا لا توجد
متغيرات

ولدينا المثال التالى :

```

interface IPPoint
{
    int x
        { set; get; }
    int y
        { set; get; } }

class mypoint
{
    private int myx;
    private int myy;
    public int x
    {
        set { x = value; }
        get { return x; }
    }
    public int y
    {
        set { y = value; }
        get { return y; }
    }
}

```

بناء الواجهات دون اى تكويid لها
ودون اى معرفات محددة

اشتقاق فصيل منها ثم عمل التكويid اللازم
اى انه يمكن اشتقاق فصيل من الواجهه اى
انها تقوم بعمل الفصيل

ملحوظه هامه : لا يمكن انشاء كائن من الواجهات او Interface لكن يمكن اشتقاق

فصائل منها وبالتالي انشاء الكائنات من classes المشتقه

لا يمكن تعريف المتغيرات داخل interface بدلًا منها خواص ودوال والمتغيرات

تعرف داخل classes المشتقه

ان عمل abstract class بديل عن الوراثه الهرميه وتشبه فى عملها interface

Types of Errors

: syntax error – 1

هذا هو الاسهل اكتشافه فور كتابة الكود والسبب ان compiler يعترض عليها لانها

غير ملائمه للقواعد التي بنيت عليها اللغة او التي تعمل بها داخل لغه معينه

: Logical errors – 2

الاخطاء المنطقية وهذه الانواع تظهر عند التنفيذ يكون الكود فيها صحيحاً لكن سير

البرنامج غير منطقى كالقسمه على الصفر مثلاً ويسمى هذا النوع Exception

او الاستثناءات

: Bugs – 3

هذه هو النوع الاشهر ولا يوجد برنامج تقريباً يخلو منه وهي نسيان حذف متغير من

الذاكره مثلاً عند غلق البرنامج ولذلك تكون هناك عدة نسخ تجريبية لاصطياد مثل هذه

الاخطاء

الشكل العام بالنسبة Exception داخل الكود :

```

try           ← كلمه محفوظه
{
    يوضع الكود الاصلي هنا
}
Catch(exception) ← كلمه معرفه ويوضعتعريف لاستثناء بين الفورين
{
    كمنغير مثل
    Exception e
رسالة الخطأ التي تزيد اظهارها في
}
حاله حدوث الخطأ
Finally ← كلمه معرفه
{
    ماذا تزيد ان تفعل هنا سواء ظهر الخطأ او لم يظهر
    وعادة ما يكتب هنا اكواد تقوم بحذف المتغيرات من
    الذاكرة
}
what you want if error or no error;

```

Collections

هي عباره عن مخزن او Container للبيانات ويوفر لك مزايا لا توفرها لك

مثيلتها من Array

ان Collections الذى يحتوى فصيل ArrayList هو namespace

واستدعائه يكون كالتالى :

```
System.Collections;
```

استدعاء فصيل التجمعات

ويعد فئة ArrayList هي اهم class فيها وتعرف على انها قائمه من Array لها نفس سمات المصفوفات من حيث تخزين مجموعه من البيانات دفعه واحده كالمتغيرات المركبه ولها ايضا مزايا القوائم او List داخل لغة البرمجه فهى اداه اقوى من عمل المصفوفات

طريقة استخدامه داخل السى شارب :

1 – يجب تعريف كائن منه لاستخدام جميع مزايا class

create object from ArrayList class: انشاء كائن من الفصيل
`ArrayList carlist = new ArrayList(); ArrayList`

you can add object : تستطيع اضافة كائنات بواسطة دالة الاضافه
`carlist.Add(temp);`

وستطيع عمل ادراج في جزء معين داخل المصفوفه بواسطة دالة الادراج
`carlist.Insert(temp);`

وستطيع مسح جميع عناصر القائمه دفعه واحدة
`carlist.Clear();`

وستطيع ازالة عنصر معين برقمه او ازالة الكائن
`carlist.RemoveAt(4);` المخزن فيها

وستطيع معرفة عدد عناصرها باستخدام دالة العد
`carlist.Count;`

وستطيع تحويلها الى مصفوفه عاديه
`ToArray`

وستطيع عكس ترتيب عناصرها
`Reverse`

وستطيع اجراء عملية البحث عن عنصر معين بداخلها
`Indexof(temp, 0);`

العنصر
بداية البحث

Preprocessor Directives

و هذه طريقة لترتيب الكود و جعله ضمن نطاقات معينة لعدم التشويش و تستخدم لتحسين و تنسيق مظهر الكود داخل ملف الشفرة المصدرية للبرنامج

```
#region "class employee"
public class employee
{
```

```
#end region
```

Comments

 بناء التعليقات داخل الشفره المصدريه :

تعتبر التعليقات جزء هام جدا لا يستهان به خاصه داخل المشاريع التي تحتاج الى
شفره مصدريه ضخمه كقواعد البيانات مثلا ولها فائده كبيره جدا عند الرجوع مره
اخرى الى الكود وعند عمل الفريق الجماعى

XML Comments:

Important elements in comments :

المقطع بداية التعليق وسيكون بنوع خط مختلف عن الكود

<C> the following must be in differnt fonts

المقطع الذي يحتوى على الشفره المصدريه او الكود

<Code> the following is code

المقطع الذي يحتوى على الاستثناءات او الاخطاء ومعالجتها

<exception> file containg exceptions

المقطع الذي يشرح بارامتر معين داخل الكود ووظيفته

<param> explain parameter in function

عادة لا تستخد جميعها مره واحده
في كل كود تكتبه انما تستخدم
حسب الحاجه اليها

المقطع الذي يضم بناء فضيل معين

///<Summary>
///employee class of the company

المقطع الذي يشرح وظيفة الفيمه التي تعود من دالة

معينة داخل الكود

<returns>explain the function result

الشرح المختصر لجزء معين من الكود

<Summary>abstract explain to code

المقطع الذي يشرح الخاصيه والقيميه التي تعطيها لمتغير او

القيميه التي تعود بها

```
{  
/// <summary>  
/// class of all operations  
/// class operations  
/// <c>  
/// now we build new class contain on all operation  
/// we need  
/// </c>  
///<code>  
///  
class Operations  
{int x, y, z;  
    public int X { ///<value>thie property well set value to number x </value>  
        set{x = value;}  
        get{return x;} }  
    public int xsquare(int a) {///<param>a well give the value of x</param>  
        x = a;  
        ///<return>function well return square of value x</return>  
        return x ^ 2; }  
///</code>  
/// </summary>
```

انتهى الجزء الاول من هذا الكتاب من اساسيات البرمجه فى لغة السى شارب
اعتمد على المعلومه المباشره واهتم بها اكثر وعذرا على عدم الاهتمام بالتنسيق

memorycode_84@yahoo.com