



برمجة متقدمة 1

البرمجة غرضية التوجه - الوراثة



أ. فؤاد المهاوش

م. تغريد حرفوش

المعهد التقاني للحاسوب بحماة

الوراثة

البداية

إن الوراثة من أهم مرتكزات البرمجة غرضية التوجه، ومن خلالها يمكن إنشاء صفوف جديدة بالإعتماد على صفوف سابقة مما يسهل عمليات إعادة الاستخدام (reusability) وتكون الوراثة بلغة C# بالشكل التالي:

```
class Base
{

}

class Derived : Base
{

}
```

ان الحرف ":" يستخدم لتحديد بأن هذا الصف يرث من صف آخر، حيث أن الصف Derived يرث من الصف Base وبالتالي فان الصف الابن Derived يرث كافة الأعضاء الموجودة في لصف الأب Base باستثناء التابع الباني فلا يتم توريثه.

ملاحظة: على الرغم من أن الصف الابن يرث كافة الأعضاء في الصف الأب ولكنه لا يمكن أن يصل الى الأعضاء من ذات المحدد private

مثال

ليكن لدينا صف يعبر عن الأشخاص اسمه "Person" يحوي حقلين للإسم وتاريخ الميلاد وتابع يعيد نص يعبر عن الجمع بين الإسم مع تاريخ الميلاد.

```
class Person
{
    string name, dateOfBirth;
    public string Name
    { get { return name; } set { name = value; } }

    public string DateOfBirth
    { get { return dateOfBirth; } set { dateOfBirth = value; } }

    public string Greeting()
    {
        return "Hello My Name is:" + Name + "\n BirthDate" +
        DateOfBirth;
    }
}
```

والآن سننشئ صف يعبر عن الطالب "Student" الذي سيملك نفس الحقول الموجودة لدى الصف "Person" من خلال عملية الوراثة بالإضافة إلى حقل جديد مميز يمثل رقم التسجيل للطالب:

```
class Student : Person
{
    private int registrationNum;
    public int RegistrationNum
    { get { return registrationNum; } set { registrationNum =
value; } }
}
```

وبالتالي يمكننا انشاء غرض من صف الطالب وهذا الغرض سيحوي جميع الخاصيات والطرائق الموجودة بالصف الأب والصف الابن:

```
static void Main(string[] args)
{
    Student s1 = new Student();
    s1.Name = "New Student";
    s1.DateOfBirth = "17-07-1990";
    s1.RegistrationNum = 37;
    Console.WriteLine(s1.Greeting());
}
```

التابع الباني في الوراثة

عند انشاء غرض من صف مشتق فإنه يتم استدعاء التابع الباني للأب أولاً ثم التابع الباني للإبن، وعند عدم تحديد تابع باني للصف يكون هناك تابع باني افتراضي لا يأخذ أي معامل ولا يعيد أي نتيجة ولكن في حال تم تحديد تابع باني يأخذ معاملات، وكان هذا الصف أب لصف آخر، فيجب علينا في الصف الإبن تمرير هذه المعاملات، وإلا فسيحدث خطأ برمجي مفاده أن التابع الأب لا يملك تابع باني لا يقبل أي معاملات.

مثلاً لو عدلنا الصف "Person" ليملك التابع الباني الذي يأخذ معاملين كما يلي:

```
class Person
{
    string name, dateOfBirth;

    public Person(string name, string dateOfBirth)
    {
        this.name = name;
        this.dateOfBirth = dateOfBirth;
    }

    public string Name
    { get { return name; } set { name = value; } }

    public string DateOfBirth
    { get { return dateOfBirth; } set { dateOfBirth = value; } }

    public string Greeting()
    {
        return "Hello My Name is:" + Name + "\n BirthDate" +
        DateOfBirth;
    }
}
```

فيجب علينا أن ننشئ تابع باني في الصف الإبن ليستدعي الباني في الأب ويمرر له المتحولات اللازمة.

ويكون ذلك بالكلمة المفتاحية "base" كما يلي:

```
class Student : Person
{
    private int registrationNum;

    public Student(string name, string dateOfBirth, int registrationNum)
        : base(name,dateOfBirth)
    {
        this.registrationNum = registrationNum;
    }

    public int RegistrationNum
    { get { return registrationNum; } set { registrationNum = value; } }
}
```

الكلمة المفتاحية "base" تمثل استدعاء للتابع الباني للأب، وليست تعريف جديد للتابع، ولذلك نحن لا نعرف معاملات جديدة وإنما نمرر المتحولات اللازمة بحسب التعريف الخاص للتابع الباني في الصف الأب.

الكلمة المفتاحية Sealed

عند بناء صف وتمييزه بالكلمة المفتاحية sealed فهذا يعني أن هذا الصف "عقيم" ولا يمكن أن يرث منه أي صف آخر.

مثلاً لو كان لدي صفين أحدهما يرث من الآخر كما يلي:

```
class A
{
}

class B : A
{
}
```

لو عدلنا الصف الأب A بالكلمة المفتاحية Sealed فسيحدث خطأ في عملية الوراثة:

```
sealed class A
{
}
class B : A
{
}
```

'B': cannot derive from sealed type 'A'

تعددية الأشكال Polymorphism

هي قدرة الصفوف على تنفيذ مهام مختلفة للطرائق التي تملك نفس الاسم ويتم استدعاؤها بذلك الاسم

استخدام نمط الصف الأب لإنشاء غرض من نمط الصف الابن

مثال

```
class A
{
    public void MethodA()
    {
    }
}

class B : A
{
    public void MethodB()
    {
    }
}
```

وفي لغة C# يمكننا استخدام نمط الصف الأب لإنشاء غرض من نمط الصف الابن:

```
A a = new B();
```

لاحظ انشاء غرض من الصف الابن ولكن من نوع الصف الأب, وبشكل أوضح نقول أن مؤشر من النمط الأب يؤشر على غرض من النمط الابن
وهذا ممكن في C# وهنا يجب أن نعامل هذا الغرض كغرض من الصف الأب وليس الابن.
أي أنه يمكننا أن نصل الى أعضاء الصف الأب مثلاً:

```
a.MethodA();
```

ولا يمكن أن نصل الى أعضاء الصف الابن أي:

```
a.MethodB(); // Error
```

استخدام الطرائق التي لها نفس الاسم في الصف الأب والصف الابن

لو كان لدينا طريقتين لهما نفس الاسم (نفس الترويسة) في الصف الأب والصف الابن وقمنا بانشاء غرض من الصف الابن باستخدام مؤشر من نوع الصف الأب , وقمنا باستدعاء هذا الطريقة (الموجودة في كلا الصفيين) فإنه سيتم تنفيذ الطريقة الموجودة في الصف الأب مثال :

```
class Shape
{
    public void Draw()
    {
        Console.WriteLine("This is class Shape");
    }
}

class Circle : Shape
{
    public void Draw()
    {
        Console.WriteLine("This is class Circle");
    }
}
```

ولو قمنا بتعريف غرض كما يلي:

```
static void Main(string[] args)
{
    Shape s1 = new Circle();
    s1.Draw();
}
```

سيكون الناتج كما يلي:



```
This is class Shape
```

الكلمات المفتاحية Virtual و Override

لو أردنا تجاهل الطريقة الموجودة في الصف الأب وتنفيذ الطريقة الموجودة في الصف الابن يتوجب علينا وضع الكلمة المفتاحية virtual عند الطريقة الموجودة في الصف الأب وكلمة Override عند الطريقة الموجودة في الصف الابن التي سيتم تنفيذها فعلياً وهذا يعني بأن الطريقة الموجودة Virtual بالصف الأب إفتراضية وفي حال وجد مقابل لها لها override في الابن فسيتم تنفيذ الطريقة في الصف الابن:

```
class Shape
{
    virtual public void Draw()
    {
        Console.WriteLine("This is class Shape");
    }
}

class Circle : Shape
{
    override public void Draw()
    {
        Console.WriteLine("This is class Circle");
    }
}
```


وعند تنفيذ التعليمات التالية:

```
static void Main(string[] args)
{
    Shape s1 = new Circle();
    s1.Draw();
}
```

سيصبح الناتج:



ملاحظات:

يمكن استخدام أكثر من override لطريقة virtual واحدة وذلك اذا كان هناك أكثر من صف يرثوا من صف ما ويكون نتيجة التنفيذ حسب نوع الغرض المنشأ وليس حسب نوع المؤشر .
يمكن استخدام الكلمة المفتاحية New عند الطريقة في الصف الابن التي لها نفس اسم الطريقة في الصف الأب
للدلالة على أنه لا يوجد تعددية أشكال .