التركيب
Composition

الكلمات المفتاحية

التركيب، التجميع، الصفوف المتداخلة، الصفوف الجزئية.

ملخص الفصل

خُصِّص الفصل الحالي لشرح مفهوم التركيب كأحد المفاهيم التي تدعم إعادة استخدام الرمّاز المصدري، فقد يحتاج صف ما إلى أغراض نمط معطياتها هو صف آخر.

ولتوضيح الفرق بينه وبين العلاقات الأخرى التي تربط بين الصفوف.

الأهداف التعليمية

يتعرّف الطالب في هذا الفصل على:

- التركيب وكيفية استخدامه في لغة C#
- الفارق بين التركيب والتجميع
- الصفوف المتداخلة وكيفية تعريفها
- الصفوف الجزئية وكيفية استخدامها

مقدمة

يفرض مبدأ إعادة الاستخدام Reusability نشوء علاقات بين الصفوف، ومن بين هذه العلاقات التركيب والتجميع والتداخل والتجزئة.

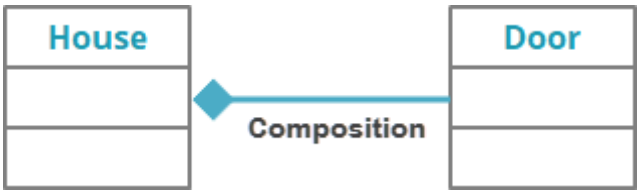
1. مفهوم التركيب

من المعلوم أنّه عند كتابة صفٍ جديد، يتم إنشاء نمط بيانات Data Type جديد يمكن استخدامه لإنشاء أغراض منه. فعلى سبيل المثال، عند كتابة الصف Door (باب) يمكن استخدامه ضمن الصف House (منزل) كأحد الأعضاء المكوّنة له. نسمّي العلاقة التي تربط بين هذين الصفين بالتركيب Composition لأنّ كل غرض من الصف House سيكون من بين أعضائه غرضاً من الصف Door.

```
class Door
{
//....
}

class House
{
Door door = new Door();
//.....
}
```

وتُمثّل العلاقة بين الصفين بـ "composed-of" أي يتركّب من، فمفهوم التركيب يُعبِّر عن امتلاك صف لأعضاء من صفوف أخرى. وعند إنشاء غرض من الصف House سيتمّ إنشاء غرض من الصف Door له كجزء منه. وطالما أنّ الغرض الأول موجود في الذاكرة، سيبقى الغرض الثاني موجوداً أيضاً. أمّا عند تدمير الغرض الأول، سيؤدي ذلك إلى تدمير الغرض الثاني لأنّه أحد مكوّناته.



مثال:

لو قمنا بإنشاء الصف Date (تاريخ) والصف Employee (موظف)، فإنّه من الممكن للصف Employee أن يحتوي على أكثر من عضو من النمط Date، مثل BirthDate المعبِّر عن تاريخ الميلاد وHireDate المعبِّر عن تاريخ التوظيف.

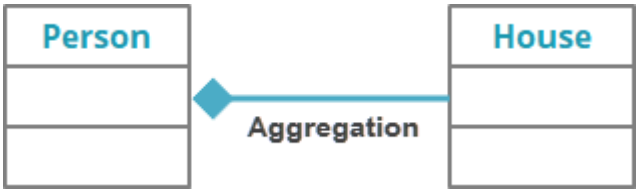
```
class Date
{
//....
}

class Employee
{
    //....
    Date BirthDate = new Date();
    Date HireDate = new Date();
    //....
}
```

وعند تدمير الغرض المنتَسَخ من الصف Employee، سيتمّ تدمير الغرضَين المنتَسخَين من الصف Date.

2. التجميع

لا بدّ لنا من التمييز بين التركيب Composition والتجميع Aggregation على الرّغم من استخدامهما في بعض الأحيان كمفهومين متكافئين.



فالتجميع يُمثَّل بالعلاقة "has-a" أي يمتلك، فلو قمنا بإنشاء الصف Person (شخص) الذي يمتلك غرضاً من الصف House كما هو موضّح فيما يأتي:

```
public class House
{
      //....
      }
public class Person
{
            private House house;
            public Person(House house)
            {
                this.house = house;
          }
            //....
      }
```

وقمنا باستخدام الصفين على النحو الآتي:

```
House house = new House();
Person person = new Person(house);
```

عندها سيتمّ إنشاء الغرض house من الصف House والغرض person من الصف Person الذي يمتلك الغرض house، وعند تدمير الغرض person لن يؤدّي ذلك إلى تدمير الغرض house، ويمكن إنشاء غرض آخر من الصف Person يمتلك نفس الغرض house.

3. الصفوف المتداخلة

يمكن إنشاء صفٍ داخل صفٍ آخر ليشكِّل أحد عناصره، فيُطلَق عليه بالصف المتداخِل Nested Class. وعند ذلك، يمكن لعناصر الصف المتداخِل أن تصل إلى جميع عناصر الصف الحاوي له سواء أكان محدِّد الوصول إليها private أو protected أو public أو غيرها. ففي المثال الآتي، يمثِّل الصف Nested أحد أعضاء الصف Container:

```
public class Container
{
    //....
    class Nested
{
        Nested() { }
        //....
    }
}
```

وكغيره من أعضاء الصف الحاوي له، يمكنه الوصول إلى جميع المكوّنات البرمجية التي يُسمح للصف الحاوي له بالوصول إليها.

ويكون محدّد الوصول الافتراضي إلى الصف المتداخل private، ويمكن استبداله بأي محدّد وصول آخر مثل public أو protected أو internal أو غير ذلك. ففي المثال الآتي، للصف المتداخل محدّد الوصول public وللصف الحاوي له محدّد الوصول public أيضاً:

```
public class Container
{
    public class Nested
    {
        Nested() { }
    }
}
```

ولإنشاء غرض من الصف المتداخل، نستخدم اسمه الكامل، فلإنشاء الغرض nest من الصف Nested المتداخل ضمن الصف Container، نكتب:

```
Container.Nested nest = new Container.Nested();
```

ويمكن وصول الصف المتداخل إلى عناصر الصف الحاوي له من خلال تعريف بانٍ يأخذ كوسيط دخل له غرضاً من الصف الحاوي، ويتمّ ذلك كما يأتي:

```
public class Container
{
    public class Nested
    {

        private Container parent;

        public Nested()
        {
        }// end default constructor

        public Nested(Container parent)
        {
            this.parent = parent;

        }// end constructor with one object of Container class
    }// end class Nested
}// end class Container
```

4. الصفوف الجزئية

نحتاج في بعض الحالات إلى تجزئة صف ما إلى عدّة صفوف جزئية Partial Classes، كالحالة التي يكون فيها حجم الصف كبيراً جداً ويكون من الأفضل تجزئته على عدّة ملفّات بحيث يمكن لمجموعة مبرمجين العمل عليه في نفس الوقت. وتتمّ تجزئة صف باستخدام المحدِّد partial مع كلّ جزء، كما في المثال الآتي:

```
public partial class Employee
{
    public void DoWork() { }
}

public partial class Employee
{
    public void GoToLunch() { }
}
```

ويجب على جميع الأجزاء أن تكون متاحة عند الترجمة، كما يجب أن يكون لها جميعاً نفس محدِّد الوصول، ويجب أن ترد الكلمة partial مباشرة قبل الكلمة class.

وفي حال تمّ التصريح عن أحد الأجزاء بأنّه مجرّد abstract، يتمّ التعامل مع الصف بأكمله كصف مجرّد أي أنّ جميع الأجزاء تصبح مجرّدة. وكذلك إذا تمّ التصريح عن أحد الأجزاء بأنّه محكم الإغلاق sealed، يتمّ التعامل مع الصف بأكمله كصف محكم.

وعندما يقوم أحد الأجزاء بوراثة صف آخر، سينعكس ذلك على جميع الأجزاء لتصبح وارثة له ويصبح بذلك الصف بأكمله وارثاً له. وعندما ترث الأجزاء واجهات مختلفة، يصبح الصف بأكمله وارثاً لجميع هذه الواجهات. وكلّ بنية struct أو واجهة interface معرّفة ضمن أحد الأجزاء، تصبح عضواً متاحاً للصف بأكمله. ونوضّح ذلك من خلال ما يأتي:

```
partial class Earth : Planet, IRotate { }
partial class Earth : IRevolve { }
```

والأسطر السابقة مكافئة للسطر:

```
class Earth : Planet, IRotate, IRevolve { }
```

ويمكن للصف المتداخل أن يكون مجزّأً أيضاً، سواءٌ أكان الصف الحاوي له مجزّأً أم لا. وتوضِّح التعليمات الآتية الصف المجزّأ Nested المتداخل ضمن الصف Container:

```
class Container
{
    partial class Nested
    {
        void Test() { }
    }
    partial class Nested
    {
        void Test2() { }
    }
}
```

أما التعليمات الآتية، فتوضّح الصف المجزّأ NestedClass المتداخِل ضمن الصف المجزّأ ContainerClass:

```
partial class ContainerClass
{
    partial class NestedClass { }
}
partial class Container
{
    partial class NestedClass { }
}
```

ملاحظة: الصفوف المجردة abstract والمحكمة sealed والواجهة interface والبنية struct ستتم دراستها لاحقاً.

5. أمثلة عن التركيب

مثال1:

في الرمّاز الآتي، نوضّح علاقة التركيب التي تربط الصف Door مع الصف House والتي تعني يتركّب من، وعلاقة التجميع التي تربط بين الصف House والصف Person والتي تعني يمتلك.

```
using System;
namespace Composition
{
    class Door
    {
        public string Color { get; set; }
        // Defining parameterless constructor
        public Door() { Color = "Blue"; }
        // Defining constructor with one argument
        public Door(string color) { Color = color; }
        public void ShowData()
        {
            Console.WriteLine("I am a door, my color is {0}.", Color);
        }
    }// end class Door
    class House
    {
        public int Area { get; set; }
        //Door member as auto-implemented Property
        public Door Door { get; set; }
        // Define constructor with one argument
        public House(int area)
        {
            this.Area = area;
            Door = new Door(); //Creating a Door object
        }
        // Method for printing the house area
        public void ShowData()
        {
            Console.WriteLine("I am a house, my area is {0} m2.", Area);
        }
    }// end class House
```

```
class Person
{
    // Static field, will be utilized to count
    // the objects instantiated from the class
    private static int instances = 0;
    // static member
    public string Name { get; set; }
    // House member as auto-implemented Property
    public House House { get; set; }
    // Defining constructor with two arguments
    public Person(string name, House house)
    {
        instances++; // Incrementing the objects number
        this.Name = name;
        this.House = house;
    }
    public void ShowData()
    {
        Console.WriteLine("\n\nMy name is {0}, instances counter is {1}"
             , Name, instances);
        House.ShowData();
        House.Door.ShowData();
    }
}// end class Person
class Test
{
    static void Main()
    {
        // creating a house, its name is SmallHouse and its area is 60
        House SmallHouse = new House(60);
        // creating a house, its name is BigHouse and its area is 160
        House BigHouse = new House(160);
        // creating a person, his name Reem and his house is BigHouse
        Person Person1 = new Person("Reem", BigHouse);
        // creating a person, his name Rami and his house is SmallHouse
        Person Person2 = new Person("Rami", SmallHouse);
        // changing the default door color to red
        Person2.House.Door.Color = "red";
        //Printing the information of both persons
```

```
            Person1.ShowData();
            Person2.ShowData();
            Console.ReadKey();
        }// end Main
    }// end class Test
}// end namespace Composition
```

- يضمّ الصف Door الخاصّية Color التي تشير إلى اللون، وبانٍ افتراضي يسند القيمة blue إلى الخاصّية Color، وبانٍ آخر يأخذ قيمة اللون كدخل له ويسندها إلى الخاصّية.
- يضمّ الصف House الخاصّية Area المعبّرة عن مساحته والخاصّية Door الممثّلة لباب المنزل وهي من النمط Door، ونلاحظ أنّ الخاصّية تحمل نفس اسم نمط البيانات Door ولم يؤدِّ ذلك إلى إصدار رسالة خطأ.
- يأخذ باني الصف House قيمة صحيحة كدخل له ويسندها إلى الخاصّية Area، ويقوم بإنشاء غرض من الخاصّية Door باستخدام الباني الافتراضي للصف Door، وهنا تتّضح علاقة التركيب.
- يمتلك الصف House الطريقة ShowData التي تطبع رسالة متضمّنة لمساحة المنزل.
- يضمّ الصف Person الحقل الساكن instances الدال على عدد الأشخاص الذين تمّ إنشاؤهم، والخاصّية Name الدالة على اسم الشخص، والخاصّية House الدالة على المنزل الذي يمتلكه الشخص. ويضمّ الطريقة ShowData التي تطبع رسالة متضمّنة الاسم وعدد الأشخاص ومعلومات (مساحة) المنزل الذي يمتلكه ومعلومات (لون) باب المنزل.
- للصف Person بانٍ يزيد عدد الأشخاص بمقدار(1) عند إنشاء شخص جديد، ويأخذ وسيطي دخل، الأول يمثل الاسم والثاني يمثل المنزل الذي يمتلكه.
- ضمن الطريقة Main من الصف Test، يتمّ إنشاء المنزل SmallHouse ذي المساحة(60) والمنزل BigHouse ذي المساحة(160). ويتم إنشاء الشخص Person1 الذي اسمه Reem ويمتلك BigHouse، والشخص Person2 الذي اسمه Rami ويمتلك المنزل SmallHaouse، ويتمّ تغيير لون منزل Rami إلى اللون الأحمر red. وأخيراً، تتمّ طباعة معلومات كلا الشخصين من خلال استدعاء الطريقة ShowData لكلّ منهما.

وبعد التنفيذ، نحصل على الخرج الآتي:

```
My name is Reem, instances counter is 2
I am a house, my area is 160 m2.
I am a door, my color is Blue.

My name is Rami, instances counter is 2
I am a house, my area is 60 m2.
I am a door, my color is red.
```

مثال2:

في الرمّاز الآتي، نوضِّح مفهوم الصفوف المتداخلة، فنعرِّف الصف Second ضمن الصف First، فيصبح الصف First حاوٍ للصف المتداخل Second.

```
using System;
namespace NestedClasses
{
    class First
    {
        private int num1;
        public int Getnum1() { return num1; }

        public class Second
        {
            public int num2;
            First f1 = new First();
            public void PrintData()
            {
                First f2 = new First();
                // nested class can access private members of
                // enclosing class. It can access private  and
                // protected members of the enclosing class,
                // including any inherited protected members.
                ++f1.num1;
                Console.WriteLine("Increment f1.num1 {0} ", f1.num1);
                // Console.WriteLine("Increment num1 {0} ", num1);//Error
                Console.WriteLine("Increment f2.num1 {0} ", ++f2.num1);
            }
```

```
        }// end class Second
    }//end class First
    class Tester
    {
        static void Main()
        {
            // creating object from First class
            First a = new First();
            // calling Getnum1 method
            Console.WriteLine(a.Getnum1());

            // creating object from Second class
            // we use the name of enclosing class followed by the name
            // of nested class
            First.Second ab = new First.Second();
            // accessing field found in the nested class Second
            Console.WriteLine(++ab.num2);
            // calling PrintData from Second class
            ab.PrintData();
            Console.ReadKey();
        }// end Main
    }// end Tester
}//end namespace NestedClasses
```

ويضمّ الصف First، إضافة للصف Second، الحقل num1، والطريقة Getnum1 التي تعيد قيمة الحقل. أمّا الصف Second، فيضمّ الحقل num2 والغرض f1 من النمط First والطريقة PrintData التي تحوي بدورها غرضاً من النمط First واسمه f2. وتقوم الطريقة بزيادة f1.num1 بمقدار (1) ثم تطبع قيمته مرفقة برسالة توضيحية، وتكرّر نفس الأمر مع f2.num1.

وفي الطريقة Main التابعة للصف Tester، يتمّ إنشاء الغرض a من النمط First، وطباعة قيمة حقله a.num1 من خلال استدعاء الطريقة a.Getnum1. ويتمّ إنشاء الغرض ab من النمط Second، وطباعة قيمة حقله ab.num2 بعد زيادة قيمته بمقدار (1)، ثمّ يتمّ استدعاء الطريقة ab.PrintData لطباعة قيمتي الحقلين f1.num1 و f2.num1 بعد زيادة كلّ منهما بمقدار (1)، حيث تمّ إنشاء f1 و f2 ضمن الصف Second.

وبعد التنفيذ نحصل على الخرج الآتي:

```
0
1
Increment f1.num1 1
Increment f2.num1 1
```

نلاحظ أن الطريقة PrintData تمكّنت من الوصول إلى الحقل num1 ضمن الصف First على الرّغم من أنّ محدّد وصوله private.

وعند محاولة وصول الصف المتداخل Second مباشرة إلى أحد عناصر الصف الحاوي له (الحقل num1)، نحصل على رسالة الخطأ:

```
An object reference is required for the non-static field, method,
or property 'First.num1'
```

والتي تفيد بضرورة إنشاء غرض من الصف، واستخدام الحقل الخاص بالغرض المُنشَأ.

مثال3:

في الرمّاز الآتي، نوضِّح كيفية استخدام الصفوف الجزئية:

```
using System;
namespace PartialClasses
{
    // The first part
    public partial class Point
    {
        private int x;
        private int y;
        public Point(int x, int y)
        {
            this.x = x;
            this.y = y;
        }
    }// partial class Point 1
    // The second part
    public partial class Point
    {
```

```
        public void PrintPoint()
        {
            Console.WriteLine("Point: {0},{1}", x, y);
        }
    }//partial class Point 2
    class TestPoint
    {
        static void Main()
        {
            Point myPoint = new Point(10, 15);
            myPoint.PrintPoint();
            // Keep the console window open in debug mode.
            Console.ReadKey();
        }// end Main
    }// end TestPoint
}// end namespace PartialClasses
```

ينقسم الصف Point إلى جزأين، يضمّ الأول الإحداثيات الخاصّة x وy وبانٍ لإسناد قيم لهما، ويضمّ الجزء الثاني الطريقة PrintPoint لطباعة قيم الإحداثيات. وفي الطريقة Main ضمن الصف TestPoint، يتمّ إنشاء myPoint ذي الإحداثيات 10 و15 كغرض من الصف Point، ثمّ يتمّ استدعاء الطريقة PrintPoint.

وبعد تنفيذ الرمّاز السابق، نحصل على الخرج الآتي:

```
Point: 10,15
```

مثال 4:

فيما يأتي، نوضّح كيفية تجزئة البنية struct والواجهة interface (ستتمّ دراستهما في فصول قادمة).

```
using System;
namespace PartialStructInterface
{
    // First part of The interface
    partial interface ITest
    {
        void Interface_Test();
    }// end interface ITest 1

    // Second part of The interface
    partial interface ITest
    {
        void Interface_Test2();
    }// end interface ITest 2
    // First part of The struct
    partial struct S
    {
        void Struct_Test() { }
    }// end struct S 1
    // Second part of The struct
    partial struct S
    {
        void Struct_Test2() { }
    }// end struct S 2
}// end namespace PartialStructInterface
```

ولا تقتصر العلاقات بين الصفوف على ما تمت دراسته في هذا الفصل، بل يمكن أن ترتبط الصفوف أيضاً بعلاقة وراثة Inheritance. وتُعتبر الوراثة بين الصفوف من أهم الأسس التي تقوم عليها البرمجة غرضية التوجه وسنخصص لها الفصل القادم.

الأنشطة المرافقة

التمرين الأول: صف التاريخ Date وصف الموظف Emolyee

1. قم بتعريف الصف Date وضمنه الأعضاء الآتية:
   - الحقل day ويجب أن تنتمي قيمه إلى المجال [1-31].
   - الحقل month ويجب أن تنتمي قيمه إلى المجال [1-12].
   - الخاصية التلقائية Year ولا يُسمَح بتغيير قيمتها من خارج الصف.
   - الخاصية Month الموافقة للحقل month وتقذف استثناءً في حال كانت القيمة المسندة إليها لا تنتمي إلى المجال المطلوب.
   - الخاصية Day الموافقة للحقل day وتقذف استثناءً في حال كانت القيمة المسندة إليها لا تنتمي إلى المجال المطلوب، أو في حالة الشهر هو شباط والقيمة المسندة إليها لا تنتمي إلى المجال [1-28]، أو في حالة كانت السنة كبيسة، قيمة Year من مضاعفات الـ(4)، والشهر هو شباط والقيمة المسندة إليها لا تنتمي إلى المجال [1-29].
   - الطريقة ToString التي تعيد سلسلة محرفية تمثل التاريخ مكتوباً بالصيغة dd-mm-yyyy حيث يمثل dd اليوم وmm الشهر وyyyy السنة.
   - بانٍ يأخذ ثلاثة قيم كوسائط دخل له ويقوم بإسنادها إلى الخاصيات الثلاث السابقة الذكر.

2. قم بتعريف الصف Employee وضمنه الأعضاء الآتية:
   - الحقل الساكن count الذي يشير إلى عدد الموظفين.
   - الخاصية FirstName ولا يُسمَح بتغيير قيمتها من خارج الصف.
   - الخاصية LastName ولا يُسمَح بتغيير قيمتها من خارج الصف.
   - الخاصية BirthDate من النمط Date ولا يُسمَح بتغيير قيمتها من خارج الصف.
   - الخاصية HireDate من النمط Date ولا يُسمَح بتغيير قيمتها من خارج الصف.
   - الطريقة ToString التي تعيد سلسلة محرفية تمثل معلومات الموظف (الغرض)الذي يستدعيها.
   - بانٍ يأخذ أربعة قيم كوسائط دخل له ويقوم بإسنادها إلى الخاصيات الأربع السابقة الذكر.

3. اكتب صفاً بلغة C# وسمِّهِ EmployeeTest، وضمّنه الطريقة Main واستخدمه لاختبار الطرائق التي عرفتها في الصفين السابقين.

التمرين الثاني: صف الحساب Account والصف الجزئي Emolyee

1. قم بتعريف الصف Account وضمنه الأعضاء الآتية:
   - حقل الرصيد balance والخاصية Balance الموافقة له والتي يجب أن تكون القيم المسندة إليها موجبة.
   - الطريقة Credit والتي تأخذ وسيط دخل وحيد يمثل مبلغاً يجب إضافته إلى الرصيد.
   - بانٍ يأخذ وسيط دخل وحيد يمثل الرصيد الأولي للحساب.
2. قم بتعديل صف الموظف السابق بحيث يصبح صفاً جزئياً، ثم عرِّف جزءاً ثانٍ له يضم الأعضاء الآتية:
   - الخاصية Account من النمط Account ويجب إسناد قيمة أولية لها عن طريق الباني.
   - الحقل socialSecurityNumber المعبّر عن رقم الضمان الصحي.
   - الحقل grossSales المعبّر عن ثمن المبيعات التي قام بها الموظف.
   - الحقل commissionRate المعبّر عن العمولة التي ستُدفع للموظف كنسبة مئوية من مبيعاته.
   - الحقل salary الممثل للراتب الأساسي للموظف.
   - الطريقة العامة Earnings والتي مهمتها حساب مستحقات الموظف الشهرية.
3. عدّل الطريقة Main في الصف EmployeeTest واختبر الأعضاء التي أضفتها في الصفين السابقين.

المراجع

1. https://www.c-sharpcorner.com/article/difference-between-composition-and-aggregation/, Updated date: May 10, 2019, written by: Phil Curnow.
2. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/partial-classes-and-methods, Updated date: Jul 20, 2015.
3. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/nested-types, Updated date: Abr 08, 2020.
4. Dan Clark: Beginning C# Object-Oriented Programming, Berkeley, CA, Apress, 2013.
5. "التصميم والبرمجة غرضية التوجّه"، الدكتور سامي خيمي، الإجازة في تقانة المعلومات، من منشورات الجامعة الافتراضية السورية، الجمهورية العربية السورية، 2018.