اللهم إني أسألك فهم النبيين، و حفظ المرسلين، و الملائكة المقربين، اللهم اجعل ألسنتنا عامرة بذكرك، و قلوبنا بخشيتك، وأسرارنا بطاعتك، إنك على كل شيء قدير، حسبنا الله و نعم الوكيل

شروحات فيديوهات المهندس دانتي للغة السي شارب دوت نت ٢٠٠٨

بسم الله الرحمن الرحيم

في هذا الكتاب سوف تجد شرح وافيًا لمبادئ الكونسل و **object oriented programming** ، وأنا لا أدعي تأليف الكتاب ولا أدعي شرحه، فكل هذه الشروحات قام بإنتاجها المهندس «دانتي ليو» وسوف تجد الروابط الخاصة بشروحات الفيديو على موقع عرب هاردوير ، وسوف تجد في هذا الكتاب تطبيق لهذه الفيديوهات على فيجوال سي شارب ٢٠٠٨، فإذا وجدت خطأ فقم بتصحيحه؛ لأن الخطأ سوف يكون ناتجًا عن سوء فهمي وذلك لقلة خبرتي في المجال البرمجي ، فما قمت به هو تجميع هذه الدروس في ملف حتى يسهل علي استرجاعها واستذكارها ، وكذلك كتبت النقاط المهمة، وأغفلت عن كثيرًا نظرًا لضيق الوقت، وبكل صدق نحن مدينون كثيرًا للمهندس «دانتي»، كما أشكره شكرًا جزيلاً على ما قدمه للمسلمين للعرب، سائلاً المولى عز وجل أن يجعل هذا العمل الطيب في ميزان حسناته.

وهذه الدروس أعتبرها ثروة طيبة في البرمجة وخاصة للمبتدئين مثلي ، فأنا ممن عانيت كثيرًا في البحث عن شرح وافي، وبكل صراحة لم أجد أفضل من هذا الشرح الوافي.

هذه هي روابط دروس الفيديو المهندس «دانتي» على منتدى عرب هاردوير :

الموقع الذي يحمل كورس الأستاذ دانتي على روابط متعددة

http://arabhardware.net/forum/showthread.php?t=181723

أو

رابط الكورس مجمع في رابط واحد

http://www.mediafire.com/?oxzl24jso4w85

ونسأل الله التوفيق والسداد

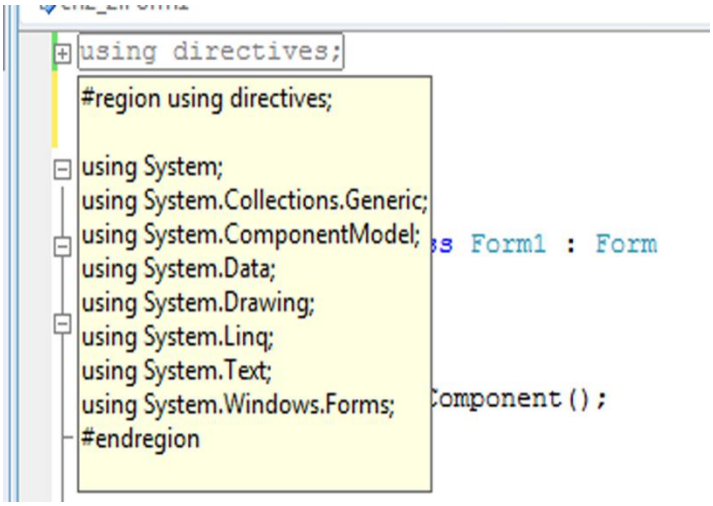
تجميع الطالب/ رفعت يسري

refaatyousry@yahoo.com

(*) **vc3** : إذا أردت إخفاء المكتبات الموجودة في قسم التصريحات أعلى البرنامج (أعلى منطقة الكود) تكتب في بداية المكتبات السطرين المظللين باللون الأحمر كما نشاهد.

```
   #region using directives;
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;
   #endregion
```

في حالة الضغط على علامة الزائد كما في الصورة التالية تجد كافة المكتبات التي تستخدمها في البرنامج.



(*) لعمل تحويل للمتغيرات وذلك لنستطيع أن نجري عليهم عمليات حسابية. مثل الطرح كما في المثال التالي:

```
static void Main(string[] args)
          {
//declaration part
sbyte one;
sbyte two;
sbyte dif;
int x;
//input part
one = 50;
two = 20;
//prossing part
dif = Convert.ToSByte (one  - two);
x = one - two;
//result part
Console.WriteLine("the ok is {0}", x)
}
```

قمنا بتحويل المتغيرات حتى نستطيع عمل طرح المتغيرات من بعضها. لماذا اخترنا sbyte ولم تختر **int** وذلك لأننا نريد استخدام أرقام بسيطة #.

(*) نفس المثال السابق لكن المستخدم يدخل الأرقام بنفسه كما نشاهد.

```
sbyte one;
sbyte two;
byte dif;
Console.WriteLine("input the first number");
one = Convert.ToSByte ( Console.ReadLine());

Console.WriteLine("input the second number");
two = Convert.ToSByte(Console.ReadLine());

dif = Convert.ToByte(one - two);
Console.WriteLine("the deference is {0}", dif);
```

(*) في هذا المثال التالي يفضل أن يدخل المستخدم رقم مع علامة عشرية مثل ٥.٢ مثلاً وذلك لأننا استخدمنا **double**:

```
double one, two, dif;

Console.WriteLine("input the first number");
one = Convert.ToDouble (Console.ReadLine());

Console.WriteLine("input the second number");
two = Convert.ToDouble (Console.ReadLine());

dif = Convert.ToDouble (one - two);

Console.WriteLine("the one value is {0} and two value is {1} the deference is
{2}", one, two, dif);
```

(*) التعامل مع string

نفس الكود السابق وتضيف عليه ما هو مظلل باللون الأحمر.

```
/////// التعامل مع string (النصوص)

double one, two, dif;
string universty;

Console.WriteLine("input universty and name");
universty = Convert.ToString (Console.ReadLine ());

Console.WriteLine("input the first number");
one = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("input the second number");
two = Convert.ToDouble(Console.ReadLine());

dif = Convert.ToDouble(one - two);

Console.WriteLine(universty);

Console.WriteLine("the one value is {0} and two value is {1} the deference is
{2}", one, two, dif);

  }
```

(*) عملية حسابية بسيطة.

```
//العمليات الحسابية
    {
    int x1 = 10, x2 = 3;
    Console.WriteLine(x1 * x2);
}
```

أو عند الحاجة إلى إخراج باقي القسمة:

```
{
int x1 = 10, x2 = 3;
Console.WriteLine(x1 % x2);

}
```

تجد الناتج يساوي ١ عند تشغيل البرنامج في بيئة الكونسل.

(*) مثال لزيادة **x3** بمقدار واحد كما نشاهد.

```
int x1 = 9 , x2 = 3 , x3;

x3 = ++x1;
Console.WriteLine(x3);
int x1 = 9 , x2 = 3 , x3;
```

سوف تجد الناتج يساوي ١٠ عند تشغيل البرنامج في بيئة الكونسل.

(*) مثال لزيادة **x1** بمقدار واحد ،

```
x3 = x1++;
 Console.WriteLine(x3);
 Console.WriteLine(x1);
```

سوف تجد ناتج **x3** يساوي ٩ كما هو و **x1** تجده يساوي ١٠

تابع العمليات الحسابية على المتغيرات.

قاعدة ثابتة//موجب أو سالب قبل المتغير زيادة او نقصان ، بعد المتغير لا زيادة ولا نقصان

```
    int var1, var2 = 6, var3 = 6;
var1 = var2++ * --var3 ;


//var1= 6++ * --6
//var1 = 6 * 5
  Console.WriteLine(var1);
```

مثال آخر : ؟

```
int var1, var2 = 6, var3 = 6;
var1 = ++var2 * 5 + 5 * 2;
//var = ++6 * 5 + 5 * 2
//var = 7 * 5 + 5 * 2
  Console.WriteLine(var1);// تجد قيمتها تساوي
```

مثال آخر:

```
int x1 = 10, x2 = 3, x3 = 3;
x3 = x1++;


Console.WriteLine(x3);// عشرة تجدها قيمتها
Console.WriteLine(x1);//عشر إحدى تجدها قيمتها
```

مثال آخر: لمزيد من التخصيص على العمليات الحسابية قبل المتغير وبعد المتغير.

استخدم الأقواس دائمًا حتى تتأكد من ان العمليات تتم بشكل صحيح.

```
int total, salary = 1600, bounus = 5, tax = 5, rent = 10, living = 300;
total = (salary + bounus) - (tax + rent + living);

  Console.WriteLine(total);
```

سوف تكون النتيجة ١٢٩٠

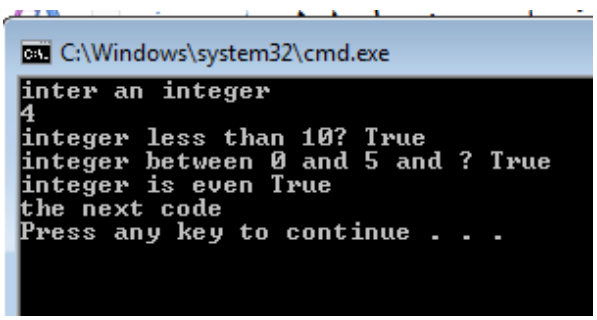
(*) علامة == تعني مقارنة وغالبًا تأتي مع الجملة الشرطية أو مع **true** و **false**

العلامة	الفئة	المثال	الحالة
==	ثنائي (مزدوج)	**Var1=var2==var3;**	١ تساوي صح في حالة التساوي بين ٢ و٣ أو العكس.
!=	ثنائي (مزدوج)	**Var1=var2!=var3**	١ تساوي صح في حالة عدم تساوي ٢ و٣، والعكس.
<	ثنائي (مزدوج)	**Var1=var2 < var3**	١ تساوي صح في حالة ٢ أصغر من ٣، والعكس.
>	ثنائي (مزدوج)	**Var1=var2 > var3**	١ تساوي صح في حالة ٢ أكبر من ٣ ، والعكس.
=>	ثنائي (مزدوج)	**Var1=var2 >= var3**	١ تساوي صح في حالة ٢ أكبر من أو تساوي ٣
=<	ثنائي (مزدوج)	**Var1=var2 <= var3**	١ تساوي صح في حالة ٢ أصغر من أو تساوي ٣

فيديو1 vc4 part

```
Console.WriteLine("inter an integer");
int myint = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("integer less than 10? {0}", myint < 10);
Console.WriteLine("integer between 0 and 5 and ? {0}",(0 <= myint) &&(myint <
5));
Console.WriteLine("integer is even {0}", myint % 2 == 0);

goto you;

Console.WriteLine("have a good day");

you:
   Console.WriteLine("the next code");
```

تكون النتيجة إذا كتبنا رقم (٤) هكذا.



تأكدنا أن **(myint < 10)** وفي حالة عكس ذلك يضع القيمة **false** . وتأكدنا أن **(myint < 5)** وإذا كانت عكس ذلك يضع القيمة **false** .

وتأكدنا أن ناتج باقي قسمة **myint** ÷ ٢ = صفر ، أي أن قيمة **myint** زوجية **(even)**. وإذا كانت فردية يضع **false**

وتجد عند كتابة **go to you;** لا ينفذ البرنامج التعليمة التي أسفل **go to you;** ثم يذهب البرنامج إلى التعليمة **you:** وينفذ ما تحتها .

ملحوظة: علامة && تشترط أن تكون القيمتين السابقة والتالية صحيحتين.

فيديو vc4 part1 - CH4_B ==========

بعض حالات التحويل **(convert)**:

```
//short = convert.ToInt16
//long=convert.ToInt64
//double = convert.ToDouble

int mysring;

Console.WriteLine("input value");


mysring = Convert.ToInt32(Console.ReadLine());

string res = (mysring < 10)? "mystring is less than 10" : "mystring is greater
than 10";
Console.WriteLine(res);
```

(*) حالة **if** : **(if statement)**

إذا كان الجملة الشرطية بها أكثر من احتمال فلا بد من عمل قوسين المجموعة **{}** على الشرطين ، أما إذا كانت الجملة بها احتمال واحد فليس ضروري عمل قوسين المجموعة وذلك كالتالي.

١-حالة وجود احتمال واحد:

```
if (x == 7)  Console.WriteLine("seven");
```

٢- حالة وجود احتمالين:

```
if (x == 7)
{
    Console.WriteLine("seven");
    Console.WriteLine("sept with francais");
  }
```

مثال على حالة **if** :

```
int x = 7;
if (x == 5) Console.WriteLine("five");
if (x == 6) Console.WriteLine("six");
if (x == 7)
{
    Console.WriteLine("seven");
    Console.WriteLine("sept with francais");
}
if (x == 8) Console.WriteLine("eight");
if (x!= 6 && x !=5 && x != 7) Console.WriteLine("UnNoun");

        }
```

ولكن من الأفضل عمل نفس المثال السابق بهذه الطريقة لضمان قوة الكود وهنا تظهر أهمية **else**.

```
int x = 7;
if (x == 5) Console.WriteLine("five");
else if (x == 6) Console.WriteLine("six");
else if  (x == 7)
{
    Console.WriteLine("seven");
    Console.WriteLine("sept with francais");
}
else if (x == 8) Console.WriteLine("eight");
else Console.WriteLine("UnNoun");

          }
```

و **else** الأخيرة تخبر البرنامج بأنه عكس كل ما سبق قم بكتابة **("unNoun")**.

---

(١) أول برنامج مقارنة الأرقام

قم بكتابة برنامجي قوم بالمقارنة بين رقمين ويقوم بطباعة الرقم الأول أكبر أو الرقم الثاني أكبر أو الرقمين متساويين بعد المقارنة بين الرقمين. **Ch4**

```
int a,b;
g:
Console.WriteLine("input the first (A)");

a = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("input the second(B)");
b = Convert.ToInt32(Console.ReadLine());

if (a > b) Console.WriteLine("A is greater than B");
else if (b > a) Console.WriteLine("B is greater than A");
else Console.WriteLine("A equals B");
goto g;
          }
```

استخدمنا **goto** حتى يعود المستخدم لكتابة الأرقام مرة أخرى .

```
# فيديو vc4 part1 - CH4_2 ======================================
//2-  برنامج آيس كريم
```

(٢) شركة ايس كريم قامت بشراء ثلاثة خطوط إنتاجية لإنتاج أنواع جديدة من الأيس كريم ، موظفوا الضرائب قاموا بزيارة الشركة في نهاية السنة المالية وقاموا بحساب الضريبة كالآتي:

- قيمة الأرباح هي عبارة عن قيمة المبيعات مطروح منها المصاريف والتي تشمل بدورها مصارف تصنيعية ومصاريف نثرية.

٠١:٠٦:٤٣

- ٥% ضريبة على المنتج الأول
- ٣% ضريبة على المنتج الثاني

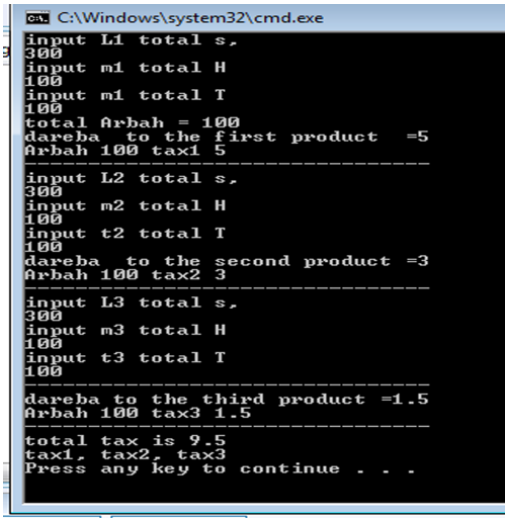
- ١.٥% ضريبة على المنتج الثالث.

الأرباح = المبيعات – المصاريف (التصنيعية + النثرية)

المطلوب إيجاد قيمة كل ضريبة ومجموع الضريبة الكلي خلال السنة المالية.

```
int L1, L2, L3;//إجمالي مبيعات لكل منتج من الثلاث منتجات
int m1, m2, m3;//إجمالي المصاريف التصنيعية لكل منتج من الثلاث منتجات
int t1, t2, t3;//إجمالي المصاريف النثرية لكل منتج من الثلاث منتجات
int p1, p2, p3; //الأرباح لكل منتج
double tax1, tax2, tax3,totaltax;//الضرائب لكل منتج والإجمالي

                                        //المنتج الأول
                                //وتعني اقرأ إجمالي قيمة المبيعات للمنتج الأول

Console.WriteLine("input L1 total s, ");
L1 = Convert.ToInt32(Console.ReadLine());
                                        //وتعني اقرأ القيمة التصنيعية للمنتج الأول
Console.WriteLine("input m1 total H");
m1 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("input m1 total T");//// وتعني اقرأ القيمة التصنيعية للمنتج
الأول
t1= Convert.ToInt32(Console.ReadLine());//الأرباح = المبيعات - المصاريف التصنيعية
+ النثرية
p1 = (L1) - (m1 + t1);
Console.WriteLine("total Arbah = {0}",p1);
tax1  = ( 5 * p1 ) / 100;
Console.WriteLine("dareba  to the first product  ={0}", tax1);
Console.WriteLine("Arbah {0} tax1 {1}", p1, tax1);
Console.WriteLine("---------------------------------");

                                                //المنتج الثاني
Console.WriteLine("input L2 total s, ");
L2 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("input m2 total H");
m2 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("input t2 total T");
t2 = Convert.ToInt32(Console.ReadLine());

p2 = (L2) - (m2 + t2);
tax2 = (3 * p2  ) / 100;
Console.WriteLine("dareba  to the second product ={0}", tax2);
Console.WriteLine("Arbah {0} tax2 {1}", p2, tax2);
Console.WriteLine("---------------------------------");
                                                //المنتج الثالث
Console.WriteLine("input L3 total s, ");
L3 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("input m3 total H");
m3 = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("input t3 total T");
t3 = Convert.ToInt32(Console.ReadLine());


Console.WriteLine("---------------------------------");
p3 = (L3) - (m3 + t3);
tax3 = (1.5 * p3) / 100;
Console.WriteLine("dareba to the third product ={0}", tax3);
Console.WriteLine("Arbah {0} tax3 {1}", p3, tax3);
Console.WriteLine("-----------------------------------");
totaltax = tax1 + tax2 + tax3;
Console.WriteLine("total tax is {0}", totaltax);
```

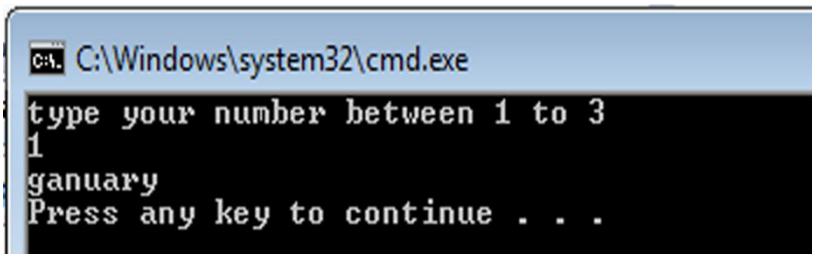
```
//معرفة أي منتج ضرائبه أكثر

if ((tax1 > tax2) && (tax1 > tax3))
{
 if (tax2 > tax3) Console.WriteLine("tax1, tax2, tax3");
 else Console.WriteLine("tax1, tax3, tax2");

}

else if ((tax2 > tax1) && (tax2 > tax3))
{
    if (tax1 > ta x3) Console.WriteLine("tax2, tax1, tax3");
    else Console.WriteLine("tax2, tax3, tax1");

}
if ((tax3 > tax1) && (tax3 > tax2))
{
    if (tax1 > tax2) Console.WriteLine("tax3, tax1, tax2");
    else Console.WriteLine("tax3, tax2, tax1");
}
```



(٣) قم بكتابة برنامج يقوم بقراءة رقم وطباعة يناير حينما تكون قيمة الرقم ١ وطباعة فبراير إن كانت قيمته ٢ وطباعة مارس إن كانت قيمته ٣ وطباعة خارج الربع الأول من العام لأي قيمة أخرى **ch4_3**

```
# فيديو vc4 part1 - CH4_3 ======================================

//الطريقة الأولى حالة سويتش

int x;
Console.WriteLine("type your number between 1 to 3 ");
x = Convert.ToInt32(Console.ReadLine());

switch (x)
{
    case (1): Console.WriteLine("ganuary"); break;
    case (2): Console.WriteLine("fep"); break;
    case (3): Console.WriteLine("fep"); break;
```

```
    default: Console.WriteLine("out of 1st year quarter"); break;
}
```



الطريقة الثانية طريقة **if**

```
g:
int x;

Console.WriteLine("type your number between 1 to 3 ");
x = Convert.ToInt32(Console.ReadLine());
if (x == 1) Console.WriteLine("ganuary");
else if (x == 2) Console.WriteLine("febreuary");
else if (x == 3) Console.WriteLine("march");
else Console.WriteLine("out of quarter");
goto g;

        }
```

(*) أمثلة على الحلقات التكرارية **do while** ، **while**، **for** .

```
  int i = 1;

do
{
    Console.WriteLine("{0}. refaat", i);
    i++;

} while (i <= 10);

                    ==========================================الطريقة الثانية//
int i = 1;
while (i <= 10)
{
    Console.WriteLine("{0}. refooooooooo", i);
    i++;

}

                  ==========================================الطريقة الثالثة//for
for (i = 1; i <= 10; i++)
{
    Console.WriteLine("{0} refaat hamed", i);

}
```

```
//CH4_4----------------------------------------------------------------------
```

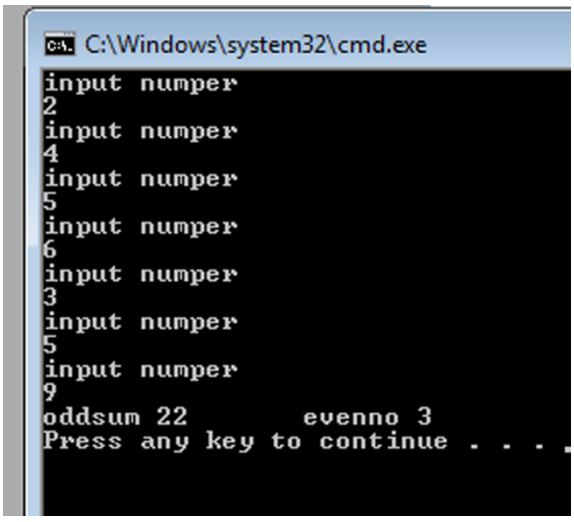
(٤) قم بكتابة برنامج يقرأ مجموعة من الأرقام تنتهي بالرقم ٩ ثم قم بإيجاد عدد الأرقام الزوجية ومجموع الأرقام الفردية **ch4_4**

Ex
2 4 5 6 7 8 9
4 even no
21 odd sum

Even تعني الأرقام الزوجية، و **odd** تعني الأرقام الفردية.

الحل:

```
int n;
int oddsum=0, evenno=0;

do
{
Console.WriteLine ("input numper");
n=Convert.ToInt32 (Console.ReadLine ());
    if(n%2==0) evenno=evenno +1;
    else oddsum =oddsum +n ;
}
while (n!=9);

Console.WriteLine("oddsum {0}       evenno {1}", oddsum, evenno);
}
```



(٥) قم بكتابة برنامج يقوم بإيجاد قيمة الفكتوريل **ch4_5**

الحل

CH4_5----------

ما معنى (الفكتوريل) .

لو عندنا رقم ٥ مثلاً ما هو الفكتوريل له؟

5 =
5*4*3*2*1

أي أن الفكتوريل هو مكونات العدد خمسة (حاصل ضرب ٥×٤×٣×٢×١) = ١٢٠

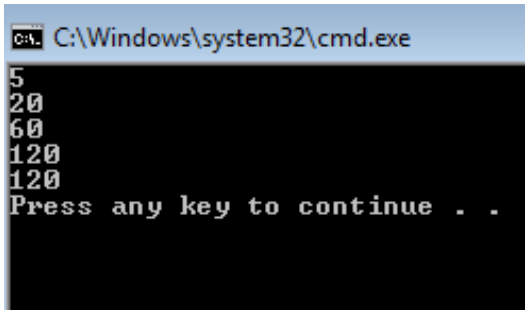
```
int x = 5;
int f = 1;

for (int i = x; i >= 1; i--)
{
    f = f * i;
}
    Console.WriteLine("={0}",f );
        }
```

سوف يظهر الناتج على شاشة الكونسل = ١٢٠

أما إذا أردنا أن تظهر الفكتوريل للرقم (٥) بالترتيب كمصفوفة نكتب الكود بالشكل التالي

```
int x = 5;
int f = 1;

for (int i = x; i >= 1; i--)
{
    f = f * i;
    Console.WriteLine("{0}", f);
}
```

حتى تظهر النتيجة كما نشاهد في الصورة.



إذا أردت المستخدم يتحكم في الإدخال فتكتب الكود كما يلي:

```
int f = 1;

int re;
Console.WriteLine ("input");

re=Convert.ToInt32 (Console.ReadLine());
for (int i = re; i >= 1; i--)
{
    f = f * i;
}
        Console.WriteLine("{0}", f);
        }
```

(٦) بكتريا مزروعة في وسط مختبر تتكاثر بطريقة الانقسام الخيطي بمقدار الضعف كل ٣٠ ثانية ، فإذا كان عدد البكتريا المزروعة هو ١٠ عند الساعة ٨ صباحًا فكم سيكون عددها بعد مرور ساعة **ch4_6**

الحل

* علمنا أن البكتريا تتكاثر بمقدار الضعف كل ٣٠ ثانية، أي أننا لنعرف مقدار التكاثر لمدة ساعة : فإن البكتيريا سوف تتكاثر في ٦٠ دقيقة ١٢٠ مرة . لأنها تتكاثر مرتين في الدقيقة.

سوف نستخدم حلقة **for**

الطريقة الأولى: يصبح الكود كالتالي:

```
   double  pct = 10;

for (int i = 1; i <= 120; i++)
{
    pct = pct * 2;

}
Console.WriteLine("{0} ", pct);
}
```

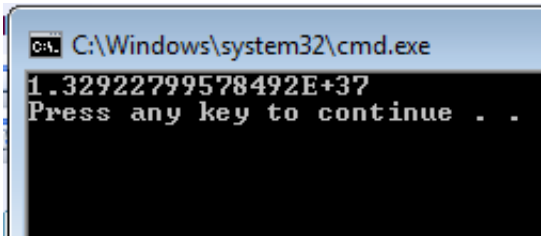
قمنا باختيار **double** حتى نستطيع تخزين عدد البكتريا خلال ساعة (٦٠ دقيقة) وذلك لأن التكاثر كما علمنا يزيد بمقدار الضعف أي مرتين في الدقيقة أي ١٢٠ مرة في الساعة.

الطريقة الثانية: نستطيع كتابة الكود بهذا الشكل أيضًا

```
   double  pct = 10;

for (int i = 1; i <= 60; i++)
{
    pct = pct * 4;

}
Console.WriteLine("{0} ", pct);
        }
```

في هذه المرة قمنا بتغير **i<=60** بدلاً من ٢٠ وضربنا البكتريا **pct** × (٤) بدلا من ٢

فتعطينا نفس النتيجة كما نرى في الصورة.



(٧) في استراليا نظام قروض المنازل يعتمد على تسديد دفعة أولية تساوي ٥% من قيمة المنزل زائد ٢٠ ألف دولار كمعاملة ويتم حساب فائدة ٥ بالألف لكل مائة ألف دولار لمدة ال١٢ شهر الأولى ثم تستمر نفس نسبة الفائدة شهريًا لكن مع إمكانية البدء بتسديد مبالغ إضافية تخصم من سعر البيت الأصلي علمًا أن أسعار المنزل تتراوح بين ٣٠٠ ألف ومليون دولار قم بإيجاد الآتي:

١- لو كان بعد مضي الـ ١٢ شهر الأولى قام المشتري بتسديد مبلغ ثابت شهريًا يزيد عن الفائدة فقط بعرض جدول يتضمن عدد الأشهر التي يستمر التسديد فيها وحجم الفائدة في كل شهر وحجم المبلغ المخصوم وحجم المبلغ المتبقي.

٢- مع نهاية هذا الجدول قم بعرض سعر البيت الأصلي ومجموع الفائدة الكلي وسعر البيت الإجمالي بعد إضافة قيمة المعاملة ومجموع الفائدة إليه **ch4_7**

مثال:

Home=300000
First pay = 5% = 15000
File=20000 فلوس معاملة
300000-15000= 285000
First 12 month 0.005 fayda in all 100,000 $
285000 * 0.005 fayda = 1425
1425 * 12 month = 17100 in the first year

في أول سنة سوف تدفع فائدة ١٧١٠٠

في السنة الثانية مطلوب منك سداد فائدة ١٤٢٥ أيضًا عن كل شهر

فرضًا لو دفعت زيادة عن ١٤٢٥ شهريًا يبتدي يخصم من سعر البيت الذي هو (٢٨٥٠٠٠)

* مطلوب منك سداد قيمة المنزل (٢٨٥٠٠٠) × فائدة ٠.٠٠٥% = (١٤٢٥)

أي: لو سددت ٣٠٠٠ دولار في شهر ما : فسوف تذهب ١٥٧٥من سعر البيت و ١٤٢٥ فائدة. أي أن: ٣٠٠٠ =١٥٧٥ - ١٤٢٥

فيكون باقي الحساب بعد نهاية الشهر الأول من السنة الأولى

٢٨٥٠٠٠ – ١٥٧٥= ٢٨٣٤٢٥

* في الشهر الثاني مطلوب منك سداد قيمة المنزل (٢٨٣٤٢٥) × فائدة ٠.٠٠٥% = (١٤١٧)

أي قيمة الفائدة قلت عن السابق.

الحل كالتالي:

```
{
        static void Main(string[] args)
        {
double  price;// السعر الرئيسي للبيت

          سعر البيت بعد خصم 5 بالمائة//
double  priceAfter;
double precent;//المائة في أي

       إجمالي الفائدة عن السنة الأولى//
double firstyear;

      مقدار الدفع الشهري//
double pay;

      مقدار الخصم الشهري//
double p2;


inputagine:
Console.WriteLine("input the price");
price = Convert.ToInt32(Console.ReadLine());
if ((price < 300000) || (price > 1000000)) goto inputagine;
precent = 0.05 * price;

Console.WriteLine ("5 % = {0}",precent);

Console.WriteLine("to open file you must pay 20000");
priceAfter = price - precent;

Console.WriteLine ("priceAfter = {0}" , priceAfter );
firstyear = priceAfter * 0.005 * 12;
Console.WriteLine("profit payed for the first 12 mounths ={0}", firstyear);
Console.WriteLine("How many to pay each month");

pay=Convert.ToDouble (Console.ReadLine ());

int month=0;
double tprofit=0;


do
{
 month ++;
    p2=0.005 * priceAfter  ;
    tprofit += p2;//or- tprofit = tprofit + p2


   priceAfter = priceAfter - (pay - p2);
   Console.WriteLine("{0} home price left {1} profit month {2} home pay
{3}",month ,priceAfter,p2,pay - p2);

} while (priceAfter > 0);

Console.WriteLine("---------------------------------");


    double total = price + 20000 + firstyear + tprofit ;
    Console.WriteLine("total home coast {0} ", total);
        }
```

```
    }
```

VC5Part1 CH5_1==========================

```
static void Main(string[] args)
        {

int a;

float b=6.0f;

a = checked((int)b);
Console.WriteLine(a);
Console.ReadKey();

        }
```

CH5_2 part1==========================

```
            ushort x1;
char x2 = 'b';
x1 = x2;
Console.WriteLine("ushort ={0}", x1);
Console.WriteLine("char={0}", x2);
//==========================
ushort x3;
char x4 = 'B';
x3 = x4;
Console.WriteLine("ushort ={0}", x1);
Console.WriteLine("char={0}", x2);


//=================================

char x5;
int i;

for ( i = 1; i <= 100; i++)
{
    x5 = (char)i;
    Console.WriteLine("char  ={0}", x5);
    Console.WriteLine("number={0}", i );
 }
```

CH5_3 part1==========================

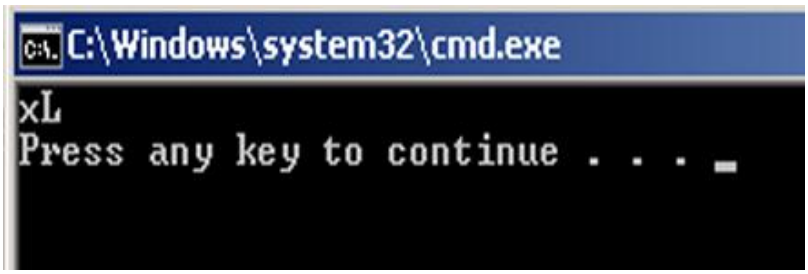
```
short shortResult, shortVal = 4;
int integerVal = 67;
long longResult;
float floatVal= 10.5F;
double doubleResult, doubleVal = 99.999;
string stringResult, stringVal = "17";
bool boolVal = true;


Console.WriteLine("Variable Conversion Examples\n");
doubleResult = floatVal * shortVal;
Console.WriteLine("Implicit, - > double: {0} * {1} - > {2}", floatVal,shortVal,
doubleResult);
```

الأكبر يمكن أن يحتوي الأكبر **implicit**. وحتى يحتوي الأصغر الأكبر لا بد من **explicit**

Type	Can safely be converted to
Byte	short, ushort, int, uint, long, ulong, float, double, decimal
Sbyte	short, int, long, float, double, decimal
Short	int, long, float, double, decimal
Ushort	int, uint, long, ulong, float, double, decimal
Int	long, float, double, decimal
Uint	long, ulong, float, double, decimal
Long	float, double, decimal
Ulong	float, double, decimal
Float	Double
Char	ushort, int, uint, long, ulong, float, double, decimal

#VC5Part2 CH5_4==========

```
class Program
    {

        enum clothes
        {
L = 1,
xL,
xxL,
xxxL = L+7

        }
        static void Main(string[] args)
        {
clothes x ;
x =(clothes)2;
Console.WriteLine(x);
}
    }
```



VC5Part2 CH5_5==========

(*) يكتب الكود في أعلى **static void Main(string[] args)**

```
        enum orientation : byte
        {
            north = 1,
            south = 2,
            east = 3,
            west = 4
        }
          {

        byte directionByte;
        string directionString;

orientation myDirection = orientation.north;

Console.WriteLine("myDirection = {0}", myDirection);
directionByte = (byte)myDirection;
directionString = Convert.ToString(myDirection);
Console.WriteLine("byte equivalent = {0}", directionByte);
Console.WriteLine("string equivalent = {0}", directionString);
          }
```

قمنا بتعريف المتغير **directionByte** من نوع **byte** وذلك حتى نحصل على قيمة ١ أو ٢ أو ثلاثة أو اربعة. على حسب تعريف **myDirection**

(*) نتحدث عن **struct** وفوائدها.

```
class Program
    {
        struct student
        {
            public string name;
            public int mark1,mark2,mark3;
            public float averag;


        }

        static void Main(string[] args)
        {
            student student1, student2, student3;

            student1.name = "ahmed";
            student1.mark1 = 60;
            student1.mark2 = 65;
            student1.mark3 = 85;

            student1.averag = (student1.mark1 + student1.mark2 + student1.mark3)
/ 3;
            Console.WriteLine(student1.averag);
            Console.WriteLine("{0}, {1}", student1.name, student1.mark1);
          }
```

سوف يظهر لنا الناتج ٧٠ و (أحمد ، ٦٠).

ويمكن وضع **struct** بداخل **struct** أو أكثر .

حتى تستطيع قراءة أحدهما من الآخر. ولا تنسى عمل **public** لعناصر الـ **struct** حتى تستطيع قراءتها ضمن الـ **struct** الأخرى.

وإليك نفس المثال السابق مع إضافة **struct** جديد . شاهد المثال ولاحظ الإضافة:

```
class Program
    {
        struct note
        {
            public string Birthday;

        }

         struct address
        {
            public int sector, streat, home;

            public note nt;

        }
        struct student
        {
            public string name;
            public int mark1,mark2,mark3;
            public float averag;


            public address adres; // strcut address بـداخل strcut student

        }

        static void Main(string[] args)
        {


            student student1; //, student2, student3;



            student1.adres.nt.Birthday = "1397";
            student1.name = "ahmed";
            student1.mark1 = 60;
            student1.mark2 = 65;
            student1.mark3 = 85;
            student1.adres.sector = 510;
            student1.adres.home = 10;
            student1.adres.streat = 30;

            student1.averag = (student1.mark1 + student1.mark2 + student1.mark3)
/ 3;
            Console.WriteLine(student1.averag);
            Console.WriteLine(student1.adres.sector );// قـيمة طبـاعة strcut adres
            Console.WriteLine("{0}, {1}", student1.name, student1.mark1);
            Console.WriteLine(student1.adres.nt.Birthday);
             }
```

كما نرى أدخلنا **struct adress** بداخل **struct student**.

والآن بقي لنا أن نعرف كيف نستدعي **enum** بداخل **struct** :

إليك المشروع بالكامل بعد إقحام **enum** وطباعة عنصر منها.

```
}
        struct note
        {
            public string Birthday;

        }

         struct address
        {
            public int sector, streat, home;

            public note nt;

        }
        struct student
        {
            public string name;
            public int mark1,mark2,mark3;
            public float averag;

                                                    //سوف نقوم بتعريف إينم
            public direc orient;

            public address adres; // strcut address بداخل strcut student

        }

        static void Main(string[] args)
        {


  student student1; //, student2, student3;
  student1.orient = (direc)1;

  student1.adres.nt.Birthday = "1397";
  student1.name = "ahmed";
  student1.mark1 = 60;
  student1.mark2 = 65;
  student1.mark3 = 85;
  student1.adres.sector = 510;
  student1.adres.home = 10;
  student1.adres.streat = 30;

  student1.averag = (student1.mark1 + student1.mark2 + student1.mark3) / 3;
  Console.WriteLine(student1.averag);
  Console.WriteLine(student1.adres.sector );// قيمة طباعة strcut adres
  Console.WriteLine("{0}, {1}", student1.name, student1.mark1);
  Console.WriteLine(student1.adres.nt.Birthday);
  //طباعة الإينم
  Console.WriteLine(student1.orient);
   }
```

VC6Part1 CH6_1==========

```
class Program
    {

//function (write)

     static void write ()
     {
         Console.WriteLine("one");
         Console.WriteLine("two");
         Console.WriteLine("three");

    }
        static void Main(string[] args)
        {
//طريقة استدعاؤه
            write();

          }
```

VC6Part1 CH6_2==========

قبل هذا الدرس لا بد لك أن تراجع الفيكتوريل اضغط هنا . ص (١٥)

هذه هي الطريقة القديمة التي ممكن تقوم بعمل الفيكتوريل لأكثر من متغير مثل: **x1, x2, x3**

هما المثال التالي:

```
//متغير من لأكثر الفيكتوريل على للحصول القديمة الطريقة
            // 5! = 5*4*3*2*1= 120 , 4! = 4*3*2*1= 24 , 3! = 3*2*1= 6 ===== 150
            // y = 5! + 4! + 3! اي y = 120 + 24 + 6

            int x1 = 5, x2 = 4, x3 = 3;
            int y, f1 = 1, f2 = 1, f3 = 1;

            //لنستطيع الحصول على الفكتوريل للمتغير إكس ون

            for(int i =1; i<= x1 ; i++)
            {
                f1 *= i;
            }

            for (int i = 1; i <= x2; i++)
            {
                f2 *= i;
            }
            for (int i = 1; i <= x3; i++)
            {
                f3 *= i;
            }
            y = f1 + f2 + f3;
            Console.WriteLine(y);
          }
```

أما الطريقة المثلى هي استخدام **function**

```
//نكتب هنا الفانكشن

        static int face (int x)
        {
            int f = 1;
            for (int i = 1; i <= x; i++)
            {
                f *= i;

            }

            return (f);
        }


        static void Main(string[] args)
        {


            y = (face (x1) + face(x2) +  face (x3));
            Console.WriteLine(y);

            Console.WriteLine(face (x1) + face (x2) + face (x3));
```

سوف تجد النتيجة (١٥٠) وفي السطر الثاني (١٥٠) .

مثال آخر لتدعيم إنشاء الفانكشن:

```
class Program
    {


//        هنا نكتب فانكشن ثاني إذا أردنا أن نجمع مجموعة أرقام على سبيل المثال:

        static int addition(int x1, int x2, int x3, int x4)
        {
            int sum;
            sum = x1 + x2 + x3 + x4;
            return (sum);
        }

        static void Main(string[] args)
        {

            int y, x1 = 5, x2 = 4, x3 = 3;


            Console.WriteLine(addition(10,20,30,40));
```

سوف تجد النتيجة (١٠٠) .

VC6Part1 CH6_3========== 00:22:14

يتحدث هذا الجزء عن **params** :

نقوم بعمل **function** كالتالي:

```
class Program
    {
    //function

        static int ser(params int[] d)
        {
            int m = d[0];
            foreach (int val in d)
            {
                if (m < val) m = val;}
            return (m);
        }

            static void Main(string[] args)
        {
            Console.WriteLine(ser(1, 4, 3,4,5,6));
      }
    }
```

قمنا بالكتابة

سوف تجد النتيجة ٦ لأن رقم (٦) هي أكبر قيمة في الأرقام التي تمت كتابتها في الكونسل.

VC6Part1 CH6_4========== ٠٠:٣١:٤٨

سوف نتحدث عن تمرير المتغير عن طريق **reference**

```
  static void doublenumber(ref int val)
        {
            val *=2;
            Console.WriteLine(val);
          }

        static void Main(string[] args)
        {
            int mynumber = 10;
            Console.WriteLine(mynumber);
            doublenumber(ref mynumber);
            Console.WriteLine(mynumber);
        }
```

سوف تجد النتيجة في الكونسل في السطر الأول ١٠ والثاني ٢٠ والثالث ٢٠.

 # VC6Part1 CH6_5========================== 00:38:09

نتحدث عن الكلمة المفتاحية **out** والفرق بينها وبين **reference**

الفرق أن **out** تتعامل مع المتغير الذي ليس له قيمة وتعطيه قيمة وحتى إذا كان هذا المتغير له قيمة فإذا كتبنا كلمة **out** قبله فيكون ليس له قيمة ويأخذ قيمة متغير الدالة **maxIndex** .

أما **reference** تتعامل مع المتغير الذي له قيمة ومن ثم لا تغير قيمته وتحتفظ بقيمة الدالة الأصلية.$

```
//function 5 out

        static int MVi(int[] intArray, out int maxIndex)
        {
            int maxVal = intArray [0];
            maxIndex = 0;
            for (int i = 1; i < intArray.Length; i++)
            {
                if (intArray[i] > maxVal)
                {
                    maxVal = intArray[i];
                    maxIndex = i;
                }
            }
            return maxVal;
        }

        static void Main(string[] args)
        {

            //function 5 out
            int[] A = { 1, 8, 3, 6, 2, 5, 9, 3, 0, 2 };
            int Index; //←
```

//هنا لو كتبنا أن هذا المتغير يساوي أي قيمة فلن يلتفت إليها البرنامج وسوف ينفذ قيمة **Maxval**

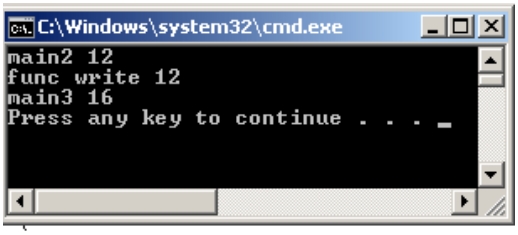
```
            Console.WriteLine("The maximum value in myArray is {0}",MVi(A, out
Index));
            Console.WriteLine("The first occurrence of this value is at element
{0}",
            Index + 1);


        }
```

VC6Part1 CH6_6========================== 00:48:01

سوف نتحدث عن **variable scope** أو «بُعد المتغيرات»

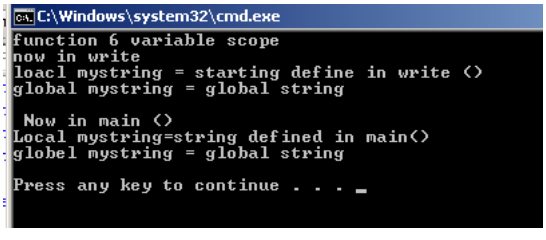
```
class Program
    {
        static int x;

        static void write ()

        {
        Console.WriteLine("func write {0}", Program.x);
            x = 18;
        هنا تعتبر إكس هي العالمية لأنه لا يوجد إكس محلية في الفانكشن                //
        }
        static void Main(string[] args)
        {

            int x = 5;
            Program.x = 12;
            Console.WriteLine("main1 Local {0}", x);

            Console.WriteLine("main2 {0}", Program.x);
            x = 16;
            write ();
            Console.WriteLine("main3 {0}", x);

        }
```

وسوف تجد النتيجة التي تظهر في شاشة الكونسول كالتالي:



VC6Part1 CH6_7==========================01:00:45

```
class Program
  {
        static string mystring;

        static void write()
        {
            string mystring = "starting define in write ()";
            Console.WriteLine("now in write");
            Console.WriteLine("loacl mystring = {0}", mystring);
            Console.WriteLine("global mystring = {0}", Program.mystring);
          }
        static void Main(string[] args)
        {
            int fd;
            Console.WriteLine("function 6 variable scope");
            string mystring = "string defined in main()";
            Program.mystring = "global string";
            write();
            Console.WriteLine("\n Now in main ()");
            Console.WriteLine("Local mystring={0}",mystring );
            Console.WriteLine("globel mystring = {0}",Program.mystring);
            Console.WriteLine();
        }
```

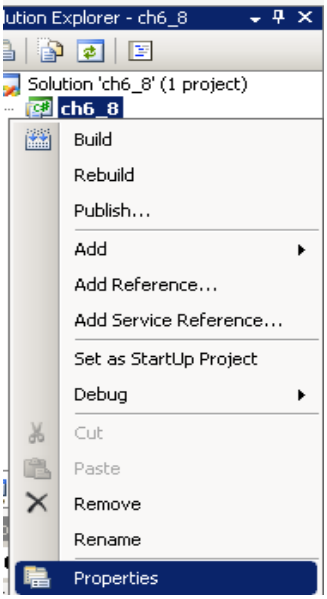


بداية الفيديو ==========**# VC6Part2 CH6_8**

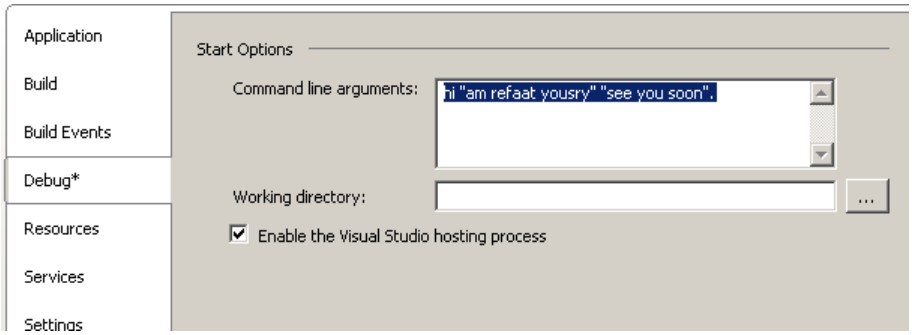
سوف نتحدث عن الـ **main**

أشكالها:

```
static void Main()
static void Main(string[] args)
static int Main()
static int Main(string[] args)
```

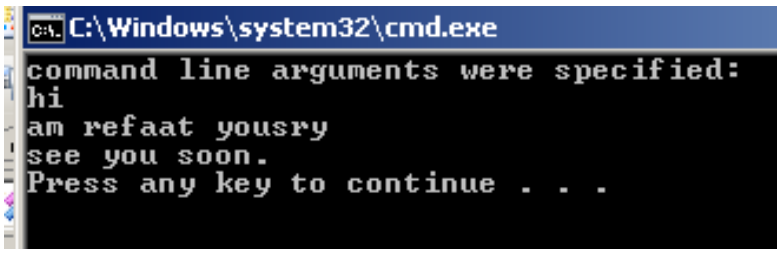


بعد ذلك نتحدث عن **debug** ونستطيع استدعاها بعد فتح قائمة **properties** ونختار **debug** كما نرى في الصورة:



إليك الكود لاستدعاء الـ **arguments** ←

```
class Program
    {
        static void Main(string[] args)
        {
Console.WriteLine("command line arguments were specified:", args.Length);

            foreach (string arg in args)
            {
            Console.WriteLine(arg);

             }
        }
  }
```



VC6Part2 CH6_9========================== 00:08:01

يتحدث عن **struct**

```
struct student
        {
            public string name;
            public int biomark;
            public int chmark;
            public int medmark;
            double average;


            public void set(string n1, int m1, int m2, int m3)
            {
                name = n1;
                biomark = m1;
                chmark = m2;
                medmark = m3;
                average = (biomark + chmark + medmark) / 3;
            }

            public void print()
            {
                Console.WriteLine("------------- student------------- ");
                Console.WriteLine("name is {0}", name);
                Console.WriteLine("biomark is {0}", biomark);
                Console.WriteLine("chmark is {0}", chmark);
                Console.WriteLine("medmark is {0}", medmark);
                Console.WriteLine("average is {0}", average);
                Console.WriteLine("-------------------------------- ");
            }

            static void Main(string[] args)
            {
                student s1 = new student();
                student s2 = new student();
                student s3 = new student();

                s1.set("mohamed", 29, 30, 12);
                s2.set("ahmed", 23, 25, 10);
                s3.set("sara", 24, 30, 15);

                s1.print();
                s2.print();
                s3.print();
            }
        }
```

VC6Part2 CH6_10==========00:28:21

تأمل كيف يتم التعامل مع الأوفر لودنج overloading

```
class Program
    {
        //overloading
        static int add(int a, int b)
        {return (a + b);}

        static int add(int a, int b ,int c)
        {return (a + b + c);}

        static int add(ref int a, ref int b)
        {
            a += 3;
            b += 4;
            return (a + b);
        }

        static double add(double a, double b)
        {return (a + b);}

        static string add(string a, string b)
        {return (a +" "+ b);}


        static void Main(string[] args)
        {
        int m1 = 10;
        int m2=5;
        int m3 = 7;

        double d1= 10.3;
        double d2 = 5.3;

        string s1 = "maha";
        string s2 = "ashraf";

            Console.WriteLine("integer 3 = {0}",add (m1,m2,m3));
            Console.WriteLine("integer normal = {0}",add(m1,m2));
            Console.WriteLine("integer ref={0}",add(ref m1,ref m2));
            Console.WriteLine("double= {0}",add(d1,d2));
            Console.WriteLine("string ={0}",add(s1, s2));

        }
      }
```

VC6Part2 CH6_11==========00:42:42

سوف نتكلم عن التفويض **delegate** وأهميته.

```
class Program
      {
            // التفويض

         delegate double processDelegate (double  multiply, double divide);

        static  double multiply(double  param1, double param2)
        {
            return param1 * param2;

        }

        static double divide (double param1, double param2)
        {
            return param1 / param2;
        }

        static void Main(string[] args)
        {
            processDelegate process;
            Console.WriteLine("inter 2 number with comma");
            string input = Console.ReadLine();
            int commapos = input.IndexOf(',');
            Console.WriteLine("length to input.length ={0} ",input.Length );

//هنا input سوف تساوي  طول الكتابات

            double x1 = Convert.ToDouble (input.Substring(0, commapos));
            double x2 = Convert.ToDouble (input.Substring(commapos + 1,
(input.Length - commapos - 1)));

            Console.WriteLine("enter M to multiply or D to divide");
            input = Console.ReadLine();
            if (input == "M")
            process = new processDelegate (multiply);
            else
            process = new processDelegate(divide);

            Console.WriteLine("result is :{0}",process (x1,x2));

            Console.WriteLine("the place of commapos here <,> = {0} ",
commapos);

            }
    }
```

بداية فيديو جديد # VC7 Part1 CH7_1==========

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

           /* DateTime d = DateTime.Now;
            Console.WriteLine("today is {0}", d);
            Debug.WriteLine(string.Format("------------------------------"));
            Debug.WriteLine(string.Format("today is {0}", d));
            Debug.WriteLine(string.Format("------------------------------"));
            */

            int[] testArray = {4, 7, 4, 2, 7, 3, 7, 8, 3, 9, 1, 9};
            int[] maxValIndices;

            int maxVal = Maxima(testArray, out maxValIndices);

Console.WriteLine("Maximum value {0} found at element indices:",maxVal);
foreach (int index in maxValIndices)
{
Console.WriteLine(index);
}
Console.ReadKey();
        }
// function
        static int Maxima(int[] integers, out int[] indices)
{
Debug.WriteLine("Maximum value search started.");
indices = new int[1];
int maxVal = integers[0];
indices[0] = 0;

         /*   Trace.Assert(maxVal == 10, "Variable out of bounds.",
"Please contact vendor with the error code Arab01.");*/
int count = 1;
Debug.WriteLine(string.Format(
    "Maximum value initialized to {0}, at element index 0.", maxVal));

for (int i = 1; i < integers.Length; i++)
{
Debug.WriteLine(string.Format("Now looking at element at index {0}.", i));
// ""إذا كانت الآي تساوي ٥ أو تساوي ٨ اكتب الجملة التالية
Debug.WriteLineIf(i == 5 || i == 8,string.Format("hi danti"));

if (integers[i] > maxVal)
{
maxVal = integers[i];
count = 1;
indices = new int[1];
indices[0] = i;

    Debug.WriteLine(string.Format(
"New maximum found. New value is {0}, at element index {1}.",maxVal, i));
 }
else
{
if (integers[i] == maxVal)
{
count++;
int[] oldIndices = indices;
indices = new int[count];
oldIndices.CopyTo(indices, 0);
indices[count - 1] = i;
Debug.WriteLine(string.Format("Duplicate maximum found at element index {0}.",
i));
}
}
}
Trace.WriteLine(string.Format("Maximum value {0} found, with {1} occurrences.",
maxVal, count));

            Debug.WriteLine("Maximum value search completed.");
return maxVal;
}
    }
```

VC7 Part1 CH7_2========== 00:18:35 or 00:19:10

أولاً: طريقة debug:

مثال بسيط على التعامل مع **debug** وأهميته.

```
class Program
    {
        static void Main(string[] args)
        {
            int ro = 1, x= 5;

            for (int i = 1; i <= x; i++)
            {
                ro = ro * i;

                                هنا يوجد شرط إذا كانت الآي بثلاثة اكتب قيمتها          //
                Debug.WriteLineIf(i==3,string.Format("ro={0}, i={1}", ro, i));
                                                لحساب الفيكتوريل                       //
                Debug.WriteLine(string.Format("ro={0}, i={1}",ro,i));

} // tracepoint  من الممكن أن نقف هنا لعمل

            Console.WriteLine(string.Format("factorial of 5 is {0}",ro));

            Console.WriteLine("ok");
            Debug.WriteLine("debug mood");
            Trace.WriteLine ("trace mood");
        }
```

هذا ما سوف يظهر في شاشة **output** عند تشغيل البرنامج بطريقة **debug**

مع العلم أن هذه المعلومات لا تظهر للمستخدم ، وتظهر للمبرمج فقط للتأكد من صحة الأرقام.

```
ro=1, i=1
ro=2, i=2
ro=6, i=3
ro=6, i=3
ro=24, i=4
ro=120, i=5
debug mood
trace mood
```

ثانيًا: طريقة **trace point:**

وهي تعتبر من إحدى طرق الـ **debug**، وتعمل عند تشغيل البرنامج بطريقة الـ **debug**

وطريقة تشغيلها كالآتي: تختار السطر الذي تريد معرفة نتائجه ، ثم كليك يمين ← **Breakpoint** ← **tracepoint** .

ونكتب في المربع الأول الجملة التي نريد أن نعرضها مثلاً:

Result now is {ro} and {i}

وكذلك نستطيع وضع شرط **condition**

ثالثًا : طريقة **Breakmode** :

وطريقة تشغيلها أيضًا كالآتي: تختار السطر الذي تريد معرفة نتائجه، ثم تضغط كليك يسار لتظهر أمامك نقطة حمراء ثم تشغيل الـ **debug**.

أنواع الشاشات:

١-الشاشة **local** :

تعتبر شاشة محلية تقوم بمراقبة النتائج التي توجد ما بين علامتي المجموعة **{}** ولا يمكن أن تراقب نتائج كل البرنامج.

٢-الشاشة **Auto**:

الهدف منها مراقبة نتائج كل البرنامج مهما كانت عدد المتغيرات أو عدد الأسطر .

٣-الشاشة **watch** :

تعتبر الشاشة **watch** بمثابة شاشة مراقبة ،وتستطيع من خلاله أن تعطي البرنامج قيم أخرى لاختبار النتائج.

٤-شاشة **call stack**

هذه الشاشة تعرف من خلالها أين أنت موجود في البرنامج ورقم السطر.

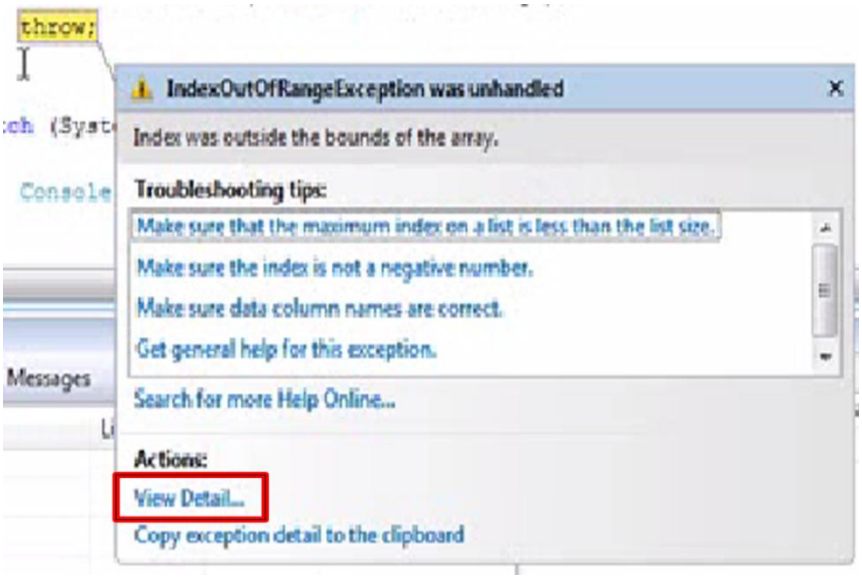
VC7 Part2 CH7_4==========

رابعًا : **error handling** اصطياد أو منع الأخطاء: **طريقة try و catch**

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

            int[] a = { 0, 1, 2 };

            int w = 0;
            try
            {
                                                    هذه جملة الخطأ                //
            //a[6] = 9;

                                    الخطأ الثاني لا يجوز القسمة على صفر          //

                Console.WriteLine(8 / w);
                }

            catch (System.IndexOutOfRangeException e)
            {
                Console.WriteLine("catch 1 is " + e.Message);
            }

            catch (System.DivideByZeroException e)
            {
                Console.WriteLine("catch 2 is " + e.Message);
            }
            finally
            {
                Console.WriteLine("there are an error");

            }
        }
```

إذا أردت معرفة أنواع الأخطاء فيمكنك ذلك من **deubg ← exceptions**

أو يمكنك مباشرة معرفة نوع الخطأ تختار **view Detail..** كما في الصورة:



سوف نتحدث بعد ذلك عن

Object Oriented Programing

Interface

يدعم العديد من الكلاسات

لتفريغ الذاكرة يمكن ذلك من خلال **disposable**

ويكون الكود هكذا:

```
< ClassName > < VariableName > = new < ClassName > ();
...
Using ( < VariableName > )
{
...
}
```

أو تضع **using** في البداية هكذا:

```
using ( < ClassName > < VariableName > = new < ClassName > ())
{
...
}
```

وبالتالي يقوم بعملية تحرير للذاكرة.

Interface: وتعني توصيل أو اتصال.

تجميع مجموعة من المعلومات في **methode** ثم تلحقها بالكلاس الذي تريده أو أكثر من كلاس لأن **Interface** يدعم أكثر من كلاس.

الوراثة: **Inheritance**:

يمكن توريث الكلاس الأب إلى الإبن حتى يجعل الابن يحتوي على جميع خصائص الأب.

إذا كان الكلاس الأب من نوع **sealed**

تعدد الأشكال : **polymorphism**:

الهدف منه اختصار الكود.

وهذا هو المثال

```
Cow myCow = new Cow();
Chicken myChicken = new Chicken();
myCow.EatFood();
myChicken.EatFood();
```

عرف تلقائيًا أن **MyCow** لها **EatFood** خاص بها. و **myChicken** لها **EatFood** آخر.

العلاقات بين **Relationships Between Objects :object**:

تعتبر العلاقات بين الأوبجكت من نوعين:

1- Containment://احتواء

بمعنى أنه يكون هناك صلة بين كلاسين ، الأولى يستطيع التدخل في عمل الثاني بمعنى أنه يستطيع أن يضيف تغييرات أو معلومات على الثاني.

وهو يشبه الوراثة ولكن الفرق أن الثاني يكون عنده قدرة على التحكم في الأول والأول يتحكم في الثاني على حسب ما نريد.

2- Collections://مجموعات

Operator overloading:

ممكن ايضًا من خلاله اختصار الكود

مثال:

```
if (cowA.Weight > cowB.Weight)
{
...
}
```

ممكن بدلاً من ذلك اختصار الكود بهذا الشكل.

```
if (cowA > cowB)
{
...
}
```

بداية الفيديو #//VC8 Part1 CH8_1==========

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ((Button)sender).Text = "clicked";
            //button1.Text = "click";

            Button newButton = new Button();


            //newbutton.Click += new EventHandler (newbutton_Click);

            newButton.Click += new EventHandler(newButton_Click);

            newButton.Text = "newButton";
            Controls.Add(newButton);
        }
        private void newButton_Click(object sender, EventArgs e)
        {
            ((Button) sender).Text= "cllicked";
            //newButton.Text = "clicked";

            MessageBox.Show(" newButton.Click += new
EventHandler(newButton_Click) good");


          }
```

بداية الفيديو #//VC10 Part1 CH9_1==========

سوف نتحدث عن الكلاس **Class**

الكلاس الافتراضي يكون من نوع **internal** وتعني داخلي، يمكن قراءته في المشروع فقط .

```
internal abstract class abstra
        {

        }

        class sun : abstra
        {
        }
        static void Main(string[] args)
        {
//    abstra ref = new abstra (); لا يمكن قرائتها
          }
```

أما إذا جعلت الكلاس من نوع **public** فيمكن قراءته في المشاريع الأخرى.

```
public class abstra1
        {
        //يمكن قراءته داخل المشروع والمشاريع الأخرى أيضًا لأنه عام
        }
```

* من الممكن أن يكون الكلاس من نوع **abstract** أي لا يمكن استنساخه عن طريق **new** لكن يمكن ان نشتق أو (نورث) من **abstract** كلاس آخر كما نرى.

```
abstract class indian
        {
        }

        static void Main(string[] args)
        {
        indian ind = new indian(); لا يمكن ذلك فلا يمكن قراءتها
        }
```

* ملحوظة: **Abstract** ممكن أن يكون **internal** ومن ممكن أن يكون **Public**

* أما إذا كان الكلاس من نوع **sealed** فيعني أننا لا نستطيع اشتقاق أو استنساخ كلاس جديدة . أي أنها تجعل الأب عقيم لا يمكن الوراثة منه. لكن نستطيع عمل **new** كما نرى.

```
class Program
    {

        internal sealed class refaat
        {
        }

        static void Main(string[] args)
        {
         refaat refo = new refaat ();
        }
      }
```

* التوريث: الكلاس الابن لا يمكن أن يكون له إلا أب واحد. ولا يمكن للابن أن يجعل الابن الثاني أبًا له.

* لا يمكن توريث كلاس **public** (عام) من كلاس **internal** (داخلي) . لكن يمكن العكس.

ويمكن توريث **internal** إلى **Public** أو **public** إلى

```
class arab {}

        public class egypt : arab { }
               public من نوع egyptو internal من نوع arab بالطبع لا يمكن ذلك لأن //

            * لا يمكن أن يرث الكبير من الصغير لكن يمكن أن يرث الصغير من الصغير

public class arabchild : arabOk { }
        //public class egypt : arab {} لا يمكن أن يرث الكبير من الصغير
        class egypt : arab { }// يمكن أن يرث الصغير من الصغير
```

Modifier	Meaning
none or internal	Class accessible only from within the current project .
public	Class accessible from anywhere .
abstract or internal abstract	Class accessible only from within the current project, and cannot be instantiated, only derived from .
public abstract	Class accessible from anywhere, and cannot be instantiated, only derived from .
sealed or internal sealed	Class accessible only from within the current project, and cannot be derived from, only instantiated .
public sealed	Class accessible from anywhere, and cannot be derived from, only instantiated .

سوف نتطرق إلى موضوع **interface**:

لا يوجد في الـ **interface** لا **abstract** ولا **sealed**

interface يمكن أن يكون للإبن فيه عدة آباء، لكن **class** لا يمكن أن يكون للإبن غير أب واحد.

```
internal interface skill { }

 interface skills { }

 interface it : skill,skills  { }

 class im : it { }

 --------------------------
 class lion{}
 class tiger { }
 interface  jungle{ }
 interface  desert { }
 interface mountain { }

class wildcat: tiger , jungle , mountain { }   // هنا يمكن التوصيل بهذا الشكل


 class cat : tiger, jungle, mountain { }
```

أصبح **tiger** هو الأب ولا يمكن توريث أكثر من أب للكلاس **wildcat** (القط البري) ويمكن توصيل أكثر من **interface** كما شاهدنا.

//VC10 Part1 CH9_2========== بداية الفيديو

```
class Program
    {
        public abstract class Mybase { }

        class Myclass : Mybase { }


        interface Mybaseinterface { }
        public interface  myBaseinterface2 {}
        interface myinterface { }

        interface AllMyinterface : Mybaseinterface, myBaseinterface2 { }

        sealed class AllCompletleyclass : Myclass, myinterface { }

        static void Main(string[] args)
        {
            AllCompletleyclass opjec = new AllCompletleyclass();
      Console.WriteLine(opjec.ToString());
        }
    }
```

#//VC10 Part1 CH10_3========== بداية الفيديو

```
class human
        {
            string s;

            int age;

            //consructor 1
            public human ()
            {
                Console.WriteLine("you just initiate in inctance ");
            }

            //costructor 2

            public human(string  s1, int age1)
            {
                s = s1;
                age = age1;

                Console.WriteLine("{0}, {1} ",s, age);
             }
        }

        static void Main(string[] args)
        {
            human a = new human();// سوف يقوم بطباعة الكونستركتر الأول لأنه لا
يستقبل قيم

* a.s="mohamed";/ إذا لم يظهر حرف الإس فمعنى ذلك أن المتغير إس ليس ببلك
    human b = new human("mohamed", 23);//  هنا بحث عن الكونستركتر الذي يستقبل
قيمتين ووجده فطبع قيمتهم لأن السن١ هو السن و إس١ هي إس
//b.s = "refoo";
            //b.age = 29;
            //Console.WriteLine("name is {0}, age {1}",a.s,a.age);
            //Console.WriteLine("name is {0}, age {1}", b.s, b.age);
```

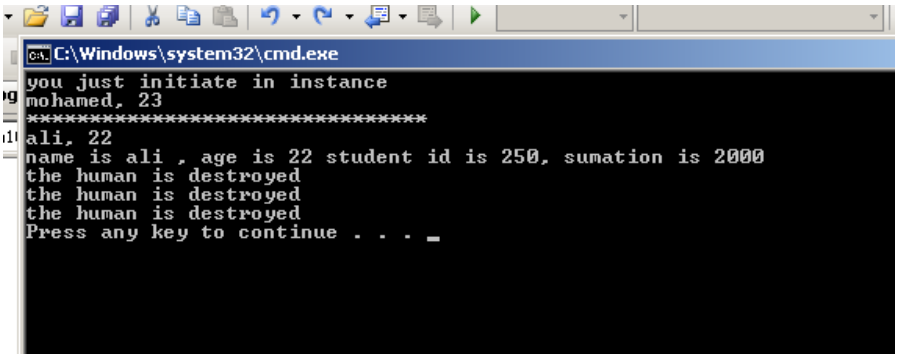
نتحدث عن نقطة مهمة وهي **incapsulation** أي التغليف والهدف منها الحفاظ على أمنية البيانات حتى لا نضع الأوبجكتس في الـ **Main** الخاص بالبرنامج . فنضع الأوبجكتس في الكلاس حتى تصبح مغلفة ويعطيني ذلك حماية أكثر للبرنامج.

كما نستطيع عمل **overloading** وتعني عمل أكثر من **construtor** (بناء تشييد) لنفس الكلاس.

الآن نريد تطوير البرنامج أكثر كما سوف نرى:

```
class Program
    {

        public class human
        {
            string s;
            int age;

            //consructor 1 المشيد أو الباني
            public human ()
            {
                Console.WriteLine("you just initiate in instance ");
            }


            //costructor 2 المشيد أو الباني

            public human(string  s1, int age1)
            {
                s = s1;
                age = age1;

                Console.WriteLine("{0}, {1} ",s, age);
             }

            // الهدف منها تدمير الأوبجكت حتى لا يحجز مساحة في الرام
            ~human()
            {
                Console.WriteLine ("the human is destroyed");
            }
        }
        // كلاس جديد يرث من الكلاس هيومن

        public class student : human// لا بد أن يكون الكلاس الأب هيومن بابلك
}
            int id;
            int sum; // مجموع سبع مواد دراسية مثلاً

            public student() { }
            public student(string s2 , int age , int id1, int sum1) :
base(s2,age)
            {
                id = id1;
                sum = sum1;
                Console.WriteLine("name is {0} , age is {1} student id is {2},
sumation is {3} ", s2, age, id1, sum1);
            }
```

```
        }

        static void Main(string[] args)
        {
            human a = new human لا الأول لأنه الكونستركتر بطباعة يقوم سوف//:()
يستقبل قيم
            human b = new human("mohamed", 23); هنا بحث عن الكونستركتر الذي //
يستقبل قيمتين ووجده فطبع قيمتهم لأن السن\ هو السن و\إ هي إس

            Console.WriteLine("**********************************");

            //a.s = "mohamed";
            //a.age = 23;

            //human b = new human();
            //b.s = "refoo";
            //b.age = 29;
            //Console.WriteLine("name is {0}, age {1}",a.s,a.age);
            //Console.WriteLine("name is {0}, age {1}", b.s, b.age);
.
            student c = new student("ali",22, 250,2000);


        }
    }
```



لاحظ أن **ali, 22** لها علاقة بالجملة **: base(s2,age)**

هنا أضاف الأستاذ دانتي معلومة أخرى وهي إكمال على المشيد الفارغ **studeng:**

```
public student():this ("mohamed",0,0,0) { }
```

وفي الـ **main** تكتب هذه التعليمة:

```
student d = new student ();
```

فسوف يقوم البرنامج بالذهاب إلى المشيد الفارغ **public student** وسوف يجد بجواره **this** فيبحث عن المشيد المناسب الذي يوجد به القيم اربعة قيم.

بداية الفيديو **//VC10 Part1 CH10_4==========**

تحدث عن **class view** و **object browser** وأهميتهم. وحتى تستطيع استحضار الأداتين تضغط من قائمة **view** على إحداهما.

وأهميتهما سهولة الوصول إلى الكلاسات والأوبجكتس .

```
class Program
    {
        class ararb
        {
            int x1;
            int x2;
            }

        class arab1 : ararb ,er
        {

        }
        interface er { }

        class b747h : b747 { }
        static void Main(string[] args)
        {

        }
    }
```

وهذه هي الكلاس التي تم إنشاءها وبجوارها العديد من الكلاسات

```
class Class1
    {

    }

    class b747 { }
    class b747b : b747 { }
    interface fly { }
    interface fly1 : fly { }
    //للي لوب
    class b747c : b747, fly ,fly1 { }
```

بداية الفيديو **//VC10 Part1 CH10_5 class liberary ==========**

قمنا بعمل مكتبة كلاس اخترنا بدلا من الكونسل أو بدلا من كلاس اخترنا هذه المرة **classLiberary** حتى تكون معنا مكتبة من الكلاس ويفضل عند كتابة اسم المكتبة عدم دون وضع نقطة أو وضع مسافة فارغة في الاسم.

* أنشأنا في هذه المكتبة كلاسين وسمينا الأول: **externalClass** والثاني: **internalClass**

الأول : **public** والثاني: داخلي.

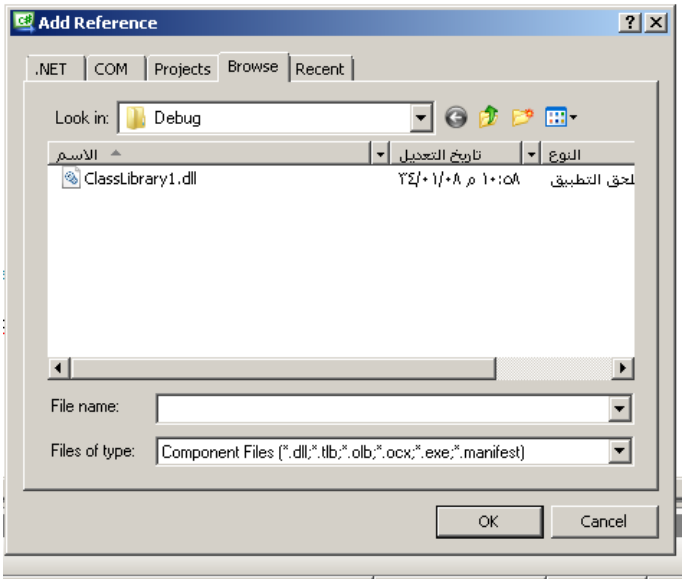
بعد ذلك لا ننسى القيام بعمل **build solution** وحفظ الكل.

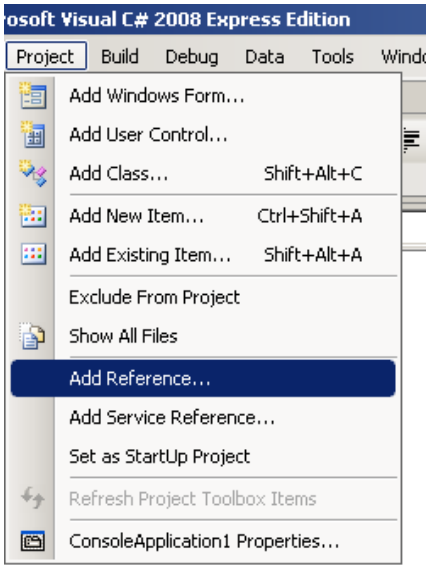
وسوف نشاهد كيف نستدعي المكتبة في البرنامج التالي:

بداية //VC10 Part1 CH10_5 project==========

الفيديو

الآن قمنا بعمل مشروع برنامج كونسل حتى نضع فيه المكتبة التي قمنا بإنشائها.





نقوم بإضافة المكتبة الجديدة في أعلى البرنامج.

```
using ClassLibrary1;
```

وبهذه الطريقة نستطيع أن نستدعي المكتبة

```
class Program
    {
        static void Main(string[] args)
        {
            externalClass a = new externalClass();
            Console.WriteLine("{0} ", a.ToString());

        }
```

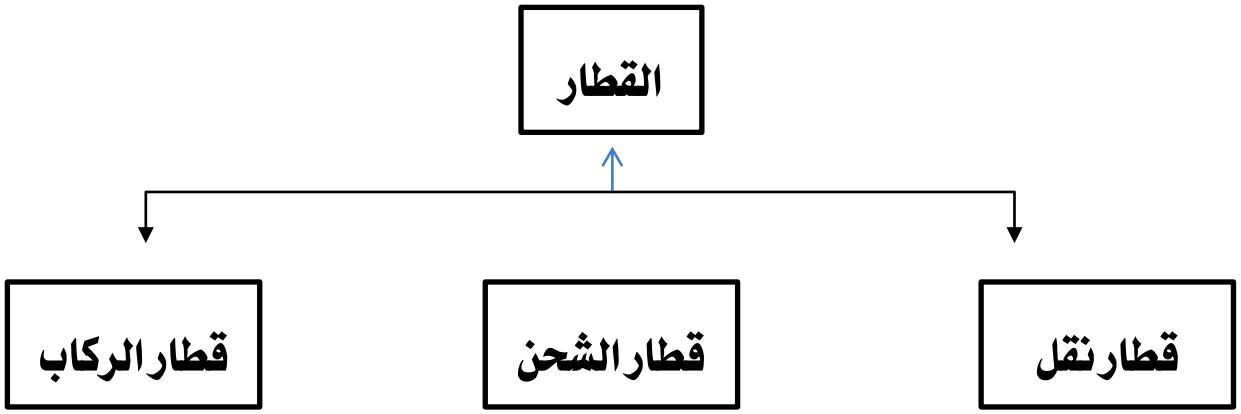
٠٠:٣٤:٣٧

بداية//VC10 Part1 CH10_6 project==========

متى أستخدم **abstract class** وأضع بداخله معطيات

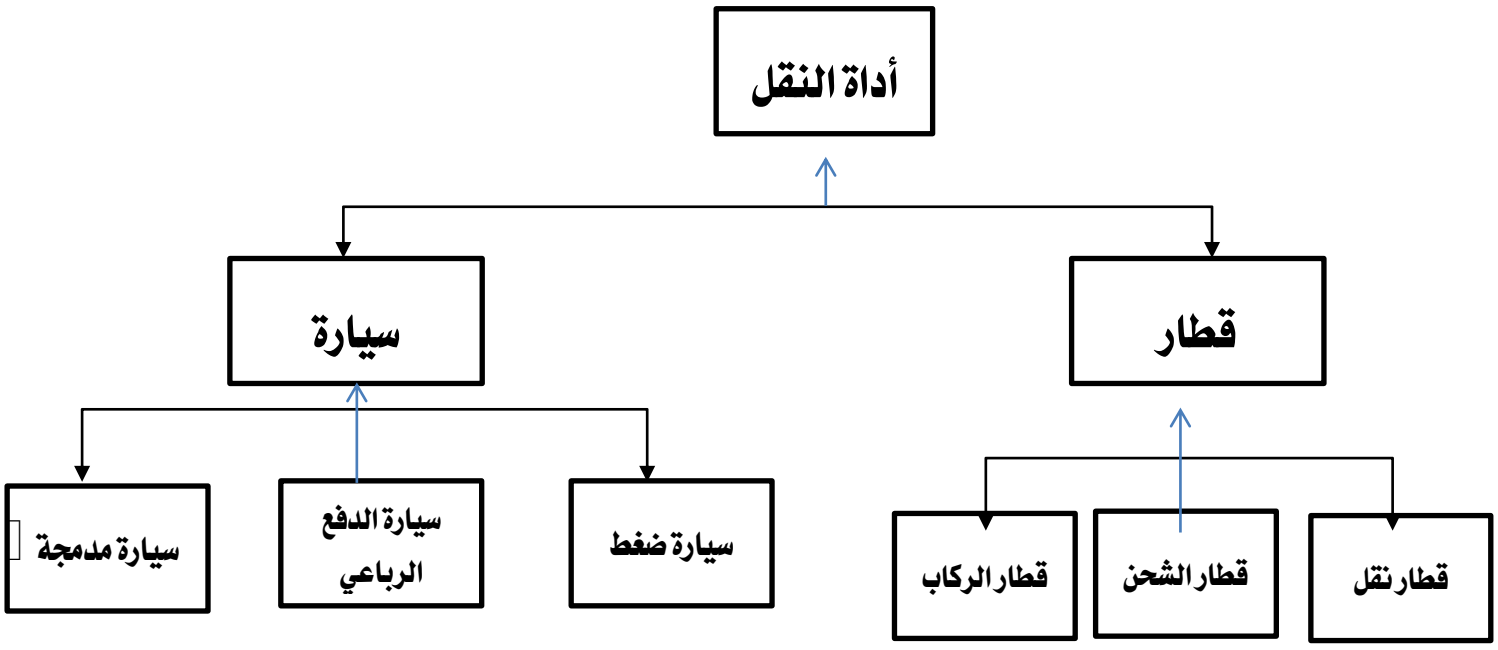
```
abstract class cat
     // سؤال متى استخدم ابستركت كلاس وأضع بداخله معطيات
     { }
     class kit : cat { }

     interface details { }// ومتى استخدم انترفيس
```

```
static void Main(string[] args)
{
                إذا أردنا عمل intstance من cat فلا يمكن ذلك لأنها abstract
    kit a = new kit();
}
```



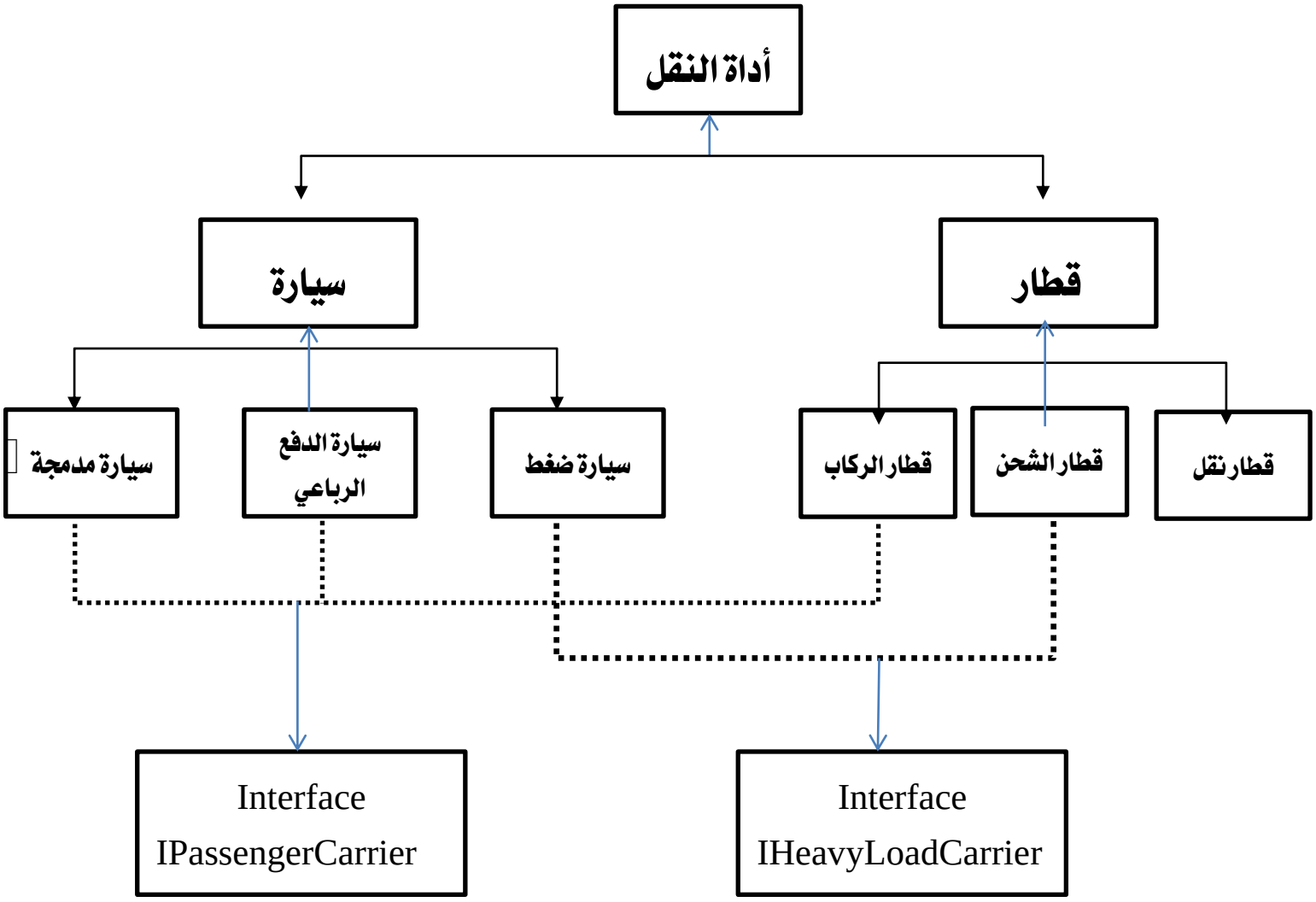
القطار يكون من نوع **abstract** وما أسفله (قطار نقل– قطار شحن – قطار الركاب) يكونون الأبناء .

هذه الطريقة أيضًا تعتبر من نوع **abstract** لأن الأبناء يرثوا من الآباء.



في هذا المثال سوف نستخدم **interface** وذلك لأن الابن «سيارة مدمجة» يشترك مع «سيارة الدفع الرباعي» و «قطار الركاب» في نفس الخصائص فهما جميعًا مشتركين في القدرة على

حمل جميع الركاب. وأيضًا «سيارة ضغط» و «قطار الشحن» مشتركين في نفس الخصائص وهي حمل الأشياء الثقيلة ، فلا بد من عمل **interface**. كما نرى في الشكل التالي:



:Struct type

Struct يكون من نوع **value Type**

أما الكلاس من نوع **reference Type**

عند عمل **instance** لكلاس سوف يؤشر **instance** على نفس المكان في الذاكرة .

لو كان عندي **class** اسمه `class MyClass{public int val}` وقمت بعمل **Instance** منه

objectA و **objectB** كما نشاهد:

```
MyClass objectA = new MyClass();
MyClass objectB = objectA;
```

فلو كانت قيمة :

```
objectA.val = 10;
objectB.val = 20;
```

وقمت بعمل

```
Console.WriteLine("objectA.val = {0}", objectA.val);// = 20
Console.WriteLine("objectB.val = {0}", objectB.val);// = 20
```

سوف تجد قيمة **objectA** = ٢٠ و **objectB** = ٢٠ أيضًا وذلك لأن آخر قيمة هي التي تحجز المساحة في الذاكرة.

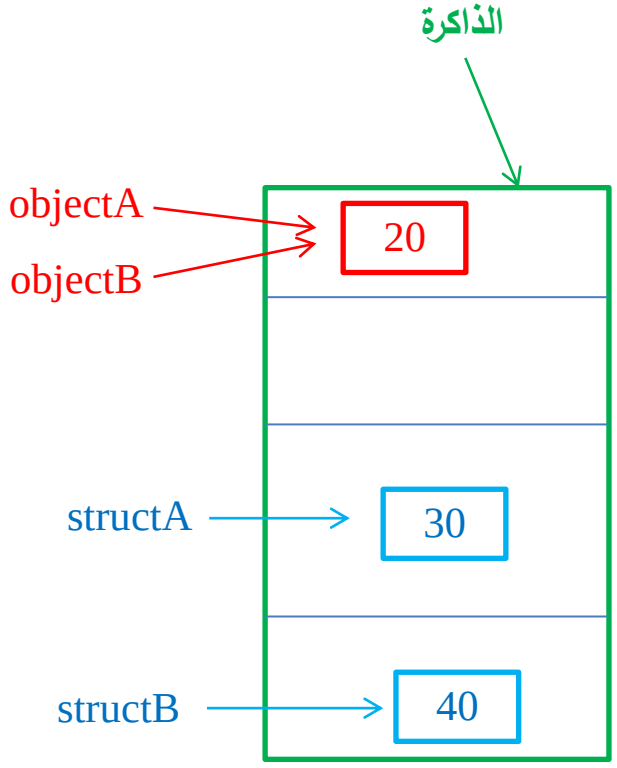
احفظ التالي كما هو:

Classes الكلاس : من نوع **ref type** يؤشر على نفس المكان في الذاكرة ، والحجم متغير.

Stract الاستراكت : من نوع **value type** يؤشر على أماكن مختلفة في الذاكرة ، والحجم متغير أيضًا.

String: من نوع **ref Type** تؤشر على أماكن مختلفة في الذاكرة والحجم متغير.

هذا مثال بسيط



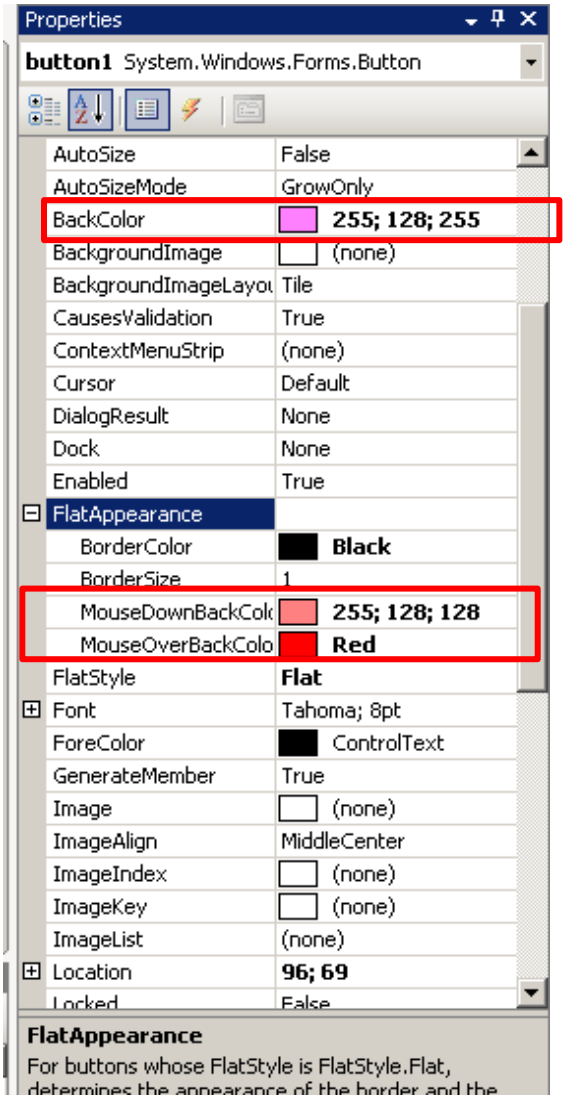
هذه التعليمة حصلت عليها من النت عسى تفيدكم.

للحصول على اسم الملفات الموجودة في البارتشن السي وحجمها

```
public void TestSize(string path)
{
DirectoryInfo di = new DirectoryInfo(path);
FileInfo[] fiArr = di.GetFiles();
// Directory Name
MessageBox.Show(di.Name);
foreach (FileInfo f in fiArr)
// File Name + " " File Size
// MessageBox.Show Will Use As Follow MessageBox.Show("Message","Message Title
(Header)","Button Type  ","Button Icon")
MessageBox.Show(f.Name + " " + f.Length);
}

        private void button1_Click(object sender, EventArgs e)
        {

            TestSize("C:\\");
      }
```

أهم الدروس المستفادة من شرح الأستاذ/ احمد فرحات

Messagebox.show("welcom"+ textbox1.text);



بالنسبة لزر الأمر **button**

* تستطيع في حالة كتابة عنوان له أن تختار **auto size** تجعلها **true** وذلك حتى يؤثر حجم الكتابة على حجم زر الأمر.

* من **style** **flat** اجعلها **flat** حتى يفتح لك **appearance** حتى عند المرور يأخذ الزر لون آخر.

٠٠:٣٦:٠٣