การเขียนโปรแกรมเชิง object (2)

Suphot Sawattiwong
tohpus@gmail.com

Accessibility

- คือ การเข้าถึงข้อมูลและ method ใน object ซึ่งเป็นการกำหนดสิทธิ์ของข้อมูลและ method รวมทั้งปิดกั้นข้อมูลไม่ให้ออกจาก object ได้หลายช่องทาง
- การเขียนโปรแกรมเชิงobject จำเป็นต้องปิดกั้นข้อมูลให้มีการเข้าออกเป็นช่องทางไป เนื่องจากเกี่ยวข้องกับความปลอดภัยของข้อมูล
- ใน C# การเข้าถึงสามารถทำได้โดยใช้ Access Modifiers ซึ่งใน C# มี public, private, internal และ protected

Access Modifiers

- คือการกำหนดสิทธิ์ในการเข้าออกข้อมูลและ method หรือเป็นการควบคุมการเรียกใช้ส่วนประกอบต่างๆใน class
- ใน C# มี Access Modifiers อยู่ 4 แบบ
  - public
  - private
  - internal (ไม่ได้กล่าวถึง ใน วิชานี้)
  - protected (ไม่ได้กล่าวถึง ใน วิชานี้)

public

- การประกาศ public ข้างหน้า method หรือ ตัวแปร จะถือว่าเป็นสถานะ public
- คือการกำหนดสิทธิและการอนุญาตใช้งานให้แก่ method กับ ตัวแปร **ให้ใช้ได้อย่างอิสระ ไม่ว่า class ใดก็สามารถเรียกใช้ได้**

ตัวอย่าง การใช้ public (Program.cs)

```
using System;
namespace ClassPublic
{
    class MyClass
  {
        public int x;
        public int y;
  }
    class Program
  {
        public static void Main()
    {

            MyClass myobject = new MyClass();
            myobject.x = 10;
            myobject.y = 20;
            Console.WriteLine("x = " + myobject.x);
            Console.WriteLine("y = " + myobject.y);
            Console.ReadLine();
    }
  }
}
```

private

- การประกาศ private ข้างหน้า method หรือ ตัวแปร จะถือว่าเป็นสถานะ private
- หรือ การไม่ใส่ access modifiers ลงไปในตัวแปร หรือ method จะถือว่าเป็น private เช่นเดียวกัน
- คือการกำหนดสิทธิและการอนุญาตใช้งานให้แก่ method กับ ตัวแปร **<u>ให้ใช้ได้เฉพาะใน class เท่านั้น</u>**

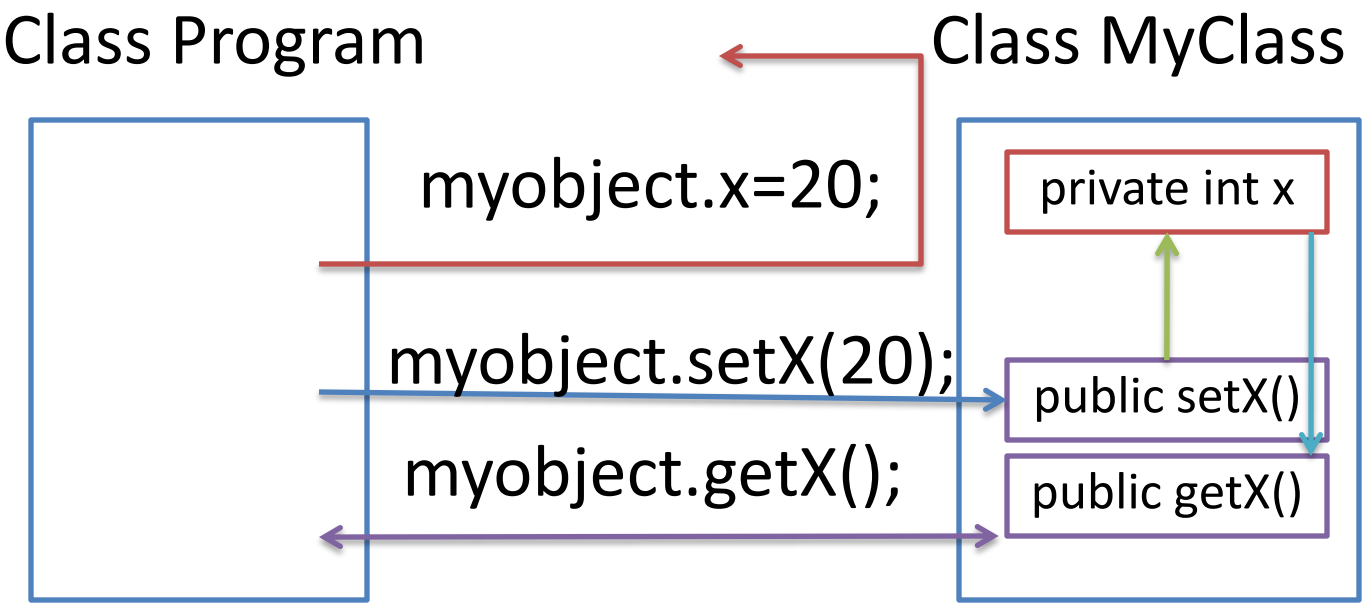
ตัวอย่าง ClassPrivate (MyClass.cs)

```
using System;
namespace ClassPrivate
{
    class MyClass
    {
        private int x;
        int y;
        public void setX(int x)
        {
            this.x = x;
        }
        public void setY(int y)
        {
            this.y = y;
        }
        public int getX()
        {
            return this.x;
        }
        public int getY()
        {
            return this.y;
        }
    }
}
```

ตัวอย่าง ClassPrivate (Program.cs)

```
using System;
namespace ClassPrivate
{
    class Program
    {
        public static void Main()
        {
            MyClass myobject = new MyClass();
            myobject.setX(10);
            myobject.setY(20);
            Console.WriteLine("x = " + myobject.getX());
            Console.WriteLine("y = " + myobject.getY());
            Console.ReadLine();
        }
    }
}
```

จากตัวอย่าง


Class Program
Class MyClass
myobject.x=20;
private int x
myobject.setX(20);
public setX()
myobject.getX();
public getX()

การสร้าง property บน C#

- หากต้องมานั่งตั้งค่า get กับ set เหมือนตัวอย่าง เราคงต้องเสียเวลามาก หากมีตัวแปรหลายๆตัว
- C# มีวิธีให้เราทำงานง่ายขึ้นโดยการทำดังต่อไปนี้

```
int data;
public int Data
{
    get { return data; }
    set { data = value; }
}
```

หรือ

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

ตัวอย่างการใช้ property

```
using System;
namespace PropertiesTest
{
    class TestObject
    {
        //public int Data { get; set; }
        int data;
        public int Data
        {
            get { return data; }
            set { data = value; }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            TestObject x = new TestObject();
            x.Data = 10;
            Console.WriteLine(x.Data);
            Console.ReadLine();
        }
    }
}
```

Static กับ Dynamic

- static กับ dynamic ก็เป็น modifier เหมือนกัน แต่ ไม่ใช่ access modifiers เท่านั้น
- ตัวแปร ที่เป็น static คือ ตัวแปรที่เป็นของ class ไม่ใช่ตัวแปรที่เป็นของ instance ใด instance หนึ่ง
- หากตัวแปร static มีการเปลี่ยนแปลงค่า ทุก instance ของ class นั้น ค่าตัวแปร static นี้ก็เปลี่ยนไปด้วยเช่นกัน

Static กับ Dynamic

- ตัวแปร dynamic คือ ตัวแปรทั่วไปที่เราใช้กัน เพียงแต่เราไม่ได้ระบุคำว่า dynamic ลงไปข้างหน้าเท่านั้น
- static method คือ method ที่ถูกสร้างขึ้นเพื่อทำงานแบบตัวแปร static หรือ method ที่ถูกเรียกใช้ผ่านทาง class เท่านั้น

Class เป็น Type

- จากที่ผ่านมา จะเห็นได้ว่า class เปรียบเสมือน กับ ชนิดของตัวแปร ที่เราสามารถกำหนดเองได้
- class ที่เราสร้างกันมา วิธีใช้งานของมันเหมือนกับตัวแปรทั่วไปที่เราใช้
- เราสามารถใช้งาน class ได้เหมือนกับตัวแปรชนิด int, float, string
- จำไว้ว่า

class = ประเภทข้อมูล (type)
ตัวแปร = instance = attribute = ตัวแปรใน class
function = method

ตัวอย่าง (Num.cs)

```
using System;
namespace ClassNumExample
{
    class Num
    {
        int data;
        public Num(int data)
        {
            this.data = data;
        }
        public int Data
        {
            get { return data; }
            set { this.data = value; }
        }
        public static Num FindMax(Num x, Num y)
        {
            if (x.Data > y.Data) return x;
            return y;
        }
    }
}
```

ตัวอย่าง (Program.cs)

```
using System;
namespace ClassNumExample
{
    class Program
    {
        static void Main(string[] args)
        {
            Num a = new Num(7);
            Num b = new Num(5);
            Num max = Num.FindMax(a,b);
            Console.WriteLine(max.Data);
            Console.ReadLine();
        }
    }
}
```

Overloading Method

- คือการสร้าง function ที่มีชื่อ function เหมือนกัน แต่ parameter ต่างกัน
- ประโยชน์เพื่อให้เราสามารถเขียนโปรแกรมให้ยืดหยุ่นมากขึ้นได้ หากใช้ parameter น้อย ก็ยังสามารถรันได้ ยกตัวอย่าง เช่น Console.WriteLine() เป็นต้น

Override Method

- การทำ Method ชื่อเหมือนกัน และ parameter เหมือนกัน เพียงแต่ว่า อยู่คนละ class กัน และ class นั้นต้องมีความสัมพันธ์เชิงแม่ลูก เช่น ToString ใน class ต่างๆ เป็นต้น