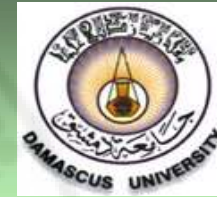


البرمجة بلغة C#

C# Programming

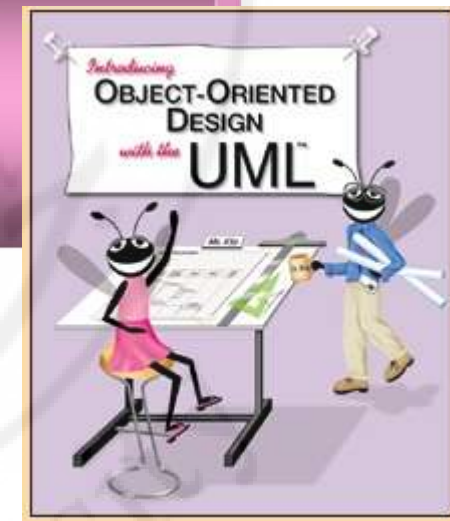


Chapter 1

Dr.Eng.M.Younes



C# Programming
Java Programming



Dr.M.Younes



المحاضرة (1) **C# Programming**

الصفوف في لغة C# Classes In C#

Chapter 1

Dr.Eng.M.Younes



C# Programming



Dr.M.Younes

محتويات الفصل الأول

- بنية الصف في C#
- محددات الوصول
- دراسة الصف student و الصف Point والصف Box
- دراسة الصف Time و الصف Stack
- الكلمة المحجوزة **this**
- التحميل الزائد للبواني
- الخصائص
- تجزئة الصف



ما هي البرمجة غرضية التوجه ؟

What's Object Oriented Programming

• تعتبر البرمجة غرضية التوجه طريقة قوية للتعامل مع المسائل البرمجية ، فقد ظهرت في كل نقطة من مراحل تطورها طريقة جديدة تساعد المبرمجين على التعامل مع البرامج المتزايدة التعقيد .

• رغم أن البرمجة البنيوية كانت تعطي نتائج ممتازة عندما تطبق على برامج متوسطة التعقيد ، إلا أنها كانت تفشل عند بعض النقاط حين يصل البرنامج إلى حد معين .

• لكتابة برامج أكثر تعقيداً كانت الحاجة إلى طريقة جديدة في العمل البرمجي وتلبية لهذه الحاجة فقد ظهرت البرمجة غرضية التوجه واختصاراً " OOP " .

• تأخذ البرمجة OOP أفضل ميزات البرمجة البنيوية وتضم إليها مبادئ قوية وجديدة تسمح بتنظيم البرامج بفعالية أكبر .

• تعتمد البرمجة OOP على تحليل المشكلة إلى أجزائها الأساسية بحيث يصبح كل عنصر أو جزء عبارة عن غرض object محتوي ذاتياً يضم تعليماته الخاصة والمعطيات العائدة إليه . وأصبح المبرمج يستطيع إدارة برامج أكبر .

سمات البرمجة غرضية التوجه

- تشترك جميع لغات البرمجة OOP بما فيها C# بثلاث سمات مميزة مشتركة وهي :

1. التغليف Encapsulation (الصفوف Classes) .

2. الوراثة Inheritance .

3. تعدد الأشكال Polymorphism .

التغليف

Encapsulation (Classes)

- يطلق مصطلح التغليف على عملية ربط البيانات ونص البرنامج الذي يتعامل معها وحمايتها من التداخل وسوء الاستخدام الخارجيين .
- نطلق على البيانات ونص البرنامج المرتبطين بهذه الطريقة اسم " **الغرض Object** " . فالغرض هو الآلية التي تدعم التغليف في الـ **OOP** .
- يمكن أن تكون البيانات أو نص البرنامج أو كليهما خاصة بهذا الغرض أو عامة .
- يمكن الوصول إلى البيانات ونصوص البرامج الخاصة في الغرض فقط من قبل أجزاء البرنامج داخل الغرض أما عندما تكون البيانات أو نصوص البرنامج عامة يمكن الوصول إليها من أجزاء البرنامج خارج الغرض .

الوراثة inheritance

- يطلق مصطلح الوراثة على العملية التي يكتسب غرض ما من خلالها على خصائص غرض آخر .
- أي يستطيع غرض ما أن يرث مجموعة عامة من الخصائص ثم يضيف إليها الميزات الخاصة به وحده .
- في الوراثة يستطيع صفأ ما أن يرث صفأ أساسياً وذلك بتحديد الصف الأساسي الذي ينتمي إليه بالإضافة إلى الصفات الخاصة التي تجعله مميزاً عن غيره .

تعدد الأشكال polymorphism

- تعدد الأشكال تمثل الميزة التي تسمح لاسم واحد أن يُستخدم لهدفين مترابطين أو أكثر لكنهما مختلفين تقنياً .
- أما لغة C# فتدعم تعدد الأشكال ويمكن استدعاء هذه التوابع بنفس الاسم ويحدد نوع البيانات المستخدم لاستدعاء التابع النسخة المحددة من التابع الواجب تنفيذها .
- تكمن فائدة تعدد الأشكال في أنها تساعد على تخفيف التعقيد بالسماح لواجهة واحدة أن تعرف بمجموعة عامة من الأفعال .
- تسمح طريقة تعدد الأشكال بالتعامل مع تعقيد أكبر من خلال إنشاء واجهة قياسية تشمل جميع الفعاليات المترابطة .

الصفوف في C#

- تعتبر الصفوف من أهم ميزات C# , فهي الآلية التي تستخدم لإنشاء الأغراض Objects .
- يصرح عن صف باستخدام الكلمة المحجوزة **class**
- البنية العامة للصف لها الشكل التالي :

بنية الصف في C#

- الصيغة العامة للصف هي :

```
class classname {  
    type variable1;  
    type variable2;  
    // ...  
    type variableN;  
    type func1( parameter-list )  
    {  
        // body of method  
    }  
}
```

المتحولات

التوابع

```
type func2(parameter-list) {  
    // body of method  
}  
// ...  
type funcN(parameter-list) {  
    // body of method  
}  
} //End class
```

التوابع

- يتكون الصف Class Name من :

القسم الأول : يتضمن أعضاء المعطيات Data Members للصف أو ما يسمى بالصفات Properties :

```
type variable1;  
// ...
```

```
type variableN
```

القسم الثاني : يتضمن طرائق الصف Class Methods (التوابع) أو ما يسمى بالأعمال :

```
type func1(parameter-list) {  
    // body of method  
}  
// ...
```

```
type funcN(parameter-list) {  
    // body of method  
}
```

- يبدأ الصف بالتعليمة **class** (الكلمة المحجوزة) ثم اسم الصف **class_Name** ثم جسم الصف بين القوسين { } .
- ثم المتحولات والتوابع المصرح عنها داخل الصف وتدعى بأعضاء الصف **class members** .
- تكون جميع المتحولات والتوابع المصرح عنها داخل الصف خاصة بهذا الصف .
- يتم الوصول إليها فقط من قبل الأعضاء الأخرى في هذا الصف .
- للتصريح عن الأعضاء الخاصة في الصف نستخدم الكلمة المحجوزة **private**
- للتصريح عن الأعضاء العامة في الصف نستخدم الكلمة المحجوزة **public** .
- يمكن الوصول إلى الأعضاء العامة بعد الكلمة **public** من قبل أي جزء آخر من البرنامج الذي يحتوي الصف أما **الأعضاء الخاصة** لا يمكن الوصول إليها إلا من قبل **الأعضاء العامة للصف نفسه** .
- فيما يلي مثال بسيط للتصريح عن صف اسمه **student** :

مثال (1)

```
using System;
class student {
    private int id ;
    private string student_name;
    private double credit ;
    public void setstudent ( int d , string s , double c ){
        id = d;
        student_name = s;
        credit = c;
    }
    public void print_stud ( ) {
        Console.WriteLine( "id : " + id + "\n " +
            student_name : "+student_name + "\n"+
            "credit : " + credit );
    }
} // End class
```

```
class test {  
static void Main(string[] args){  
  
    student ob = new student() ;  
    ob.setstudent( 230 , " Ali " , 85 ) ;  
    ob.print_stud ( );  
} //End main  
} // End class
```

Result

id : 230

student_name : Ali

credit : 85

Press any key to continue

المعامل new :

يستخدم هذا المعامل لإنشاء غرض جديد أو مصفوفة جديدة

```
student ob = new student() ;
```

- يبدأ الصف بالتعليمة **class** ثم اسم الصف **student** .
- ثم المعطيات الخاصة **private** والتوابع العامة **public** . وذلك ضمن القوسين { } وتسمى بجسم الصف .
- تدعى المحددات **private** و **public** بمحددات طريقة الوصول إلى الأعضاء . Member Access Specifies
- **يفيد التصريح عن المعطيات الأعضاء والتوابع الأعضاء بعد التعليمة public** في جعل هذه المعطيات أو التوابع متاحة للاستخدام من أي مكان ضمن البرنامج وتسمى بالمعطيات أو التوابع العامة أو واجهة الصف Interface of class , حيث يمكن الوصول إلى أي غرض من الصف student .
- يمكن استدعاء التابعين () setstudent و () print_stud من أي جزء من البرنامج لأنهما أعضاء عامة في الصف student .
- نسمي التابع () setstudent بتابع تعديل المعطيات ، حيث يقوم بتمرير المعطيات إلى الأعضاء الخاصة للصف .

- أما المعطيات الأعضاء والتوابع الأعضاء المصرح عنها بعد التعليمات `private` فهي متاحة فقط للتوابع الأعضاء العامة المرتبطة بالصف وتسمى بالمعطيات الخاصة.
- نسمي التابع العضو الذي يحمل اسم الصف **بالتابع الباني (البناء) Constructor** المرتبط بذلك الصف.
- يقوم التابع الباني بتمرير المعطيات إلى المعطيات الأعضاء لكل غرض من أغراض الصف.
- حيث يتم استدعائه آلياً عند إنشاء الغرض.
- وسوف نرى فيما بعد **عدة توابع بانية لصف واحد** ويتم ذلك من خلال التحميل الزائد للتابع كما سنرى لاحقاً.
- إذا لم نكتب أي باني فإن المترجم يقوم تلقائياً بتقديم باني ويسمى الباني الافتراضي.
- أما إذا كان هناك أي باني فإن المترجم لا يقدم أي باني افتراضي.

- نظراً لأننا كتبنا الصف student بدون باني فإن المترجم يقوم بتقديم باني افتراضي تلقائياً وبدون وسيط .
- إذا لم نكتب أي بانٍ فإنه يجري تزويدنا ببانٍ افتراضي بحيث يمكن لمستخدم الصف أن يصرح عن أغراض .
- أما إذا احتوى الصف على بانٍ عادي ، لا يمكن التصريح عن أغراض بدون وسطاء إلا إذا احتوى الصف على بانٍ افتراضي بالإضافة إلى الباني العادي .
- الأفضل أن نكتب بانٍ افتراضي خاص بنا بدلاً من الاعتماد على المترجم .
- ويمكن أن يكون الباني الافتراضي بسيطاً بالقدر الذي نريده , ويمكن أن يكون خالياً من التعليمات . وإن سلوك الباني الافتراضي المزود من قبل المترجم هو إسناد أصفاراً إلى جميع أعضاء المعطيات .
- يجري استدعاء الباني الافتراضي عند التصريح عن غرض بدون قائمة وسطاء .
- نُعيد كتابة الصف student بوجود بانٍ افتراضي

```

using System;
class student {
    private int id ;
    private string student_name;
    private double credit ;
public student() {
    id = 0;
    student_name = "\0 ";
    credit = 0;
}
    public void setstudent ( int d , string s , double c ){
        id = d;
        student_name = s;
        credit = c;
    }
    public void print_stud ( ) {
        Console.WriteLine( "id : " + id + "\n " +
            student_name : "+student_name + "\n"+
            "credit : " + credit );
    }
}
} // End class

```

التابع الباني constructor

- نسمي التابع العضو الذي يحمل اسم الصف بالتابع الباني المرتبط بذلك الصف .
- إن التابع () student الذي يحمل اسم الصف هو عبارة عن التابع الباني والذي يستدعى عند إنشاء أغراض من الصف student ولا يعيد أي نوع .
- التابع الباني ليس له نوع من المعطيات ولا يعيد قيمة وليس من نوع void .
- ويتم استدعاء التابع الباني عند إنشاء أي غرض object في الصف بشكل آلي (تلقائي) , وذلك من قبل المترجم , ويقوم بتمرير المعطيات إلى الأعضاء الخاصة عند إنشاء الأغراض , لذلك توضع جميع عمليات التهيئة التي يحتاجها الغرض في هذا التابع .

- التوابع البانية تشبه التوابع الأخرى وتستطيع أن تأخذ وسيطاً أو أكثر .

- أما التابع setstudent

```
public void setstudent ( int d , string s , double c ){  
    id = d;  
    student_name = s;  
    credit = c;  
}
```

- يُسمى هذا التابع بتابع التعديل أي تابع تعديل المعطيات وهو يُشبه التابع الباني الذي يحتوي على وسطاء ، فهو بالإضافة لتمرير المعطيات إلى الأعضاء الخاصة يقوم بتعديل قيم هذه المعطيات في أي وقت نحتاجه .
- نُعيد كتابة الصف student بوجود بانٍ افتراضي اني وبانٍ بوسطاء والذي نسميه الباني العادي ، بدون أن يحتوي الصف على تابع تعديل المعطيات .

مثال (2)

```
using System;
public class student {
    private int id;
    private string student_name;
    private double credit;
    public student(){
        id = 0;
        student_name = "--- ";
        credit = 0;
    }
    public student(int d, string s, double c) {
        id = d;
        student_name = s;
        credit = c;
    }
    public void print_stud() {
        Console.WriteLine("id : " + id + "\n" + "student_name : " +
            student_name + "\n" + "credit : " + credit);
    }
} //end class student
```

```
public class test {  
    public static void Main(string[] args) {  
  
        student ob0 = new student();  
        student ob = new student (230, " Ali ", 85.8) ;  
  
        ob0.print_stud();  
        ob.print_stud();  
    }//End main  
}// End class
```

RESULT

id : 0

student_name : ---

credit : 0

id : 230

student_name : Ali

credit : 85.8

Press any key to continue

التابع المدمر Destructor

- التابع المدمر هو تابع بدون نوع وبدون وسطاء ولا يعيد قيمة و يحمل هذا التابع نفس اسم الصف مسبقاً بالإشارة (~) .
 - إن دور هذا التابع هو عبارة عن دور متمم لدور التابع الباني ويتم استدعاه عند تدمير الغرض (تحرير الذاكرة المخصصة للغرض) .
 - يختلف التابع المدمر في لغة C# بشكل كبير عن المدمر في لغة C++ وهذا بسبب وجود جامع النفايات الذي يؤدي إلى الاختلافين التاليين :
1. يقوم جامع النفايات garbage collector بشكل تلقائي بتنظيف الذاكرة لذلك يُستخدم المدمر في لغة C# فقط من أجل المصادر غير الذاكرة .
 2. إن استدعاء المدمر غير حتمي ، حيث يقوم جامع النفايات باستدعاء مدمر غرض ما عندما يقرر أنه لم يُشار إليه منذ فترة طويلة وبالتالي لا يمكننا التنبؤ بالفترة الزمنية التي سيتم عندها استدعاء المدمر .

- لا يأخذ التابع المدمر أي قيمة أو وسيط ولا يعيد أي قيمة وسبب ذلك هو أنه لا توجد آلية لتمرير المتحولات إلى الغرض أثناء تدميره . ويكون لكل صف تابع مدمر وحيد فقط وهو لا يقبل عمليات التحميل الزائد عليه .
- لا يعتبر المدمر ضرورياً إذا كان الصف يستخدم مقداراً قليلاً من الذاكرة لأن جامع النفايات `garbage collector` سيقوم بشكل اعتيادي بتحرير هذه المصادر أما عند استخدام شيفرة تشغل مساحة كبيرة من الذاكرة كمكونات COM فقد تتطلب إدارتها وجود تابع مدمر .
- لا يقبل المدمر التحميل الزائد و لكل صف مدمر وحيد فقط .
- نُعيد كتابة الصف `student` بوجود التابع المدمر.

جامع النفايات

Garbage Collector

- يقوم جامع النفايات Garbage Collector بتحرير فضاء الذاكرة المستخدم من قبل الأغراض والتي أصبحت غير مطلوبة وذلك خلال فترات زمنية منتظمة يقوم جامع النفايات بفحص مناطق الذاكرة الديناميكية بحثاً عن الأغراض , ويُعنون الأغراض التي لها مرجع referenced بعد أن تُكتشف جميع التوابع الممكنة والمؤدية إلى الأغراض , فإن الأغراض غير المعنونة تصبح معروفة وتُجمع ليتم التخلص منها .
- اذاً : يتم تنظيف الأغراض (أي تحرير فضاء الذاكرة) من خلال جامع

النفايات .

مثال (3)

```
using System;
public class student {
    private int id;
    private string student_name;
    private double credit;
    ~student() {
        Console.WriteLine("Destructor : " );
    }

    public student(int d, string s, double c) {
        id = d;
        student_name = s;
        credit = c;
    }

    public void print_stud() {
        Console.WriteLine("id :{0} \n student_name :{1} \n" credit :{2} ",
            id,student_name, credit);
    }
} //end class student
```

```
public class test {  
    public static void Main(string[] args) {  
  
        student ob0 = new student();  
        student ob = new student (230, " Ali ", 85.8) ;  
  
        ob0.print_stud();  
        ob.print_stud();  
    }//End main  
}// End class
```

RESULT

id : 230

student_name : Ali

credit : 85.8

Destructor :

Press any key to continue ...

التحميل الزائد للبواني (تعدد البواني)

- إن تمرير المتحولات إلى التابع الباني عملية سهلة وبسيطة وذلك بإضافة الوسيط المناسب إلى تصريح تعريف التابع الباني , ثم تعين قيمة الوسيط عند التصريح عن الغرض من الصف .
- تعدد البواني هو إعادة استخدام نفس اسم التابع الباني ضمن سياقات مختلفة أحدهما بدون وسيط أو بوسيط واحد أو اثنين أو ثلاثة ... الخ .
- فإن المترجم يعرف من خلال السياق أيّاً من البواني سيُطبق .
- بالنسبة للصف student :
- من الضروري فقط وجود بانيتين أحدهما افتراضي بدون وسطاء والثاني يأخذ ثلاثة وسطاء (في مثالنا) .

لنأخذ الصف student

```
class student {  
    student ( ) {.....} // تابع باني افتراضي بدون وسيط  
    student ( int d ) { ..... } // تابع باني بوسيط واحد  
    student ( int d , string s ) { ..... } // تابع باني بوسيطين  
    student ( int d , string s , double c ) { ..... }  
                                                // تابع باني بثلاث وسطاء  
} // End Class .
```

تشبه البواني في لغة C# البواني في لغة C++ ولغة Java تماماً .

- ما هو الباني ؟

الباني هو كتلة من التعليمات التي يتم تنفيذها عند إنشاء أي غرض من الصف الذي يتضمن هذا الباني . وهي أول ما يتم تنفيذه عند استخدام new لإنشاء الأغراض .

- كتابة البواني :

جميع صفوف C# لها بوان تستخدم لتهيئة الأغراض الجديدة التي تنتمي لهذا الصف ، ويتميز الباني بأن له نفس اسم الصف ، ولا يتضمن نمط إعادة ولا حتى void .

محددات الوصول

- إن مستوى الوصول هو الذي يسمح لنا بأن نتحكم بوصول الصفوف الأخرى إلى المتحول العضوي من خلال واحد من خمسة مستويات وصول هي :
private , **protected** , **public** , **internal** , **protected internal**
- إن إحدى فوائد الصفوف أنها تستطيع حماية المتحولات الأعضاء وكذلك وتوابع الأعضاء من أن تصل إليها الأغراض الأخرى .
- جميع محددات الوصول في لغة **C#** تشبه محددات الوصول في لغة **C++** ولغة **Java** .

1. محدد الوصول private :

هو يشبه محدد الوصول في لغة C++ , وهو يعتبر جميع المتحولات خاصة في الصف A ويمكن الوصول إليها من أي مكان داخل الصف , ولا يمكن الوصول إليها من خارج هذا الصف .

2. محدد الوصول protected :

هو يشبه محدد الوصول في لغة C++ , يستطيع الصف A الذي يحتوي هذه المتحولات الوصول إليها وكذلك الصفوف المشتقة (الوارثة) من هذا الصف التي هي في نفس الحزمة .

3. محدد الوصول public :

إن محدد الوصول public هو الأسهل من بين جميع محددات الوصول وهو يشبه محدد الوصول في لغة C++ , وإن أي صف لأي حزمة يستطيع الوصول إلى هذه الأعضاء , وهذه الأعضاء مرئية داخل هذا الصف وجميع الصفوف من خارج هذا الصف .

4. محدد الوصول internal :

تكون الأعضاء التي تنتمي للصف A والمعرفة بالمحدد **internal** مرئية للتوابع داخل تجميعية هذا الصف . والتجميعية Assembly عبارة عن مجموعة ملفات تبدو للمبرمج كملف تنفيذي أو DLL وحيد .

5. محدد الوصول protected internal :

تكون الأعضاء التي تنتمي للصف A والمعرفة بالمحدد **protected internal** مرئية لتوابع الصف A وجميع الصفوف المشتقة من الصف A وكذلك جميع الصفوف المعرفة داخل تجميعية الصف A .

دراسة الصف **Box**

- البرنامج التالي يوضح بناء صف اسمه **Box** الذي يحسب حجم متوازي المستطيلات ابعاده : **depth , height , width** وهي متحولات خاصة من نوع **double** ، ويحتوي الصف على :
 - باني افتراضي لاسناد القيمة 1 للمعطيات الخاصة .
 - باني عادي بثلاثة وسطاء لتمرير المعطيات للاعضاء الخاصة .
 - تابع طباعة اسمه **(volume من نوع void)** لحساب وطباعة حجم متوازي المستطيلات .
- ويحتوي البرنامج على صف اختباري اسمه **BoxTest** يحتوي على التابع **(main)** . ثم أنشئ أغراضاً في هذا التابع لاستدعاء التوابع اللازمة وطباعة النتائج .

مثال (4)

```
using System;
    public class Box {

        private double height , width ,depth;

        public Box ( )
        {
            width = 1 ;height = 1 ;depth = 1 ;
        }
        public Box ( double w ,double h ,double d )
        {
            width = w ;height = h ;depth = d ;
        }
        // display volume of a box
        public void volume( ) {
            Console.Write("Volume is ");
            Console.WriteLine(width * height * depth);
        }
    } //End class Box
```

```
public class BoxTest {  
    public static void Main(string[] args) {  
        Box mybox0 = new Box ();  
        Box mybox = new Box ( 10,20,15);  
        Box mybox1 = new Box ( 3,6,9);  
        // display volume of initial box  
        mybox0 .volume( );  
        // display volume of first box  
        mybox .volume( );  
        // display volume of second box  
        mybox1 .volume( );  
    } //end main  
} // End class BoxTest
```

RESULT

Volume is 1

Volume is 3000

Volume is 162

Press any key to continue . . .

دراسة الصف Point

- اكتب برنامجاً بلغة C# يتضمن:
- صف اسمه Point يحتوي على متحولين من نوع private من النمط int وهما: x , y. ويحتوي الصف على باني افتراضي لإعطاء قيم ابتدائية للأغراض عند انشائها ويحتوي على باني عادي لتمرير المعطيات وعلى تابع طباعة printpoint لطباعة احداثية النقطة الموجبة ، ويحتوي على تابع مدمر يطبع رسالة ما عند استدعائه ، حيث الصف يعرف دوماً نقطة موجبة.
- أنشئ صفّاً آخرّاً يتضمن التابع Main اسمه PointTest لإنشاء أغراضاً لاستدعاء التوابع اللازمة والحصول على النتائج .

مثال (5)

```
using System;
class Point
{
    private int x, y ;
    public Point ( ){ x=0; y=0 ;}
    public Point(int new_x , int new_y ) {
        if ( new_x <0 )
            new_x *= -1;
        if ( new_y <0 )
            new_y *= -1;
        x=new_x ;
        y=new_y ;
        Console.WriteLine("This is a Destructor ");
    }
    ~Point() { Console.WriteLine("This is a Destructor "); }
    public void printPoint( ) {
        Console.Write ("Point is ");
        Console.WriteLine( "( "+x+ " , " +y  + ")" );
    }
} //End class Point
```

```

public class PointTest {
    static void Main(string[] args)
    {
        Point mypoint0 = new Point();
        Point mypoint1 = new Point(10, -20);
        Point mypoint2 = new Point(-3, 6);
        // display initial Point
        mypoint0.printPoint();
        // display first Point
        mypoint1.printPoint();
        // display second Point
        mypoint2.printPoint();

    } // End main
} // End class PointTest

```

RESULT

This is a constructor mypoint0

This is a constructor mypoint1

This is a constructor mypoint2

Point is (0 , 0)

Point is (10 , 20)

Point is (3 , 6)

This is a Destructor mypoint2

This is a Destructor mypoint1

This is a Destructor mypoint0

Press any key to continue . . .

- كيف يتم استدعاء التوابع البانية والمدمرة لعدد من الأغراض ؟
- نستج من المثال السابق ما يلي :

تُستدعى البواني في البرنامج حسب ترتيب إنشائها ، أي يُستدعى الباني للغرض الأول **mypoint1** ثم للغرض الثاني **mypoint2** ، ثم للغرض الثالث **mypoint3** .

أما المدمرات عكس ذلك . أي تدمر الأغراض ابتداءً من الغرض الأخير حتى الغرض الأول بشكل مرتب، أي يتم تدمير الغرض الثالث **mypoint3** ثم الغرض الثاني **mypoint2** ، ثم الغرض الأول **mypoint1** .

الكلمة المحجوزة this

- حتى نشير إلى عضو ما من أعضاء الغرض نقوم بذكر اسم الغرض متبوعاً بنقطة ثم اسم العضو المراد ذكره :

```
int getx( ) { return x ; }  
ob.getx( )
```

- يستدعي الغرض ob التابع getx() فينتقل التنفيذ إلى التابع getx() الذي سيجد العضو البياني x دون اسم الغرض , عندها يكون الغرض ob هو الذي قام باستدعاء التابع getx() وهو مالك هذا العضو .
- لكن في بعض الأحيان نحتاج لأن نميز بين عضو معطيات (متحولات أعضاء الصف) ووسيط لتابع لهما نفس الاسم وخاصة التابع البياني وتابع تعديل المعطيات , كما هو موضح في المثال التالي :

- مثال الصف Point

```
public class Point {  
    private int x , y;  
  
    public Point(int x, int y) {  
        x = x;  
        y = y;  
    }  
    . . . . .  
} //End Class Point
```

- في التابع الباني (أو تابع تعديل المعطيات) هل أسند عضو المعطيات إلى وسيط التابع أم أسند الوسيط إلى العضو المعطيات ؟

- لكي نحل هذا الغموض يمكننا استخدام الكلمة المحجوزة **this** والتي تشير إلى الغرض الذي قام بالاستدعاء .

```
public class Point {  
    private int x , y;  
    public Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    . . . . .  
} //End Class Point
```

- بهذا الشكل نستطيع أن نقول بأن أعضاء المعطيات (وهي المسبوقة بـ this) هي التي تم اسناد القيم إليها .

- بشكل عام يفضل المبرمجون القيام باستخدام الكلمة المحجوزة **this** سواءً أكان هناك تماثل بين أسماء الأعضاء وأسماء الوسطاء أو لم يكن , لأن ذلك يجعل النص البرمجي أكثر وضوحاً .
- يبين المثال التالي لصف **Point** وكيفية استخدام الكلمة المحجوزة **this** .

مثال (6)

```
using System;
class Point {
    private int x, y;
    public Point(int x, int y) {
        if ( x < 0)
            x *= -1;
        if ( y < 0)
            y *= -1;
        this.x = x; // set "this" object's x
        this.y = y; // set "this" object's y
    }
    public void printPoint()
    {
        Console.Write("Point is ");
        Console.WriteLine("( {0} , {1} ) ",x,y);
    }
} //End class Point
```

```
public class PointTest {  
    static void Main(string[] args) {  
        Point mypoint1 = new Point(-5, -10);  
        Point mypoint2 = new Point(10, -20);  
        Point mypoint3 = new Point(-3, 6);  
        // display first Point  
        mypoint1.printPoint();  
        // display second Point  
        mypoint2.printPoint();  
        // display third Point  
        mypoint3.printPoint();  
  
    } // End main  
} // End class PointTest
```

RESULT

Point is (5 , 10)

Point is (10 , 20)

Point is (3 , 6)

Press any key to continue ...

دراسة الصف Time

- اكتب برنامجاً بلغة C# للصف Time يقوم بما يلي :
- إدخال وطباعة الوقت (hour , minute , second) وهي متحولات خاصة من نمط int بالشكل القياسي Standard والعسكري (العام) Universal .
- بحيث يتم تعيين قيم ابتدائية للشكل القياسي (12 : 0 : 0 AM) و للشكل العسكري (0 : 0 : 0) , وعند الإدخال يجب أن يطبع (0) إذا كانت hour و minute و second ذات قيم غير صحيحة .
- يتضمن الصف Time :
 - باني افتراضي يسند القيمة صفر للمتحولات الخاصة

- باني عادي يمرر المعطيات إلى الأغراض عند إنشائها
- باني وسيطه غرض
- باني بوسيط واحد يسند القيمة نفسها إلى المتحولات الخاصة
- باني بوسيطين يقوم بإسناد قيمة للساعات وقيمة أخرى للدقائق والثواني
- تابع طباعة بالشكل القياسي
- تابع طباعة بالشكل العسكري

- أنشئ صفاً آخر يتضمن التابع main اسمه TimeTest لإنشاء أغراضاً لاستدعاء التوابع اللازمة والحصول على النتائج .

مثال (7)

```
using System;
class Time
{
    private int hour, minute, second;
    public Time( ){ hour=0;minute=0;second=0 ; }
    public Time ( int h ,int m ,int s ) {
        hour=(h>=0 && h<24)? h:0;
        minute=(m>=0 && m<60)? m:0;
        second=(s>=0 && s<60)? s:0;
    }
    public void prints()
    {
        Console.WriteLine(" {0,2}:{1,2} :{2,2} :{3,2} \n",
            ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ),
            minute, second, ( hour < 12 ? "AM" : "PM" ) );
    }
    public void printu()
    {
        Console.WriteLine(" {0,2}:{1,2} :{2,2} ", hour , minute,second) ;
    }
}
```

```
public Time(Time ob) {
    hour = ob.hour;
    minute = ob.minute;
    second = ob.second;
}
public Time(int h, int ms) {
    hour = h; minute = second = ms;
}
public Time(int val) {
    hour = minute = second = val;
}
} //End class Time
```

```
public class TimeTest {
static void Main(string[] args)
{
Time    myTime0 = new Time();
Time    myTime1 = new Time( 8,15,40);
Time    myTime2 = new Time( 18,45,35 );
Time myTime3 = new Time(myTime1);
Time myTime4 = new Time(18);
Time myTime5 = new Time(10, 44);
```

```
// display initial Time
Console.WriteLine(" initial Time U ") ;
myTime0.printu();
Console.WriteLine(" initial Time S ") ;
myTime0.prints( );
// display first Time
Console.WriteLine(" first Time U ") ;
myTime1.printu( );
Console.WriteLine(" first Time S ") ;
myTime1.prints( );

// display second Time
Console.WriteLine(" second Time U ") ;
myTime2.printu( );
Console.WriteLine(" second Time S ") ;
myTime2.prints( );

// display third Time
Console.WriteLine("third Time U ") ;
myTime3.printu( );
Console.WriteLine("third Time S ");
myTime3.prints( );
```

```
// display fuorth Time
Console.WriteLine("fuorth Time U  ") ;
    myTime4.printu( );
Console.WriteLine("fuorth Time S  ") ;
    myTime4.prints( );

// display fifth Time
Console.WriteLine(" fifth Time U  ") ;
    myTime5.printu( );
Console.WriteLine(" fifth Time U  ");
    myTime5.prints( );

    }// End main
}// End class TimeTest
```

RESULT

initial Time U

0 : 0 : 0

initial Time S

12:00:00 AM

first Time U

8 : 15 : 40

first Time S

8:15:40 AM

second Time U

18 : 45 : 35

second Time S

6:45:35 PM

third Time U

8 : 15 : 40

third Time U

8 : 15 : 40 .

fuorth Time U

18 : 18 : 18

fuorth Time S

6:18:18 PM

fifth Time U

10 : 44 : 44

fifth Time S

10:44:44 AM

Press any key to continue . . .

دراسة الصف Stack

- هذا المثال يظهر قوة الأغراض , حيث ينشئ هذا البرنامج صفاً يدعى **stack** , وهو يمثل لتخزين الاعداد الصحيحة .
- هذا الصف يحتوي على متحولين خاصين هما **stck** و **tos** . تقوم المصفوفة **stck** بالاحتفاظ بالأعداد المدفوعة إلى المكس ويحتوي **tos** دليل قمة المكس .
- هذا الصف يحتوي على ثلاثة توابع عامة هي :
 - التابع () **init** لتهيئة المكس .
 - التابع () **push** لدفع قيمة إلى المكس .
 - التابع () **pop** لسحب قيمة من المكس .

- يقوم التابع () main بإنشاء مكدسين mystack1 و mystack2 , ثم يدفع عشر أعداد إلى المكس .
- إن المكدسين mystack1 و mystack2 منفصلين عن بعضهما بحيث لا تؤثر القيم المدفوعة إلى mystack1 على القيم المدفوعة إلى mystack2 , أي أن كلاً من الغرضين يملك نسخة خاصة من المتحولين stck و tos .
- يعتبر هذا المبدأ أساساً لفهم الأغراض . على الرغم من أن جميع أغراض هذا الصف تشترك بالتوابع الأعضاء نفسها . إلا أن كل غرض ينشئ ويحتفظ بأعضاء بياناته الخاصة .

```
using System;
// This class defines an integer stack that can hold 10 values.
public class Stack {
    private int []arrayStck = new int[10];
    private int  tos;
    // Initialize top-of-stack
    public Stack( ) { tos = -1 ;}
    // Push an item onto the stack
    public void push(int  item) {
        if(tos==9)
            Console.WriteLine("Stack is full.");
        else
            arrayStck[++tos] = item; } //end push
    // Pop an item from the stack
    public int pop() {
        if(tos < 0) {
            Console.WriteLine("Stack underflow.");
            return 0;
        } //end if
        else
            return arrayStck[tos--];
        } //end pop
    } //end class stack
```

مثال (8)


```

public class TestStack {
    static void Main(string[] args)
    {
        Stack mystack1 = new Stack();
        Stack mystack2 = new Stack();

        // push some numbers onto the stack
        for (int i = 0; i < 10; i++)
            mystack1.push(i);
        for (int i = 10; i < 20; i++)
            mystack2.push(i);
        // pop those numbers off the stack
        Console.WriteLine("Stack in mystack1:");
        for(int i=0; i<10; i++)
            Console.WriteLine( mystack1.pop( ) );
        Console.WriteLine("Stack in mystack2:");
        for(int i=0; i<10; i++)
            Console.WriteLine(mystack2.pop());

        }//end main
    }// end class TestStack

```

RESULT

Stack in mystack2:

19
18
17
16
15
14
13
12
11
10

Stack in mystack1:

9
8
7
6
5
4
3
2
1
0

Press any key to continue . . .

تمارين عامة على الصفوف

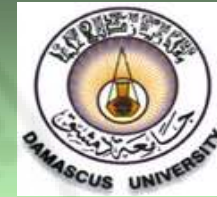
- Q1 - اكتب برنامجاً يحتوي على صف اسمه **Myclass1** يحتوي على ثلاث متحولات n, m, x من النمط `int` , يحتوي الصف على **باني عادي** لتمرير المعطيات إلى الأغراض عند إنشائها , وعلى **مدمر يطبع رسالة** ما عند استدعائه . وعلى **تابع بوسيط واحد اسمه Factorial** لحساب العامل ويعيد قيمة من النوع `int` , وعلى تابع آخر اسمه **SumFact** من النوع `void` يقوم بحساب وطباعة المقدار التالي :
$$sum = n! + \frac{n!}{x!} - m!$$

- أنشئ صفاً آخر اسمه **MyTest1** يحتوي على التابع **main()** لإنشاء أغراض من الصف **Myclass1** واستدعاء توابع الأعضاء اللازمة من الصف **Myclass1** وطباعة النتائج .

- Q2 - اكتب برنامجاً يحتوي على صفّاً اسمه **Myclass2** يحتوي على ثلاث متحولات a, b, x من النمط **double** , يحتوي الصف على **باني عادي** لتمرير المعطيات إلى الأغراض عند إنشائها , وعلى **مدمر يطبع رسالة ما** عند استدعائه . وعلى **تابع اسمه FunctY** من النوع **double** يقوم بحساب المقدار التالي : $y = x^2 + 2x + 10$
- ويحتوي على **تابع آخر اسمه FunctZ** من النوع **void** يقوم بحساب وطباعة المقدار التالي : $Z = (y - a)^2 + (y + b)^2$
- أنشئ صفّاً آخرّاً اسمه **MyTest2** يحتوي على التابع **(main)** لإنشاء أغراض من الصف **Myclass2** واستدعاء توابع الأعضاء اللازمة من الصف **Myclass2** وطباعة النتائج .

البرمجة بلغة C#

C# Programming

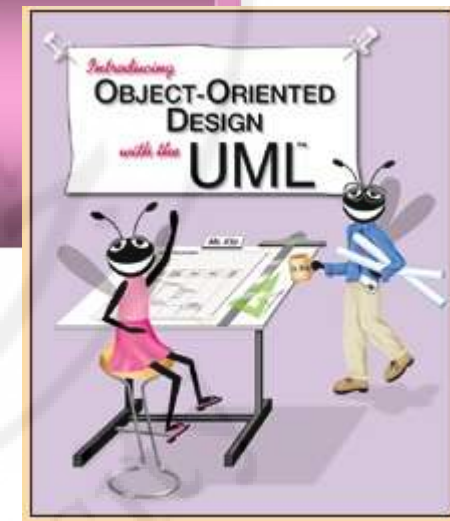


Chapter 1

Dr.Eng.M.Younes



C# Programming



Dr.M.Younes



المحاضرة (2)

C# Programming

التحميل الزائد للتوابع

- تسمح لغة C# بتحميل أسماء التوابع بشكل زائد , الأمر الذي يعني وجود أكثر من تابع يحمل الاسم نفسه وفي الصف نفسه .
- على سبيل المثال بفرض أننا نقوم بكتابة صف يمكنه جمع أعداد صحيحة وأعداد حقيقية وسلاسل محرفية .
- يجب اختيار اسم لكل تابع يقوم بعملية الجمع الخاصة بالنمط الذي سنقوم بتنفيذ العملية عليه .

using System;

مثال (9)

```
public class SumDifferent {
    private int x, y, z;
    private double dx, dy, dz;
    private String str1, str2, str3;
    public int sum(int x, int y) {
        return x + y;
    }
    public int sum(int x, int y, int z) {
        return x + y + z;
    }
    public double sum(double dx, double dy, double dz) {
        return dx + dy + dz;
    }
    public String sum(int x, String str2) {
        return x + str2;
    }
    public String sum(String str1, String str2) {
        return str1 + str2;
    }
} //end class SumDifferent
```



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

        SumDifferent ob1 = new SumDifferent( );

        //Call version1
        Console.WriteLine(" version1 : sum =" + ob1.sum(2,4));
        //Call version2
        Console.WriteLine(" version2 : sum =" + ob1.sum(2, 4, 4));
        //Call version3
        Console.WriteLine(" version3 : sum =" + ob1.sum(2.2, 4.4, 6.6));
        //Call version4
        Console.WriteLine(" version4 : sum =" + ob1.sum("Hello ",
                                                        " M.Younes"));
        //Call version5
        Console.WriteLine(" version5 : sum =" + ob1.sum(2, " tow"));

    } // end main
} // end class SumTest
```

RESULT

```
version1 : sum =6  
version2 : sum =10  
version3 : sum =13.2  
version4 : sum =Hello M.Younes  
version5 : sum =2 tow  
Press any key to continue . . .
```

- إذا كان لتابعين نفس الوسطاء ونفس الأنماط , كيف يتم التمييز بين هذه التوابع ؟
 1. تختلف التوابع المحملة بشكل زائد بعدد الوسطاء الممررة .
 2. أو أنها تختلف بنمط المعطيات للوسطاء الممررة .
 3. أو أنها تختلف بترتيب نمط اللمعطيات للوسطاء الممررة
 4. إذا كانا مختلفين بنمط القيمة المعادة أو مختلفين بمحدد الوصول فلا يعتبر هذا تحميلاً زائداً .

الخصائص (Properties)

(تغليف المعطيات بواسطة الخصائص)

- تقدم الخصائص واجهة عمومية عن معطيات حالة الأغراض ، فهي تقدم أكثر من مجرد غلاف للمتحولات الخاصة ، **فهي تتضمن كتلاً من النص البرمجي تقوم باستحضار المعطيات بواسطة get وإعداد المعطيات بواسطة set** ، مما يعطينا فرصة التحول إلى أنواع معطيات أخرى ، والوصول إلى المعطيات الخاصة للصف أو إنجاز عمليات أخرى .
- فهي تعزز مبدأ التغليف من خلال السماح للصف أو للبنية التحكمية في الوصول إلى معطياتها وأيضاً من خلال إخفاء التمثيل الداخلي للمعطيات .
- **للتصريح عن خاصية نقوم بتحديد نوع المعطيات والاسم ، ثم نتبعها بكتلة من التعليمات تتضمن كتلة get وكتلة set .**
- يتضمن البرنامج التالي نسخة جديدة من الصف Point يتم فيه إخفاء التحقيق implementation في متحولات خاصة private ويستخدم خصائص للكشف عن معطيات الحالة .

مثال (10)

```
using System;
class Point {
    private int x, y;
    public Point(int x, int y) {
        if (x < 0)
            x *= -1;
        if (y < 0)
            y *= -1;
        this.x = x; // set "this" object's x
        this.y = y; // set "this" object's y
    }
    public int fx {
        get { return x; }
        set { x = value; }
    } // fx
    public int fy {
        get { return y; }
        set { y = value; }
    } // fy
} //End class Point
```

يمكن كتابة الخاصية fx و fy التي تحتوي على الكتلة get والكتلة set كما يلي :

```
public int fx{ get ; set ;}
```

```
public int fy { get ; set ; }
```

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

        Point mypoint1 = new Point(-5, -10);
        Point mypoint2 = new Point(10, -20);
        Point mypoint3 = new Point(-3, 6);
        // display first Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint1.fx, mypoint1.fy);

        // display second Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint2.fx, mypoint2.fy);

        // display third Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint3.fx, mypoint3.fy);

    } // End main
} // End class PointTest
```

RESULT

Point is (5 , 10)

Point is (10 , 20)

Point is (3 , 6)

Press any key to continue . . .

- تقوم الكتلة `get` (`get Accessor`) ببساطة بإعادة القيمة الحالية للمتحول العضو , أما الكتلة `set` (`set Accessor`) فتسند قيمة جديدة للمتحول العضو .
- تستخدم كل كتلة `set` الكلمة المحجوزة `value` لإسناد قيمة جديدة للمتحول العضو , حيث تحمل الكلمة المحجوز `value` القيمة التي سيتم اسنادها للخاصية، ونوع هذه القيمة يجب أن يكون مطابقاً للنوع المحدد في تصريح الخاصية .
- يمكن إعادة كتابة الصف `Point` بدون الباني العادي .

مثال (11)

```
using System;
class Point {
    private int x, y;
    public int fx {
        get { return x; }
        set {
            if (value < 0)
                value = value * - 1;
            x = value;
        } //end set
    } //fx
    public int fy {
        get { return y; }
        set {
            if (value < 0)
                value = value * - 1;
            y = value;
        } //end set
    } //fy
} //End class Point
```

```

public class PointTest{

    static void Main(string[] args) {

        Point mypoint1 = new Point() {fx=-5, fy=-10};
        Point mypoint2 = new Point() {fx=10, fy=-20};
        Point mypoint3 = new Point() {fx=-3, fy= 6};
        // display first Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint1.fx, mypoint1.fy);

        // display second Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint2.fx, mypoint2.fy);

        // display third Point
        Console.WriteLine("Point is :({0},{1})",
                           mypoint3.fx, mypoint3.fy);

    }// End main
}// End class PointTest

```

RESULT

Point is (5 , 10)

Point is (10 , 20)

Point is (3 , 6)

Press any key to continue ...

إعطاء قيم ابتدائية للأغراض

- توفر لغة # C إمكانية إعطاء قيم ابتدائية للأغراض التي **تسمح لنا بإنشاء غرض وتهيئته في حالة :**
- **إذا كانت متحولات الصف عامة أي public** (`public int x, y;`) ، وتكون مفيدة عندما لا يوفر الصف باني مناسب لتلبية احتياجاتنا كما هو موضح في المثال (12) .
- **إذا كان الصف يحتوي على الخصائص التي تمكننا من استخدامها لمعالجة المعطيات الخاصة للصف** والتي لا يمكن الوصول إليها كما هو موضح في المثال (11) .

```
using System;  
class Point {
```

```
    public int x, y;
```

```
} //End class Point
```

```
public class PointTest {  
    static void Main(string[] args) {  
        Point mypoint1 = new Point{x=-5, y=-10 };  
        Point mypoint2 = new Point{x=10, y=-20 };  
        Point mypoint3 = new Point{x=-3, y= 6 };  
        // display first Point  
        Console.WriteLine("Point is ({0},{1})", mypoint1.x, mypoint1.y);  
        // display second Point  
        Console.WriteLine("Point is ({0},{1})", mypoint2.x, mypoint2.y);  
        // display third Point  
        Console.WriteLine("Point is ({0},{1})", mypoint2.x, mypoint2.y);  
    } // End main  
} // End class PointTest
```

مثال (12)

RESULT

Point is (- 5 , -10)

Point is (10 , -20)

Point is (-3 , 6)

Press any key to continue . . .

مثال نموذجي لصف يحتوي على باني وسيطه مصفوفة

اكتب برنامجاً بلغة C# يحتوي على صف اسمه **myclass** ويحتوي على وسيطين خاصين **sumOdd** , **sumEven** من النمط **int**, ويحتوي الصف على الأعضاء العامة التالية :

- **باني افتراضي** يسند القيمة صفر للمعطيات الخاصة .
- **باني عادي وسيطه مصفوفة** ويقوم بحساب مجموع الأعداد الفردية والزوجية لمصفوفة من الأعداد الصحيحة مكونة من 10 عناصر, ثم يسند النتائج للمعطيات الخاصة **sumOdd** , **sumEven** على الترتيب .
- **تابع عضو لطباعة** قيم المعطيات الخاصة اسمه **print** .
- **تابع مدمر** يطبع رسالة ما عند استدعائه .
- أنشئ صفاً آخر يتضمن التابع **main** اسمه **myclassTest** ثم **أدخل في التابع main عشر أعداد صحيحة وخرنها في مصفوفة array[10]** , ثم صرح عن غرض **ob** من الصف **myclass** لاستدعاء التوابع اللازمة لحساب وطباعة النتائج .

using System;

مثال (13)

```
public class myclass {
    `private int sumEven , sumOdd ;
    public myclass( ){ sumEven =0; sumOdd=0 ; }
    ~myclass() { Console.WriteLine(" destructor \n"); }

    public myclass(int[] array,int n) { // constructor
        int sumEven1=0 , sumOdd1 =0;
        for (int i = 0; i < array.Length; i++) {
            if( array[i] % 2 == 0 )
                sumEven1 = sumEven1 +array[i];
            else
                sumOdd1= sumOdd1+array[i];
        }//End for
        this.sumEven =sumEven1 ;
        this.sumOdd = sumOdd1 ;
    }//End constructor
    public void print( ){
        Console.WriteLine(" sumEven ={0} sumOdd= {1} ", sumEven, sumOdd);
    }
} //end class myclass
```

```

public class myclassTest {
    public static void Main(string[] args) {

        int []array = new int[10];
        Console.WriteLine(" Enter The Numbers ");
        for ( int i=0 ; i< array.Length ; i++ ) {
            Console.WriteLine(" Enter Array ");
            int x = int.Parse( Console.ReadLine() );
            array[i]=x;
        } // End for3

        myclass ob0 = new myclass();
        myclass ob1 = new myclass(array, 10);

        ob0.print( );
        ob1.print();

        } //End main
    } //End class myclassTest

```

RESULT

Enter The Numbers

1 2 3 4 5 6 7 8 9 0

sumEven = 0 sumOdd= 0

sumEven = 20 sumOdd= 25

destructor

Press any key to continue ...

الصفوف الجزئية

Partial Classes

- **عندما يكون تعريف الصف طويلاً** يصبح من الصعب التعامل معه ككتلة برمجية واحدة ، ويفضل في مثل هذه الحالات **تجزئة الصف إلى أقسام صغيرة** وأكثر قابلية للإدارة بحيث يمكن توزيعها على عدة ملفات .
- **يمكن تحقيق تجزئة الصف باستخدام الكلمة المحجوزة partial** فهي تُخبر المترجم أن هناك أجزاء من تحقيقات implementation الصف موجودة في ملفات مصدريّة أخرى .
- **ولكي نتمكن من ترجمة صف جزئي partial class** يجب أن يتضمن سطر أمر الترجمة قائمة بكل الملفات التي تتضمن التحقيقات الجزئية للصف .
- يتم في المثال التالي لإنشاء نسخة جديدة من الصف Point في المثال (6) ، بحيث تتضمن النسخة الجديدة صفين جزأين .
- الصف الجزئي الأول يتضمن تعرف المتحولات والبناني أما الثاني يتضمن تابع الطباعة فقط .

- يتم تعريف كل صف جزئي في ملف مصدري منفصل بالرغم من أنها جزء من الصف نفسه .
- لترجمة التطبيق علينا أن نكتب سطر أمر يتضمن قائمة بجميع الملفات التي تتضمن التحقيقات الجزئية للصف Point .
- هناك بعض الحالات يكون فيها تجزئة الصف مفيداً :
- ✓ عند العمل على مشاريع كبيرة .
- ✓ عند العمل مع المصادر المنشئة تلقائياً، يمكن إضافة التعليمات البرمجية إلى الصف دون الحاجة إلى إعادة إنشاء ملف المصدر .
- ✓ يستخدم Visual Studio هذا الأسلوب عندما يقوم بإنشاء Windows Forms والتعليمات البرمجية لبرنامج تضمين خدمة الويب وهكذا .
- ✓ يمكننا إنشاء تعليمات برمجية تقوم باستخدام هذه الصفوف دون الحاجة إلى تعديل الملف الذي تم إنشاؤه بواسطة برنامج Visual Studio .
- المثال (13) يوضح الصفوف الجزئية للصف Point .

```
using System;

public partial class Point {

    private int x, y;
    public Point(int x, int y) {

        if (x < 0)
            x *= -1;
        if (y < 0)
            y *= -1;
        this.x = x; // set "this" object's x
        this.y = y; // set "this" object's y
    } // end constructor
} // end partial 1
```



```
using System;

public partial class Point {

    public void printPoint() {

        Console.Write("Point is ");

        Console.WriteLine("({0},{1})", x, y);
    } // end print
} //End class Point
```

```
using System;

public class PointTest {

    public static void Main(string[] args) {

        Point mypoint1 = new Point(-5, -10);
        Point mypoint2 = new Point(10, -20);
        Point mypoint3 = new Point(-3, 6);
        // display first Point
        mypoint1.printPoint();
        // display second Point
        mypoint2.printPoint();
        // display third Point
        mypoint3.printPoint();

    } // End main
} // End class PointTest
```

تمرين (1)

- اكتب برنامجاً بلغة C# يحتوي على ثلاثة صفوف أساسية هي :
 - الصف الأساسي الأول يدعى **Building** بحيث يخزن عدد الطوابق **Floors** في البناء وعدد الغرف **Rooms** ومساحة الغرف **Sqmeter** بالمتر المربع وهي **متغيرات خاصة** . يحتوي الصف على باني افتراضي وباني عادي وتابع طباعة .
 - الصف الأساسي الثاني اسمه **House** , يخزن عدد غرف النوم **Bedrooms** وعدد الحمامات **Bathrooms** وهي **متغيرات خاصة** . يحتوي الصف على باني افتراضي وباني عادي وتابع طباعة .
 - الصف الأساسي الثالث اسمه **Office** يخزن عدد أجهزة الإطفاء **Extinguishers** وعدد الهواتف **Phones** وهي **متغيرات خاصة** . يحتوي على باني افتراضي وباني عادي وتابع طباعة .
 - أنشئ صفاً آخر يتضمن التابع **main** اسمه **BuildingTest** لإنشاء أغراضاً للصفوف الثلاثة لاستدعاء التوابع اللازمة والحصول على النتائج .

تمرین (1)

```
using System;

public class Building {

    private int Floors, Rooms;
    private double Sqmeter;
    public Building() { Floors = 0; Rooms = 0; Sqmeter = 0; }
    public Building(int f, int r, double m)
    {
        Floors = f;
        Rooms = r;
        Sqmeter = m;
    }
    public void Printbuilding() {

        Console.WriteLine(" Rooms is " + Rooms);
        Console.WriteLine(" Floors is " + Floors );
        Console.WriteLine(" Sqmeter is " + Sqmeter);
    }

} // end class Building
```

```
public class House {
private int Bedrooms, Bathrooms;
    public House()
        { Bedrooms = 0; Bathrooms = 0; }
    public House(int br, int bth) {
        Bedrooms = br; Bathrooms = bth;
    }
    public void PrintHouse() {
Console.WriteLine(" bedrooms is " + Bedrooms);
        Console.WriteLine(" bathrooms " + Bathrooms);
    }
} // end class House

public class Office {
private int Phones, Extinguishers;
    public Office(int p, int ext) {
        Phones = p; Extinguishers = ext;
    }
    public void PrintOffice() {
Console.WriteLine(" Telephones is " + Phones);
        Console.WriteLine(" Fire Extinguishers is " + Extinguishers);
    }
} // end class Office
```

```
public class BuildingTest {  
    public static void Main(string[] args) {  
Building    myBuilding1 = new Building(10, 200, 1500);  
        myBuilding1.Printbuilding( );  
        House    myHouse1 = new House(100, 150);  
        myHouse1.PrintHouse( );  
        Office    myOffice1 = new Office(150, 250);  
        myOffice1.PrintOffice();  
  
    } // end main  
} // end class BuildingTest
```

Result

Building Class :

Rooms is 200

Floors is 10

Sqmeter is 1500.0

House Class :

bedrooms is 100

bathrooms 150

Office Class :

Telephones is 150

fire extinguishers is 250

Press any key to continue . . .

تمرين (2)

- اكتب برنامجاً بلغة C# للصف Date يحتوي على ثلاثة متحولات خاصة ويحتوي على باني افتراضي وباني عادي وتابع طباعة (print) ويقوم بطباعة التاريخ بالشكل " 15 may 2006 " .
- بحيث يتم تعيين قيمة ابتدائية تعتبر (01 January 2006) , وعند الإدخال يجب أن يطبع (0) إذا كانت day أصغر أو أكبر من (31) , وأن يطبع (0) إذا كانت year أصغر من 2006 , حيث المتحول day و المتحول year من نوع int , أما المتحول month من نوع String .
- أنشئ صفاً آخر يتضمن التابع main اسمه DateTest لإنشاء أغراضاً لاستدعاء التوابع اللازمة والحصول على النتائج .

تمرين (3)

- اكتب برنامجاً بلغة C# للصف Student حيث : الصف student يقوم بإدخال الاسم الثلاثي first, mid, last كثلاثة متحولات من النوع String وطباعتها .
- وإدخال ثلاث مواد دراسية course1, course2 , course3 من النوع int وحساب المعدل الوسطي للطالب .
- أنشئ صفّاً آخر يتضمن التابع main اسمه StudentTest لإنشاء أغراضاً لاستدعاء التوابع اللازمة والحصول على النتائج .

تمرين (4)

- اكتب برنامجاً بلغة C# للصف StudentMark حيث : الصف studentMark يقوم بإدخال ثلاثة اسماء طلاب name1, name2, name3 كثلاثة متحولات من النمط String وإدخال ثلاثة درجات m1,m2,m3 كثلاثة متحولات من النمط int .
- يحتوي الصف على باني عادي وعلى تابع طباعة (max() لحساب وطباعة أعلى درجة وطباعة اسم الطالب الذي حصل عليها , وعلى تابع طباعة آخر (min() لحساب وطباعة أدنى درجة وطباعة اسم الطالب الذي حصل عليها .
- أنشئ صفّاً اختبارياً يتضمن التابع main اسمه Test لإنشاء أغراضاً واستدعاء التوابع اللازمة والحصول على النتائج .

حل التمرين (4)

```
using System;
public class StudentMark {
    private int m1, m2, m3;
    private String name1, name2, name3;
    public StudentMark(String n1, String n2, String n3,
                        int x, int y, int z)
    {
        m1 = x;      m2 = y;      m3 = z;
        name1 = n1;   name2 = n2;   name3 = n3;
    }
    public void max() {
        int max1 = m1;
        String max2 = name1;
        if (m2 > max1) {
            max1 = m2;
            max2 = name2;
        }
        else if (m3 > max1) {
            max1 = m3;
            max2 = name3;
        }
        Console.WriteLine ( " max is : " +max1 + max2);
    }
} // end max
```

```
public void min( ) {  
    int min1 = m1 ;  
    String min2 = name1 ;  
    if ( m2 < min1 ){  
        min1 = m2 ;  
        min2 = name2 ;  
    }  
    else if ( m3 < min1 ){  
        min1 = m3 ;  
        min2 = name3 ;  
    }  
    Console.WriteLine( " max is : " + min1 + min2);  
} // end min  
} // end class StudentMark
```

```
public class Test {  
    static void Main(string[] args) {  
        StudentMark Ob = new StudentMark(" Ali ", "Sameer ",  
                                           " Waseem ", 80, 65, 75);  
        Ob.max();  
        Ob.min();  
    } // end main  
} // end class Test
```

RESULT

max is : 80 Ali

min is : 65 Sameer

Press any key to continue . . .

the end

The End

The End

The End

The End

The

The End

End

The End

البرمجة بلغة C#

C# Programming

Dr.Eng.M.Younes

1

Java Programming

Dr.Mamou Younes

الوراثة

Inheritance

Chapter 7

Dr.Eng.M.Younes

2

C# Programming

Dr.Mamou Younes

محتويات الفصل السابع

- مقدمة
- التحكم بالوصول إلى الصف المشتق
- دراسة الصف **Student**
- دراسة الصف **Box**
- الوراثة المتعددة
- دراسة الصف **Box** (وراثة متعددة)
- البواني والوراثة

3

C# Programming

Dr.Mamou Younes

الوراثة Inheritance



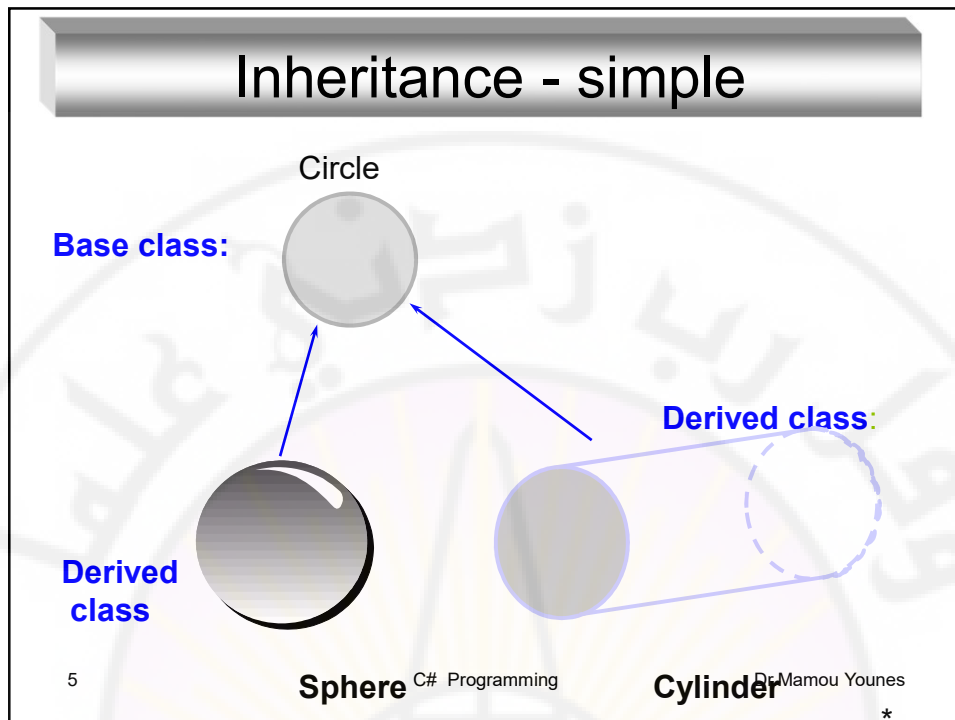
تعريف

- الوراثة هي الآلية التي بواسطتها يمكن لصف ما أن يرث خصائص صف آخر .
- تسمح الوراثة ببناء بنية هرمية من الصفوف ابتداءً من الصف الأكثر عمومية وانتهاءً بالصف الأكثر خصوصية .
- يجب تعريف مصطلحين مستخدمين بكثرة في الوراثة:
 - يدعى الصف الذي يورث بالصف الأساسي **Base Class**.
 - ويدعى الصف الذي يرث بالصف المشتق **Derived class**.

4

C# Programming

Dr.Mamou Younes



❑ يعرف الصف الأساسي **Base Class** جميع الخواص المشتركة لأي من الصفوف المشتقة **Derived Class** , أي يقدم وصفاً عاماً لمجموعة من الخصائص .

❑ يرث الصف المشتق **Derived Class** الخصائص العامة من الصف الأساسي ويضيف إليها خصائص خاصة به .

❑ لغة **C#** توفر وسيلة لتقليل حجم البرنامج حيث يستفاد من بعض التوابع في بعض الصفوف , وهذا يعني اشتقاق صفوف عديدة من صفوف محددة ثم إضافة المواصفات الإضافية الخاصة بالصف المشتق .

❑ وبهذه الطريقة لا نحتاج إلى إعادة كتابة التوابع مرة أخرى .

التحكم بالوصول إلى الصف الأساسي Base Class Access Control

- عند وراثة صف لصف آخر نستخدم الشكل العام التالي :

```
public class Derivedclass_Name : Baseclass_Name
{
    .....class Body .....
} // End subclass
```

7

C# Programming

Dr.Mamou Younes

الصفوف الأساسية والصفوف المشتقة

- يمكن في لغة C# أن يرث صف مشتق Derived class الصف الأساسي Base class بواسطة المحدد (:) .
- يرث الصف المشتق جميع المعطيات الأعضاء والتوابع الأعضاء الموجودة في الصف الأساسي والتي يمكن الوصول إليها من خلال الصف المشتق .
- لا يمكن للصف المشتق الوصول إلى الأعضاء الخاصة private في الصف الأساسي فهي غير مرئية له .
- يستطيع الصف المشتق الوصول إلى الأعضاء العامة public والمحمية protected.

8

C# Programming

Dr.Mamou Younes

• يمكن تصنيف الأعضاء التي يمكن وراثتها كما يلي :

1. تترث الصفوف المشتقة من الصف الأساسي الأعضاء التي لها المحدد `public` و `protected`.
 2. لا تترث الصفوف المشتقة من الصف الأساسي البواني لأنها ليست أعضاء.
 3. لا تترث الصفوف المشتقة من الصف الأساسي المدمرات لأنها ليست أعضاء.
- يوضح المثال (1) الأعضاء العامة والمحمية والخاصة .

9

C# Programming

Dr.Mamou Younes

مثال (1)

```
using System;

// A simple example of inheritance.
// Create a Base class.
public class ClassA {
//public int i ,j; // public by default
//protected int i, j ; // public by default
private int i, j ; // private to ClassA
public void setij(int x, int y) {
    i = x;
    j = y;
}
public void showij( ) {
    Console.WriteLine("i and j: " + i + " " + j);
}
} //End ClassA
```

10

C# Programming

Dr.Mamou Younes

```

public class ClassB : ClassA {
    private int k;
    public void set1( int x , int y ,int z )
    {
        setij( x,y);
        k =z;
    }
    public void showk( ) {
        Console.WriteLine("k: " + k);
    }
    //public void sum( ) {
    //Console.WriteLine("i+j+k: " + (i+j+k));
    //} // ERROR, j and i is not accessible if are private
} //End ClassB

```

11

C# Programming

Dr.Mamou Younes

```

public class Tester
{
    static void Main(string[] args)
    {

        ClassB Ob2 = new ClassB( );

        Ob2.set1(10, 12,8);
        Console.WriteLine("Contents of Ob2: ");
        Ob2.showij();
        Ob2.showk();

    } //End main
} //End SimpleInheritance

```

RESULT

Contents of Ob2:
i and j: 10 12
k: 8

Press any key to continue . . .

12

C# Programming

Dr.Mamou Younes

- عندما تكون المتحولات **ز.إ** في الصف الأساسي ClassA هي متحولات من نوع **public** لا يحدث أي خطأ في البرنامج عندما يحاول الصف ClassB الوصول إلى الأعضاء العامة .
- عندما تكون المتحولات **ز.إ** في الصف الأساسي ClassA هي متحولات من نوع **private** يحدث خطأ عندما يحاول الصف المشتق ClassB الوصول إلى الأعضاء الخاصة **ز.إ** في الصف الأساسي . وهذا خطأ لأن العضو الخاص في الصف الأساسي يبقى خاص به مهما كان الشكل الذي تمت به عملية الوراثة .
- عندما تكون المتحولات **ز.إ** في الصف الأساسي ClassA هي متحولات من نوع **protected** لا يحدث أي خطأ في البرنامج عندما يحاول الصف ClassB الوصول إلى الأعضاء العامة ، لأن الأعضاء المحمية للصف الأساسي مرئية من قبل أعضاء الصف المشتق .
- هذا المثال يوضح بأننا لا نستطيع الوصول إلى المعطيات الخاصة وسوف يعطينا خطأ **ERROR** .

13

C# Programming

Dr.Mamou Younes

الصف الأساسي Object Object Base Class

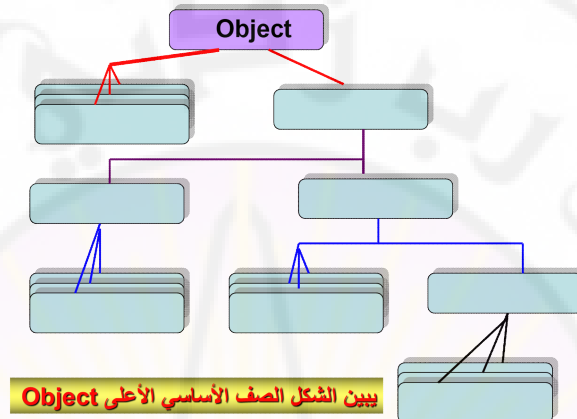
- يوجد في C# صف يعتبر هو الصف الأساسي الأعلى (أعلى صف في الشجرة الهرمية) نسميه الصف Object , وكل صف آخر يُعتبر صف مشتق له (بشكل مباشر أو غير مباشر) .
- إن أي متحول من نمط الصف Object يمكن أن يحوي مرجعاً على أي غرض آخر , مثل غرض من صف أو مصفوفة .
- إذا الصف Object هو الصف الأعلى في بنية الشجرة الهرمية , هذا الصف تُشتق منه جميع الصفوف الأخرى .
- الصف Object يوفر مجموعة من التوابيع Methods اللازمة لجميع الأغراض , فمثلاً جميع الصفوف ترث التابع toString التابعة للصف Object والتي تعيد سلسلة String تمثل شكل مطبوع للغرض .
- التابع Equals() يحدد فيما إذا كان غرضان متساويين أم لا .
- التابع GetHashCode() يسمح للأغراض بتحديد رقم التجزئة من أجل الترتيب داخل المجموعات الجزئية .

14

C# Programming

Dr.Mamou Younes

- التابع GetType() يتيح الوصول إلى الغرض الذي يمثل الصف .
- التابع Finalize() يحرر الموارد غير الذكية ، ويحقق بواسطة المدمر .
- ويمكننا القول أننا كلما اتجهنا من أعلى إلى أسفل شجرة الصفوف السابقة فإننا سنحصل على صفوف أكثر تخصيصاً في سلوكها . كما يوضح الشكل التالي :



15

C# Programming

Dr.Mamou Younes

استخدام الأعضاء المحمية

Using Protected Members

- رأينا في المثال السابق أن الصف المشتق لا يستطيع الوصول إلى الأعضاء الخاصة للصف الأساسي . هذا يعني أنه إذا احتاج الصف المشتق للوصول إلى بعض الأعضاء الخاصة للصف الأساسي فإن هذه الأعضاء يجب أن تكون عامة .
- إلا أننا في بعض الأحيان قد نرغب لأحد أعضاء الصف الأساسي أن يبقى خاصاً به مع السماح للصف المشتق بالوصول إليه ، من أجل ذلك نستخدم محدد الوصول **protected** .
- يكافئ محدد الوصول **protected** المحدد **private** ما عدا أن الأعضاء المحمية للصف الأساسي مرئية من قبل أعضاء الصف المشتق . في حين أنها غير مرئية لأي صف من خارج الصف الأساسي أو الصفوف المشتقة منه .

16

C# Programming

Dr.Mamou Younes

دراسة الصف Student

- اكتب برنامجاً بلغة C# يحتوي على صف أساسي Student يحتوي على المتحولات الخاصة التالية :
 - ❖ First الاسم الأول , mid الاسم الوسط , last الاسم الأخير
 - ❖ id الرقم الجامعي وهي من نوع string .
 - ❖ ويحتوي الصف على باني عادي وتابع طباعة .
- لدينا الصف المشتق Course الذي يرث الصف الأساسي Student ويحتوي على المتحولات الخاصة التالية :
 - ❖ cid القسم , description اسم المقرر , grade النتيجة في المقرر وهي من نوع string .
 - ❖ ويحتوي الصف على باني عادي وتابع طباعة .

17

C# Programming

Dr.Mamou Younes

مثال (2)

```
using System;
public class Student {
private String first ,mid, last,id;
// constructor
public Student( String f, String m, String l, String i ) {
    first = f;
    mid = m ;
    last = l;
    id = i ;
} // end constructor
public void printstudent( )
{
    Console.WriteLine("Name      : {0} {1} {2} {3} " ,
        first ,mid ,last,id );
} // end function print
} // end class student
```

18

C# Programming

Dr.Mamou Younes

```
// class Course
public class Course : Student {

    private String cid , description, grade ;
    public Course ( String f ,String m ,String l ,String i ,
        String ci , String cd ,String g): base( f, m, l, i ) {
        cid = ci ;
        description = cd;
        grade = g;
    }//end constructor

    public void showgrade( ) // print( )
    {
        printstudent( );           // super. print( )
        Console.WriteLine("Address : {0} {1} {2} " ,
            cid ,description ,grade );
    }
} // End class course
19
```

C# Programming

Dr.Mamou Younes

```
public class Tester
{
    static void Main(string[] args)
    {
        Course ob = new Course ( "Waseem","saalem","Ali",
            "1234 khlid street " , " Damascus","Syria ", "99887" ) ;

        ob.showgrade();

    } // end main
} // end Tester
```

RESULT

Name : Waseem salem Ali 1234 khlid street
 Address : Damascus Syria 99887
 Press any key to continue . . .

20

C# Programming

Dr.Mamou Younes

تجاوز التابع Method overriding

الاستخدام الأول للكلمة المحجوزة base

- يتم استخدام الكلمة المحجوزة **base** لاستدعاء باني الصف الأساسي من الصف المشتق وتميرير البارامترات له .
- إذا لم يكن هناك استدعاء صريح لباني الصف الأساسي فإنه يتم القيام باستدعاء آلي لباني الصف الأساسي الذي ليس له وسطاء .

الاستخدام الثاني للكلمة المحجوزة base

- يتم استخدام الكلمة المحجوزة **base** من أجل الوصول إلى التوابيع الموجودة في الصف الأساسي والمعاد تعريفها من قبل الصفوف المشتقة .
- كما ذكرنا سابقاً , المتحولات العضوية المعرفة في الصف المشتق تخفي المتحولات العضوية في الصف الأساسي والتي لها نفس الاسم .
- يستطيع الصف المشتق الوصول إلى المتحولات المخفية أو التوابيع المخفية في الصف الأساسي باستخدام الكلمة المحجوزة **base** .
- نُعيد كتابة المثال (2) بحيث يحتوي الصف **Student** على تابع طباعة اسمه **print** وكذلك الصف **Course** كما هو موضح في البرنامج التالي .

21

C# Programming

Dr.Mamou Younes

مثال (3)

```
using System;
public class Student {
private String first ,mid, last,id;
// constructor
public Student( String f, String m, String l, String i ) {
    first = f;
    mid = m ;
    last = l;
    id = i ;
} // end constructor
public void print( )
{
    Console.WriteLine("Name      : {0} {1} {2} {3} " ,
        first ,mid ,last,id );
} // end function print
} // end class student
```

22

C# Programming

Dr.Mamou Younes


```
// class Course
public class Course : Student {

    private String cid , description, grade ;
    public Course ( String f ,String m ,String l ,String i ,
        String ci , String cd ,String g): base( f, m, l, i ) {
        cid = ci ;
        description = cd;
        grade = g;
    }//end constructor

    public void print( )
    {
        base. print( );
        Console.WriteLine("Info : {0} {1} {2} " ,
            cid ,description ,grade );
    }
} // End class course
23
```

C# Programming

Dr.Mamou Younes

```
public class Tester
{
    static void Main(string[] args)
    {
        Course ob = new Course ( "Waseem","salem","Ali",
            "1234 ", " Computer Engineeering ", "Programming2 ", "85" ) ;

        ob.print();

    } // end main
} // end Tester
```

RESULT

Name : Waseem salem Ali 1234

Info : Computer Engineeering, Programming2, 85

Press any key to continue ...

24

C# Programming

Dr.Mamou Younes

البواني والمدمرات

Constructor and Destructor

- يمكن أن يكون لكل من الصف الأساسي والصف المشتق باني واحد أو أكثر ويكون لكل منهما مدمراً واحداً .
- عندما يكون لكلا الصنفين الأساسي والمشتق توابع بانية ومدمرة , فإن التوابع البانية تنفذ بنفس ترتيب الاشتقاق في حين أن التوابع المدمرة بالترتيب المعاكس .
- هذا يعني أن التابع الباني للصف الأساسي ينفذ قبل التابع الباني للصف المشتق ، أما بالنسبة للتابع المدمر فإن التابع المدمر للصف المشتق ينفذ أولاً ثم التابع المدمر للصف الأساسي .
- إذا تم تنفيذ التابع المدمر للصف الأساسي أولاً سيؤدي إلى تدمير الأغراض قبل أن يستخدمها الصف المشتق ، لذلك يجب أن ينفذ التابع المدمر للصف المشتق أولاً ثم للصف الأساسي .

25

C# Programming

Dr.Mamou Younes

دراسة الصف Box

(وراثه)

- اكتب برنامجاً بلغة C# يحتوي البرنامج على الصفوف التالية :
- 1. **يحتوي على صف أساسي Base class اسمه Box** يقوم بحساب حجم الصندوق الذي هو عبارة عن متوازي مستطيلات أبعاده height , width , length وهي متحولات خاصة private من نمط double , ويحتوي الصف على باني افتراضي لإسناد القيم الابتدائية للمتحولات الخاصة و **باني عادي** و **باني بسيط واحد** لإعطاء قيمة واحدة للمتحولات ليتحول الشكل إلى مكعب , وعلى **باني بسيطه غرض**, وعلى تابع volume يقوم بحساب حجم متوازي المستطيلات (حجم الصندوق) .

26

C# Programming

Dr.Mamou Younes

2. **يحتوي البرنامج على صف مشتق `derived class` اسمه `BoxWeight`** يرث الصف الأساسي `base class` , ويحتوي على متحول واحد خاص هو وزن الصندوق `Weight` وهو من نمط `double` , ويحتوي على باني افتراضي و باني عادي و باني وسيطه غرض , وعلى باني يُستخدم عندما يصبح الصندوق مكعباً .
3. **يحتوي البرنامج على صف اختبار اسمه `Tester`** يحتوي على التابع `main` لإنشاء الأغراض واستدعاء التوابع اللازمة والحصول على النتائج .

27

C# Programming

Dr.Mamou Younes

```
using System;
// A complete implementation of BoxWeight.
public class Box {
    private double width;
    private double height;
    private double depth;
    // construct clone of an object
    public Box(Box ob)
    { // pass object to constructor
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }
    // constructor used when all dimensions specified
    public Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }
}
```

مثال (4)

C# Programming

Dr.Mamou Younes

```

// constructor used when no dimensions specified
public Box( ) {
    width = 1; // use -1 to indicate an un initialized box
    height = 1;
    depth = 1;
}
// constructor used when cube is created
public Box(double len) {
    width = height = depth = len;
}
// compute and return volume
public void volume( ) {
    Console.WriteLine( " Volume is :"+ width * height * depth );
}
} //End Class Box

```

29

C# Programming

Dr.Mamou Younes

```

// BoxWeight now fully implements all constructors.
public class BoxWeight : Box {
    private double weight; // weight of box
    // construct clone of an object
    public BoxWeight(BoxWeight ob) : base(ob) {
        // pass object to constructor
        weight = ob.weight;
    }
    // constructor when all parameters are specified
    public BoxWeight( double w, double h, double d, double m) :
        base(w, h, d) { // call superclass constructor
        weight = m;
    }
    // default constructor
    public BoxWeight( ) : base( ) {
        weight = 1;
    }
}

```

30

C# Programming

Dr.Mamou Younes

```
// constructor used when cube is created
public BoxWeight(double len, double m) : base(len) {
    weight = m;
}
public void VolWeight(){
    volume() ;
    Console.WriteLine( " Weight is :{0}",weight );
}
} //End BoxWeight
// Class Tester r
public class Tester {
    static void Main(string[] args) {
        BoxWeight mybox1 = new BoxWeight(10,20,15,34.3);
        BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
        BoxWeight mybox3 = new BoxWeight( ); // default
        BoxWeight mycube = new BoxWeight(3, 2); //cube
        BoxWeight myclone = new BoxWeight(mybox1);
    }
}
```

31

C# Programming

Dr.Mamou Younes

```
Console.WriteLine("\n mybox1 ");
    mybox1.VolWeight();
Console.WriteLine("\n mybox2 ");
    mybox2.VolWeight();
Console.WriteLine("\n mybox3 ");
    mybox3.VolWeight();
Console.WriteLine("\n mycube ");
    mycube.VolWeight();
Console.WriteLine("\n myclone ");
    myclone.VolWeight();
} //End main
} //End Tester
```

RESUL**mybox1**

Volume is :3000 Weight is :34.3

mybox2

Volume is :24 Weight is :0.076

mybox3

Volume is :1 Weight is :1

mycube

Volume is :27 Weight is :2

myclone

Volume is :3000 Weight is :34.3

Press any key to continue . . .

32

C# Programming

Dr.Mamou Younes

الوراثة المتعددة

B1



D1



D2

• من الممكن أن يكون الصف المشتق صفاً أساسياً لصف مشتق آخر مشكلاً بذلك مراتب متعددة المستويات من الصفوف . ويصبح الصف الأساسي الأصلي صفاً غير مباشر للصف المشتق التالي كما هو مبين في الشكل .

• يبين البرنامج التالي ثلاثة صفوف كما يلي :

- الصف الأساس A
- الصف المشتق الأول B
- الصف المشتق الثاني C

33

C# Programming

Dr.Mamou Younes

ملاحظات

- - عند استخدام الصف المشتق كصف أساسي لصف مشتق آخر ، فإن البواني للصفوف الثلاثة D2 , D1 , B1 تُستدعي حسب ترتيب الاشتقاق ، أما المدمرات فتستدعي بشكل معاكس .
- - إن البواني لا تورث في الصفوف المشتقة ولذلك فإن كل صف مشتق يجب أن يصرح عن البواني الخاصة به .
- - تورث الأعضاء الخاصة بواسطة الصف المشتق لكن لا يمكن الوصول إليها من قبل الصف المشتق ، وللوصول إليها علينا التصريح عنها كأعضاء محمية protected .

34

C# Programming

Dr.Mamou Younes

دراسة الصف Building (وراثه متعددة)

- اكتب برنامجاً بلغة C# يحتوي على :
- الصف على الأساسي **Building** بحيث يخزن عدد الطوابق **Floors** في البناء وعدد الغرف **Rooms** ومساحة الغرف **Sqmeter** بالمتري المربع وهي **متغيرات خاصة**. يحتوي الصف على باني افتراضي وباني عادي وتابع طباعة , وعلى باني وسيطه غرض **ob**.
- الصف المشتق الأول اسمه **House** , يرث الصف **Building** , ويخزن عدد غرف النوم **Bedrooms** وعدد الحمامات **Bathrooms** وهي **متغيرات خاصة**. يحتوي الصف على باني افتراضي وباني عادي وتابع طباعة , وعلى باني وسيطه غرض **ob**.
- الصف المشتق الثاني اسمه **Office** , يرث الصف **House** , ويخزن عدد أجهزة الإطفاء **Extinguishers** وعدد الهواتف **Phones** وهي **متغيرات خاصة**. يحتوي على باني افتراضي وباني عادي وتابع طباعة , وعلى باني وسيطه غرض **ob**.
- أنشئ صفاً آخر يتضمن التابع **main** اسمه **Tester** لإنشاء أغراضاً في الصف المشتق **Office** , لاستدعاء التوابيع اللازمة والحصول على النتائج .

35

Dr.Mamou Younes

مثال (5)

```
using System;
public class Building {
    private int Floors, Rooms;
    private double Sqmeter;
    public Building() { Floors = 0; Rooms = 0; Sqmeter = 0; }
    public Building(int f, int r, double m) {
        Floors = f;
        Rooms = r;
        Sqmeter = m; }
    public Building(Building ob) { // pass object to constructor
        Floors = ob.Floors;
        Rooms = ob.Rooms;
        Sqmeter = ob.Sqmeter;
    }
    public void Printbuilding( ) {
        Console.WriteLine(" Rooms is " + Rooms);
        Console.WriteLine(" Floors is " + Floors );
        Console.WriteLine(" Sqmeter is " + Sqmeter); }
} //End Class Building
```

C# Programming

Dr.Mamou Younes

```

public class House : Building {
private int Bedrooms, Bathrooms;
    public House(): base(){// call superclass constructor
        Bedrooms = 0; Bathrooms = 0; }
public House(int f, int r, double m, int br, int bth)
    : base(f, r, m) { // call baseclass constructor
        Bedrooms = br; Bathrooms = bth;
    }
public House(House ob): base(ob) { // pass object to constructor
    // call baseclass constructor
    Bedrooms = ob.Bedrooms;
    Bathrooms = ob.Bathrooms;
}
public void PrintHouse( ) {
    base.Printbuilding( );
    Console.WriteLine(" bedrooms is " + Bedrooms);
    Console.WriteLine(" bathrooms " + Bathrooms);
}
} //End class House

```

C# Programming

Dr.Mamou Younes

```

public class Office : House {
private int Phones , Extinguishers;
public Office( ):base( ) { Phones = 0; Extinguishers = 0; }
public Office( int f ,int r ,double m , int br, int bth ,int p,
int ext) :base(f, r, m ,br ,bth ){ // call superclass constructor
    Phones = p; Extinguishers = ext;
}
public Office( Office ob):base(ob ){ // pass object to constructor
    // call superclass constructor
    Phones = ob.Phones;
    Extinguishers = ob.Extinguishers;
}
public void PrintOffice() {
    base.PrintHouse( );
    Console.WriteLine(" Telephones is " + Phones);
    Console.WriteLine(" Fire Extinguishers is " + Extinguishers);
}
} //End Class Office

```

38

C# Programming

Dr.Mamou Younes


```

public class Tester {
    static void Main(string[] args) {
        Office myOffice1 = new Office(10, 200, 15000 ,100, 150 ,150, 250 );
        myOffice1.PrintOffice( );
        Console.WriteLine("\n myclone is\n ");
        Office myOffice2 = new Office(myOffice1 );
        myOffice2.PrintOffice( );
    } //End main
} // End class Tester

```

39

C# Programming

Dr.Mamou Younes

RESULT

myOffice1

Rooms is 200
Floors is 10
Sqmeter is 1500.

bedrooms is 100
bathrooms 150

Telephones is 150
fire extinguishers is 250

myclone is

Rooms is 200
Floors is 10
Sqmeter is 1500.0

bedrooms is 100
bathrooms 150

Telephones is 150
fire extinguishers is 250

Press any key to continue . . .

40

C# Programming

Dr.Mamou Younes

دراسة الصف Box (وراثة متعددة)

- لتعيد كتابة البرنامج للصف Box في المثال (4) الذي شُرح سابقاً , ويتكون من :
 - الصف الأساسي Box
 - الصف المشتق BoxWeight يرث الصف الأساسي Box
 - نضيف صفّاً اسمه Shipment وهو صف مشتق من الصف BoxWeight ويعبر عن ثمن شحن الصندوق , ويحتوي على باني افتراضي وباني عادي وباني وسيطه غرض وعلى باني يستخدم عند إنشاء مكعب , ويحتوي الصف على متحول واحد هو cost هو ثمن الشحن .
 - يحتوي على صف الاختبار Tester .

41

C# Programming

Dr.Mamou Younes

مثال (6)

```
using System;
// A complete implementation of BoxWeight.
public class Box {
    private double width;
    private double height;
    private double depth;

    // construct clone of an object
    public Box(Box ob)
    { // pass object to constructor
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }

    // constructor used when all dimensions specified
    public Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }
}
```

C# Programming

Dr.Mamou Younes

```

// constructor used when no dimensions specified
public Box( ) {
    width = 1; // use -1 to indicate an un initialized box
    height = 1;
    depth = 1;
}
// constructor used when cube is created
public Box(double len) {
    width = height = depth = len;
}
// compute and return volume
public void volume( ) {
    Console.WriteLine( " Volume is :"+ width * height * depth );
}
} //End Class Box

```

43

C# Programming

Dr.Mamou Younes

```

// BoxWeight now fully implements all constructors.
public class BoxWeight : Box {
    private double weight; // weight of box
    // construct clone of an object
    public BoxWeight(BoxWeight ob) : base(ob) {
        // pass object to constructor
        weight = ob.weight;
    }
    // constructor when all parameters are specified
    public BoxWeight( double w, double h, double d, double m) :
        base(w, h, d) { // call superclass constructor
        weight = m;
    }
    // default constructor
    public BoxWeight( ) : base( ) {
        weight = 1;
    }
}

```

44

C# Programming

Dr.Mamou Younes

```

// constructor used when cube is created
public BoxWeight(double len, double m) : base(len) {
    weight = m;
}
public void VolWeight(){
    volume() ;
    Console.WriteLine( " Weight is :{0}",weight );
}
} //End BoxWeight

public class Shipment : BoxWeight {
    private double cost;
    // construct clone of an object
    public Shipment(Shipment ob) : base(ob) { // pass object to
        constructor
        cost = ob.cost;
    }
}

```

45

C# Programming

Dr.Mamou Younes

```

// constructor when all parameters are specified
public Shipment(double w, double h, double d, double m, double c)
    :base(w, h, d, m) { // call superclass constructor
    cost = c;
}
// default constructor
public Shipment( ) :base( ) {
    cost = 1;
}
// constructor used when cube is created
public Shipment(double len, double m, double c) :base(len, m){
    cost = c;
}
public void printAll() {
    base.VolWeight();
    Console.WriteLine( " Cost is :{0}$ ",cost );
}
} // End Class Shipment

```

46

C# Programming

Dr.Mamou Younes

```

public class Tester {
    static void Main(string[] args) {
        Shipment mybox1 = new Shipment(10, 20, 15, 34.3,20);
        Shipment mybox2 = new Shipment(2, 3, 4, 0.076,2);
        Shipment mybox3 = new Shipment(); // default
        Shipment mycube = new Shipment(3, 2,44); //cub
        Shipment myclone = new Shipment(mybox1);
        Console.WriteLine("\n mybox1  " );
        mybox1.printAll();
        Console.WriteLine("\n mybox2  " );
        mybox2.printAll();
        Console.WriteLine("\n mybox3  " );
        mybox3.printAll();
        Console.WriteLine("\n mycube  " );
        mycube.printAll();
        Console.WriteLine("\n myclone  " );
        myclone.printAll();
    } //End main
} //End Tester
4/

```

C# Programming

Dr.Mamou Younes

RESUL**mybox1**

Volume is :3000 Weight is :34.3 Cost is :20\$

mybox2

Volume is :24 Weight is :0.076 Cost is :2\$

mybox3

Volume is :1 Weight is :1 Cost is :1\$

mycube

Volume is :27 Weight is :2 Cost is :44\$

myclone

Volume is :3000 Weight is :34.3 Cost is :20\$

Press any key to continue . . .

48

C# Programming

Dr.Mamou Younes

البواني والوراثة

- نذكر بعض الملاحظات الهامة تتعلق بالبواني والوراثة , على الرغم من أن بعضها مذكورة سابقاً , وهي :
 1. **البواني لا تورث** و لذلك علينا كتابة جميع البواني المطلوبة في جميع الصفوف .
 2. **الصف الذي لا يملك بان** , يتم توليد **بان واحد** فقط له يدعى الباني الافتراضي وهو **باني بدون وسطاء** .
 3. **يمكن استدعاء بان الصف الأساسي من الصف المشتق** , وذلك باستخدام الكلمة المحجوزة **base** , وفي هذه الحالة يجب أن تظهر هذه الكلمة كأول تعليمة في الباني بعد **public** .

49

C# Programming

Dr.Mamou Younes

- 3. إذا لم نستخدم **base** , فإن المترجم سيقوم باستدعاء الباني الافتراضي **base()** , فإذا قمنا بوراثة صف جميع بوانيها **وسطاء** , فإن المترجم لن يقوم باستدعاء **base** الافتراضية بشكل تلقائي , لذلك علينا أن نقوم باستدعاء بان من الصف الأساسي صراحة .

50

C# Programming

Dr.Mamou Younes

تمارين عامة محلولة

Q1 - اكتب برنامجاً بلغة الـ C# يتضمن الصفوف التالية:

- صفئاً أساسياً وليكن `kia` يحتوي على المعطيات الخاصة `wheels` (عدد العجلات) و `range` (الوزن الفارغ) وهي من النوع `int` . ويحتوي الصف على بانٍ افتراضي يُسند القيمة الصفر للمعطيات الخاصة ، وبانٍ عادي بوسيطين من النوع `int` يسند قيم للمعطيات الخاصة ، وتابع `printKia` لطباعة قيم المعطيات الخاصة.
- صفئاً مشتقاً وليكن `kiaTaxi` يرث الصف الأساسي `kia` ويضيف عضو خاص من النوع `int` هو: عدد الركاب `P` ويحتوي على بانٍ عادي يسند قيم للمعطيات الخاصة ، وتابع عضو `print` للطباعة ، وتابع عضو وليكن `costWheels` لحساب ثمن عجلات الـ `kiaTaxi` ، علماً بأن ثمن العجلة الواحدة يساوي 20000 ل.س.

51

C# Programming

Dr.Mamou Younes

- صفئاً مشتقاً وليكن `kiaPickup` يرث الصف الأساسي `kia` ويحتوي على عضو خاص من النوع `int` هو: الحمولة `loadL` ويحتوي على أعضاء عامة هي: بانٍ عادي يسند قيم للمعطيات الخاصة ، وتابع عضو `print` للطباعة ، وتابع عضو وليكن `FullRange` لحساب الوزن القائم (الكلي) لـ `kiaPickup` .
- صفئاً للاختبار وليكن اسمه `KiaTest` يتضمن التابع الرئيسي `main` والذي من خلاله يتم إنشاء غرضين من الصفين: `kiaTaxi` و `kiaPickup` على الترتيب لتمرير القيم الموضحة في الجدول التالي :

الصف المشتق	wheel s	range	P	loadL
<code>kiaTaxi</code>	4	1200	4	
<code>kiaPickup</code>	6	2500		4000

- ومن ثم يُطلب طباعة كافة المعلومات التي تم إدخالها وأيضاً طباعة ثمن العجلات `kiaTaxi` والوزن الكلي لـ `kiaPickup` .

52

C# Programming

Dr.Mamou Younes

Q2- اكتب برنامجاً بلغة C# يتضمن مايلي :

- صفئاً أساسياً يمثل متوازي مستطيلات وليكن اسمه Box معطياته الخاصة L(طول القاعدة) ، W (عرض القاعدة)، H(الارتفاع) من النوع float، ويحتوي على التوابع العامة التالية:
- بانٍ افتراضي، وبانٍ عادي، وتابع عضو وليكن اسمه sq مهمته حساب مساحة قاعدة متوازي المستطيلات ومن النوع float، وتابع عضو من النوع float وليكن اسمه volume، مهمته حساب حجم متوازي المستطيلات، وتابع عضو وليكن اسمه print لطباعة قيم المعطيات الخاصة.
- صفئاً مشتقاً من الصف Box وليكن اسمه Box1 ويمثل هرمأ متوضعاً داخل متوازي المستطيلات قاعدته هي قاعدة متوازي المستطيلات ورأسه يقع في نقطة تقاطع قطري القاعدة العلوية لمتوازي المستطيلات وليس له معطيات خاصة ولكن يتضمن التوابع العامة التالية:

53

C# Programming

Dr.Mamou Younes

- بانٍ افتراضي، وبانٍ عادي، وتابع عضو وليكن اسمه volume1 لحساب حجم الهرم، وتابع عضو من النوع float وليكن اسمه volume2، مهمته حساب الحجم المتبقي من متوازي المستطيلات بعد اقتطاع الهرم، وتابع عضو وليكن اسمه printVolumes لطباعة قيم الأحجام التي تم حسابها.
- صفئاً للاختبار وليكن اسمه TextBox يتضمن إنشاء الأغراض اللازمة من أجل إدخال قيم للمعطيات الخاصة ومن ثم استدعاء ما يلزم من التوابع من أجل حساب وطباعة الأحجام المطلوبة وقيم المعطيات الخاصة للصف الأساسي.

54

C# Programming

Dr.Mamou Younes

Q3 - اكتب برنامجاً بلغة C# يتضمن الصفوف التالية:

- صفئاً أساسياً اسمه Circle يحتوي على العضو الخاص r (نصف قطر الدائرة) من النوع double . ويحوي هذا الصف على التوابيع الأعضاء العامة التالية :
بانٍ افتراضي يُسند قيمة الصفر ، وبانٍ عادي ، وتابع اسمه sq لحساب مساحة الدائرة ، وتابع اسمه printCircle لطباعة قيمة المعطى الخاص ومساحة الدائرة ، وتابع اسمه getRaduis للوصول إلى المعطى الخاص.
- صفئاً مشتقاً اسمه Cone (مخروط دوراني) يرث الصف الأساسي Circle ويحتوي على متحول خاص من النوع double هو h (ارتفاع المخروط) و يحتوي على الأعضاء العامة التالية:
بانٍ افتراضي ، و بانٍ عادي ، وتابع اسمه valumeCone لحساب حجم المخروط الدوراني ، وتابع اسمه PrintCone لطباعة قيمة المعطى الخاص وحجم المخروط وكل ما يتعلق بالصف الأساسي ، وتابع اسمه getHeight للوصول إلى المعطى الخاص .

55

C# Programming

Dr.Mamou Younes

- صفئاً مشتقاً اسمه Cylinder يرث الصف الأساسي Cone و يحتوي على الأعضاء العامة التالية:
بانٍ افتراضي يُسند القيمة صف للمعطيات الخاصة ، و بانٍ عادي ، وتابع اسمه valumeCylinder لحساب حجم الأسطوانة (حيث أن الارتفاع هو ضعف ارتفاع المخروط الدوراني) ، وتابع اسمه PrintCylinder لطباعة قيمة حجم الأسطوانة وكل ما يتعلق بالصفين Circle و Cone ، وتابع اسمه valumeExtra لحساب حجم الجزء المتبقي بعد حذف مخروطين متقابلين بالرأس من الأسطوانة آنفة الذكر.
- صفئاً للاختبار اسمه ValumeTest يحوي على التابع الرئيسي main ، والذي بدوره يتضمن إنشاء أغراض من الصف Cylinder من أجل تمرير قيم لنصف القطر والارتفاع ومن ثم إجراء الاستدعاءات اللازمة من أجل حساب مساحة الدائرة والأحجام المطلوبة وطباعتها .

56

C# Programming

Dr.Mamou Younes

Q4 - اكتب برنامجاً بلغة C# يتضمن:

- صفئاً أساسياً اسمه Equation يحتوي على ثلاثة متحولات خاصة a و b و c من نوع `float` , ويحتوي الصف على التوابع العامة التالية : بانٍ افتراضي يُسند الصفر للمعطيات الخاصة، وبانٍ عادي لتمرير المعطيات ، وتوابع وصول، وتابع عضو اسمه `delta` لحساب قيمة المميز b^2-4ac ، وتابع عضو اسمه `print` لطباعة قيم المعطيات الخاصة.
- صفئاً مشتقاً من الصف Equation وليكن اسمه `Roots` ويحتوي على متحولات خاصة x_1 و x_2 من النوع `float` ويحتوي الصف على التوابع العامة التالية: بانٍ افتراضي يُسند الصفر للمعطيات الخاصة، وبانٍ عادي لتمرير المعطيات ،تابع عضو اسمه `computeRoots` لحساب جذور المعادلة $ax^2 + bx + c = 0$ وإسناد النتائج لمعطيات الصف `Roots` ، وتابع عضو اسمه `printRoots` لطباعة قيم المعطيات الخاصة.
- أنشئ صفئاً للاختبار اسمه `Test` يحتوي على التابع الرئيسي `main` ، والذي يتضمن بدوره إنشاء أغراض من الصف `Roots` من أجل تمرير قيم لمعطيات الصفوف ومن ثم استدعاء التوابع اللازمة من أجل حساب وطباعة جذور المعادلة وقيم معطيات الصف الأساسي .

57

C# Programming

Dr.Mamou Younes

the end

The End

The End

The End

The End

The End

The End

The End

The End

58

C# Programming

Dr.Mamou Younes

تعدد الأشكال والواجهات
Polymorphism & Interfaces

Chapter 8
Dr.Eng.M.Younes

59

C# Programming

Dr.Mamou Younes

