การเขียนโปรแกรมเชิง Object

Suphot Sawattiwong
tohpus@gmail.com

การเขียนโปรแกรมเชิงobject

- การเขียนโปรแกรมเชิงobject คือการสร้างระบบให้กับ source code ของโปรแกรม โดยแบ่งแยกหน้าที่และงานออกไปเป็นหมวดหมู่ หรือกลุ่มของตัวแปรและfunctionที่มีความสัมพันธ์เกี่ยวโยงกัน โดยเรียกกลุ่มเหล่านี้ว่า “Object”
- การทำงานของภายใน object เอง ต้องไม่กระทบกับ object อื่นๆ
- การขยายความสามารถของโปรแกรมก็เป็นเพียงเพิ่มความสามารถของ object หรือ เพิ่มจำนวน object เข้าไปในระบบเท่านั้น
- การเขียนCode จากการเขียนแบบ Sequence พอเปลี่ยนมาเขียนเชิงobject ทำให้จำนวน code น้อยลงอย่างมาก

Object

- การเขียนโปรแกรมเชิงobject คือการจัดโครงสร้างของข้อมูลที่ใช้ในการเขียนโปรแกรม ให้มีลักษณะเหมือนวัตถุในโลกความเป็นจริง
- Object ในโปรแกรมก็เหมือนกับสิ่งต่างๆในชีวิตประจำวัน ไม่ว่าจะเป็น หนังสือ, บ้าน, รถยนต์, โต๊ะ แม้กระทั่งตัวของคุณเองก็เป็น object เป็นต้น
- โดย Object ทุกตัวต้องมีต้นแบบ ซึ่งต้นแบบเหล่านี้เรียกว่า Class ในโลกของความจริง ต้นแบบหรือ Class ของ Object เช่น ต้นแบบของหนังสือ ก็คือเพลตที่ใช้พิมพ์, แปลนบ้าน, ผังการออกแบบรถยนต์ เป็นต้น

Object(ต่อ)

- แบบแปลนบ้าน คือ Class และบ้านที่ถูกสร้างจากแปลนบ้านนั้น จะถูกเรียกอีกอย่างว่า instance
- ถ้าจะบอกว่า “คน” คือ Class
- ทุกคนบนโลก ก็จะเป็น instance
- ตัว object จะไม่สามารถทำอะไรได้เลย หากเราไม่มีการสร้าง instance ของ class นั้นๆ เหมือนกับเรามีแปลนบ้าน แต่ไม่มีบ้านจริงๆ

Object ประกอบด้วย

- คุณลักษณะ
- พฤติกรรม หรือ การใช้งาน

Property

- คุณลักษณะ หรือ Property หรือ Attribute หรือ field ซึ่งเป็นความหมายเหมือนกัน คือคุณลักษณะ เช่น ยี่ห้อ(Nokia, iPhone, Samsung), รุ่น(N9, iPhone 4, SII), สี(ดำ, ขาว)
- ยิ่ง object มีขนาดใหญ่ จะมีความซับซ้อนของคุณลักษณะ ยิ่งมากตามไปด้วย
- โดยคุณลักษณะ จะเก็บค่าที่ไว้ใช้ใน object นั้นๆ
- ดังนั้นเวลาตั้งชื่อจึงควรใช้ คำนาม(noun) ในการตั้งชื่อ

Method

- พฤติกรรม หรือ Method หรือ Behavior หรือ Operation หรือ Member Function หรือการใช้งาน
- เช่นหากเป็นโทรศัพท์สามารถโทรออกได้, ส่ง sms ได้, สามารถเก็บเบอร์โทรได้
- ยิ่งobject มีความซับซ้อนมาก ก็ยิ่งมี Method มากขึ้นไปด้วยเช่นกัน

ข้อดีของการเขียนโปรแกรมเชิง object

- เนื่องจากการเขียนโปรแกรมเชิง object ใช้การออกแบบเป็นกลุ่ม object และ การทำงานภายใน object ต้องไม่กระทบการทำงาน object อื่นๆ ทำให้ object แต่ละอันสามารถแยกกันเป็นอิสระออกจากกันได้ ทำให้นำไปใช้ในโปรแกรมอื่นๆได้ เรียกว่า การ reusable
- ลดจำนวนความซับซ้อนของการเขียนโปรแกรมลง
- ทำให้แบ่งหน้าที่การทำงานอย่างชัดเจน

Object กับ C#

- ภาษา C# กับ มีโครงสร้างของ object ซึ่งสามารถสร้าง class ของ object ขึ้นมาได้เอง หรืออาจใช้งาน class ที่ .Net Framework ได้เตรียมเอาไว้ เรียกว่า Base Class ก็ได้
- การเขียนโปรแกรมเชิงobject จะต้องประกอกด้วยขึ้นตอนดังนี้
  - การสร้าง class ต้นแบบของobject
  - การสร้าง instance ของ classต้นแบบ
  - เรียกใช้ instance ที่สร้างขึ้นมา

สร้าง class ต้นแบบของ object

- โครงสร้างของ class เป็นดังนี้

```
class <ชื่อ Class>
{
}
```

- ภายใน block ของ class จะเป็นที่เก็บ method และ attribute ของ class เอาไว้ และ ไม่มี attribute ใด หรือ method ใดอยู่นอก class ได้

สร้าง class ต้นแบบของ object (ต่อ)

- Attribute ของ class คือตัวแปรใน class
- Method ของ class ก็คือ function ใน class

ตัวอย่าง Class Person

```
using System;
namespace ClassExample
{
    class Person
  {
        // Attribute
        public string name, surname;
        public bool gender;
        public int age;
        public string nationality;

        // Method
        public void Walk()
    {
            Console.WriteLine("Walk...");
    }
  }
}
```

สร้าง instance จาก Class ต้นแบบ

- สามารถสร้าง instance ได้ดังนี้

```
<ชื่อ Class> <ชื่อตัวแปร> = new <ชื่อ Class>();
```

```
Person me = new Person(); // สร้างinstance
```

การเรียกใช้ instance ที่สร้างขึ้นมา

- เวลาเรียกใช้ instance ได้โดยต้องระบุชื่อ instance ตามด้วยเครื่องหมายจุด (.) และส่วนประกอบใน instance (Attribute หรือ Method)

ตัวอย่างการเรียกใช้งาน instance

```
using System;
namespace ClassExample
{
    class Program
  {
        static void Main(string[] args)
    {
            Person me = new Person(); // สร้างinstance
            me.name ="Suphot";
            me.surname = "Sawattiwong";
            me.age= 18;
            me.gender = true;
            me.nationality ="Thai";
            me.Walk();
            Console.ReadLine();
    }
  }
}
```

Constructor

- คือ Method พิเศษ ภายใน class ที่ทำงานเองแบบอัตโนมัติ ตอนที่ instance ถูกสร้างขึ้น และ Method นี้ต้องมีชื่อเดียวกับ Class และเหมือนชื่อ Class ทุกประการ

```
class <ชื่อ Class>
{
        public <ชื่อClass> (<Parameter>)
        {

        }
}
```

ตัวอย่าง Constructor

```
using System;
namespace ClassExample
{
    class TestObject
  {
        public TestObject()
    {
            Console.WriteLine("Constructor Start...");
    }
        public void SomeMethod()
    {
            Console.WriteLine("SomeMethod Start...");
    }
  }
}
```

ตัวอย่าง Constructor (ต่อ)

```
using System;
namespace ClassExample
{
    class Program
  {
        static void Main(string[] args)
    {
            TestObject myObject = new TestObject();
// Construtor ทำงาน
            myObject.SomeMethod(); // Some Method
ทำงาน
            Console.ReadLine();
    }
  }
}
```

Output

```
Constructor Start...
SomeMethod Start...
```

ตัวอย่าง Constructor แบบมี parameter

```
using System;
namespace ClassExample
{
    class TestObject
  {
        public TestObject(int x)
    {
            Console.WriteLine("Constructor Start {0}" ,x);
    }
        public void SomeMethod()
    {
            Console.WriteLine("SomeMethod Start...");
    }
  }
}
```

ตัวอย่าง Constructor (ต่อ)

```
using System;
namespace ClassExample
{
    class Program
  {
        static void Main(string[] args)
    {
            TestObject myObject = new TestObject(7);
// Construtor ทำงาน
            myObject.SomeMethod(); // Some Method
ทำงาน
            Console.ReadLine();
    }
  }
}
```

Output

```
Constructor Start 7
SomeMethod Start...
```

Destructor

- เมื่อสร้าง instance ขึ้นมา จะมีการทำงานในส่วนของ constructor ในทางทางกลับกันก่อน instance จะถูกลบออกจากหน่วยความจำ จะทำการ Destructor
- ซึ่งเป็นการควบคุม Garbage Collector
- ส่วนใหญ่เป็นการคืนค่าในหน่วยความจำที่จองไว้

```
class <ชื่อ Class>
{
        ~<ชื่อClass>()
        {

        }
}
```

Constructor & Destructor

- การเขียน Constructor และ Destructor จะเขียนหรือไม่เขียนก็ได้ ต่อให้ไม่มี Constructor ตัว instance ก็ถูกสร้างขึ้นอยู่ดี

การใช้ this แทน Class ตัวเอง

- this เป็นการเรียกแทน class ที่ตัวเองอยู่
- ช่วยให้แยกตัวแปรที่เป็น local กับ Attribute ได้

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