

การเขียนโปรแกรม ด้วยภาษา C#

Class Inheritance – Part 1

Class Inheritance

Class Inheritance

- Class Inheritance ช่วยให้เราสามารถขยายขีดความสามารถของคลาสที่สร้างไว้
 - คลาสต้นฉบับเรียกว่า base class
 - คลาสที่สืบทอดเรียกว่า derived class
- สมาชิกของ derived class ประกอบด้วย
 - สมาชิกของ base class
 - สมาชิกที่สร้างขึ้นภายใน derived class
- Derived class ถูกมองว่าเป็นส่วนขยาย (extend) ของ base class
 - Derived class ไม่สามารถลบสมาชิกของ base class ออกจากคลาสตนเองได้

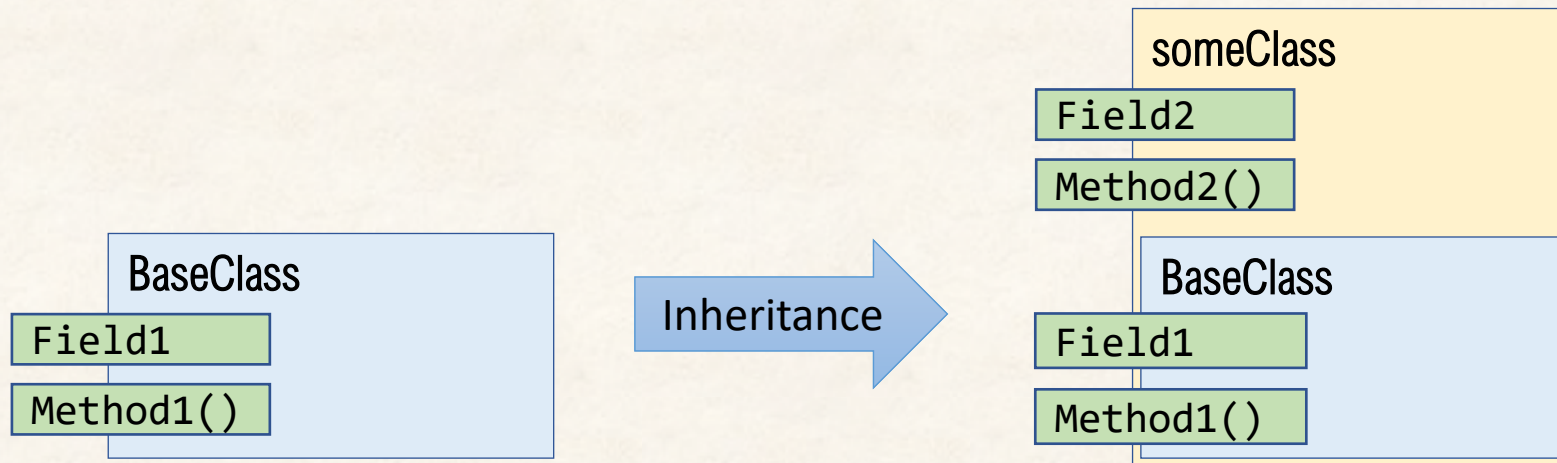
รูปแบบการเขียน Class Inheritance

Class declaration

```
class someClass : BaseClass
{
}
```

Class-base specification

Colon

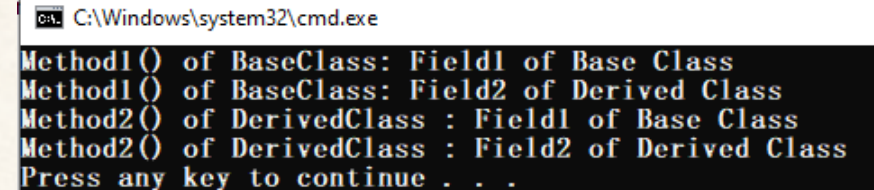


ตัวอย่าง Class Inheritance

```
class BaseClass
{
    public string Field1 = "Field1 of Base Class";
    public void Method1(string value)
    {
        Console.WriteLine($"Method1() of BaseClass: {value}");
    }
}

class DerivedClass : BaseClass
{
    public string Field2 = "Field2 of Derived Class";
    public void Method2(string value)
    {
        Console.WriteLine($"Method2() of DerivedClass : {value}");
    }
}
```

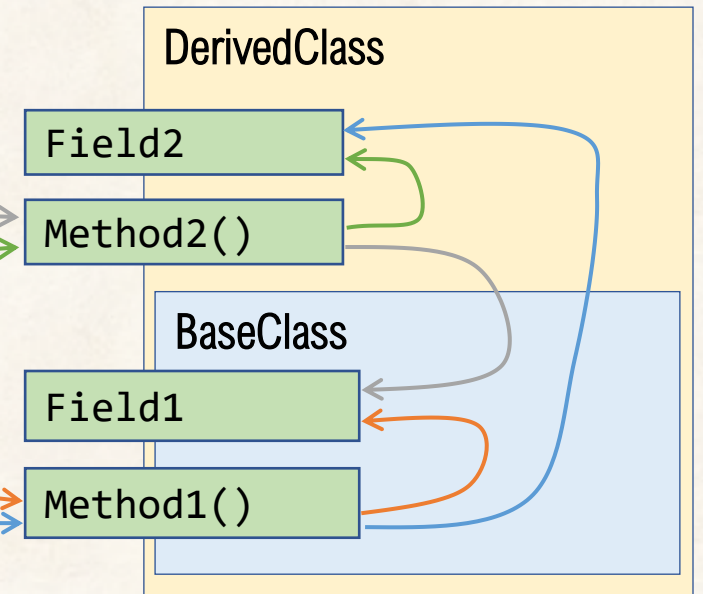
```
class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        DC.Method1(DC.Field1);
        DC.Method1(DC.Field2);
        DC.Method2(DC.Field1);
        DC.Method2(DC.Field2);
    }
}
```



```
C:\Windows\system32\cmd.exe
Method1() of BaseClass: Field1 of Base Class
Method1() of BaseClass: Field2 of Derived Class
Method2() of DerivedClass : Field1 of Base Class
Method2() of DerivedClass : Field2 of Derived Class
Press any key to continue . . .
```

การเข้าถึงสมาชิกที่ derived จาก base Class

```
class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        DC.Method1(DC.Field1);
        DC.Method1(DC.Field2);
        DC.Method2(DC.Field1);
        DC.Method2(DC.Field2);
    }
}
```



ทุกคลาสใน C# ล้วน Inherit จากคลาส Object

- Class ทั้งหมดในภาษา C# ล้วนเป็น derived class
 - ยกเว้นคลาส Object ที่เป็น base class ของคลาสทั้งหมด
 - คลาสที่ไม่ประกาศ base class จะถูกคอมไพเลอร์กำหนดให้ Object เป็น base class

```
class MyClass  
{  
  
}
```

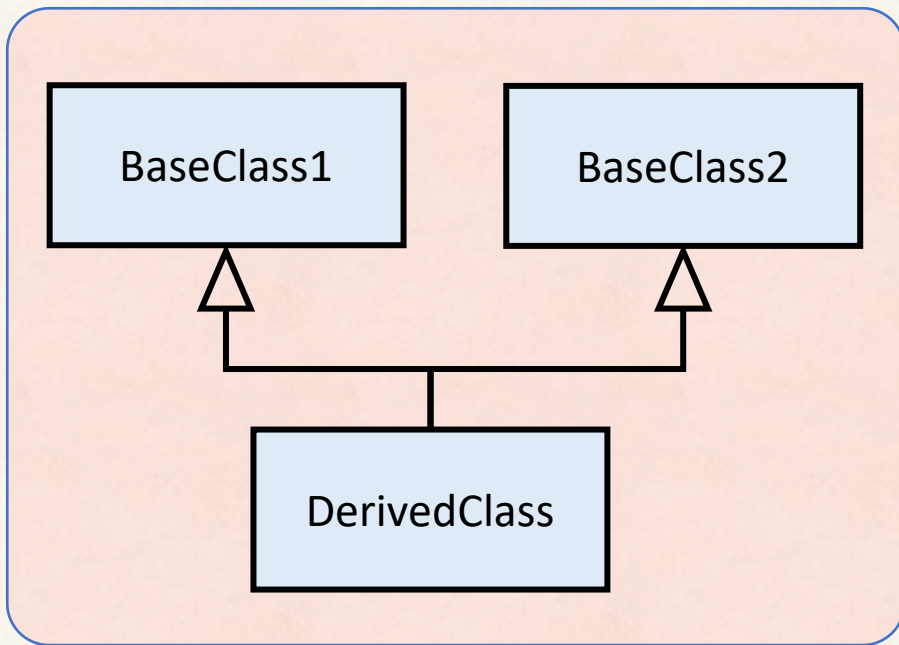
Implicitly derived จากคลาส Object

```
class MyClass : Object  
{  
  
}
```

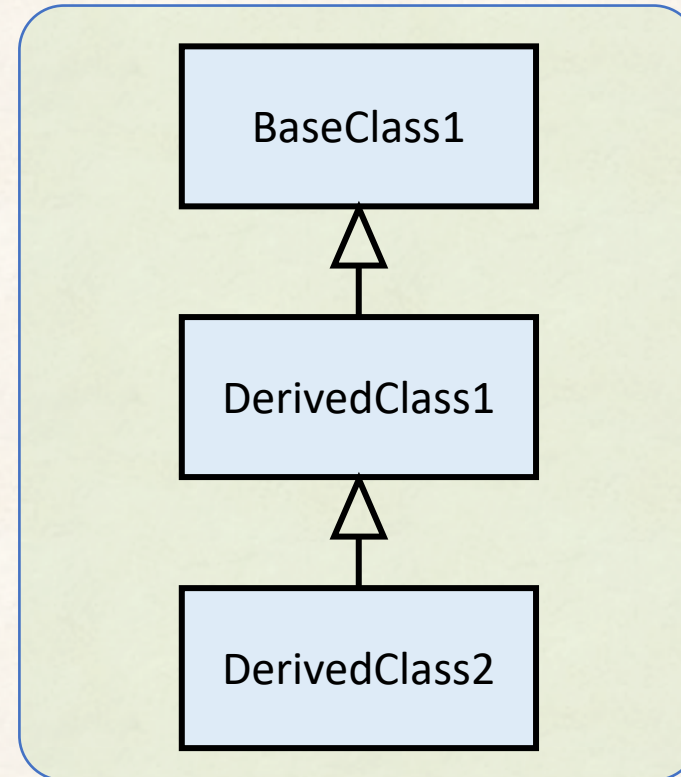
Explicitly derived จากคลาส Object

เงื่อนไขในการ inherited

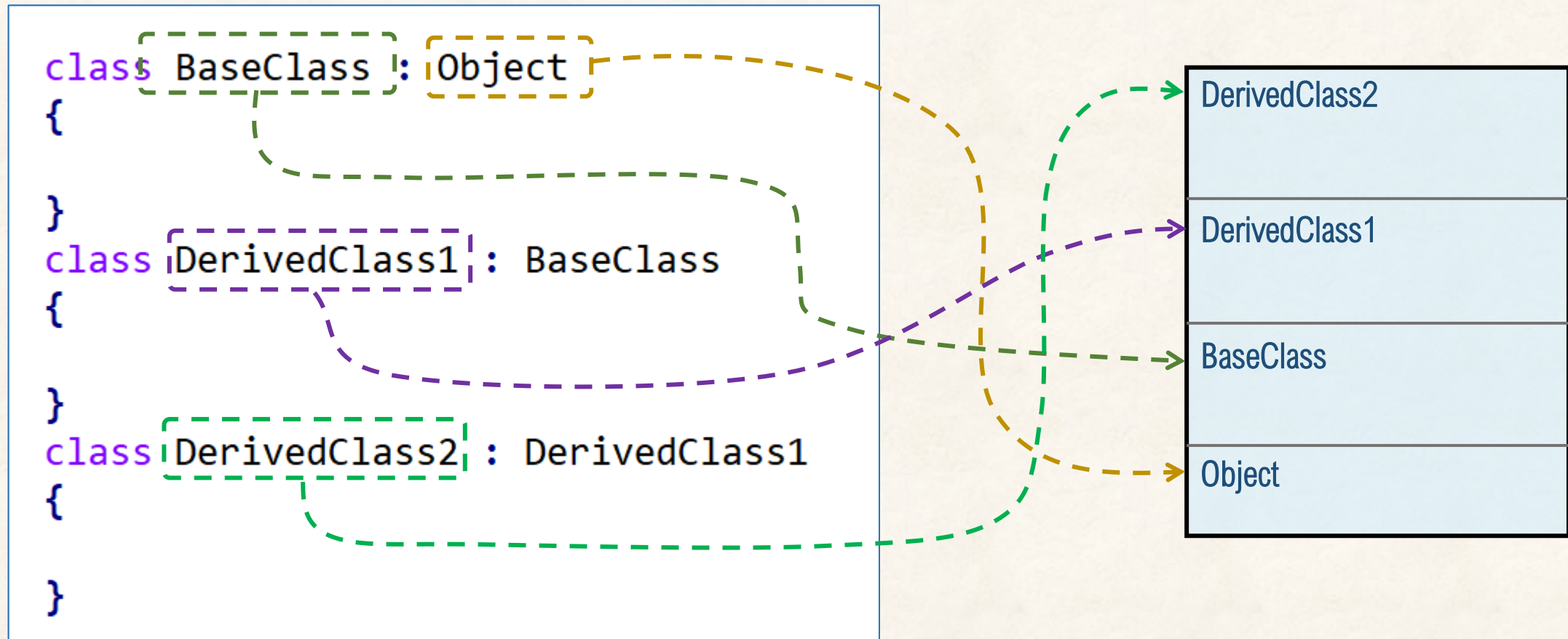
- Class สามารถ inherited จาก base class ได้เพียงคลาสเดียวเท่านั้น
- Class สามารถ inherited จาก base class ในจำนวนลำดับชั้นที่ไม่จำกัด



Error ไม่สามารถ derived จากหลาย base class



การ inherited จาก base class หลายชั้น



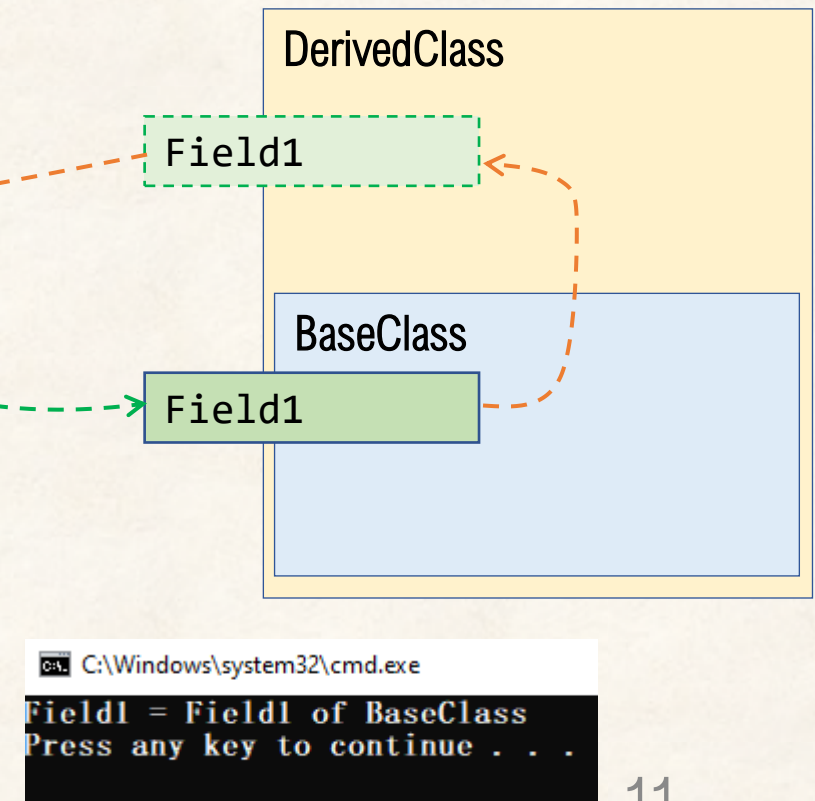
การปิดบัง (Masking) สมาชิกของ Base Class

- Derived class ไม่สามารถกลับสมาชิกของ base class ออกจากกระบวนการสืบทอด
 - แต่สามารถแทนที่ด้วยสมาชิกของตัวเองที่มีชื่อเดียวสมาชิกของ base class ได้
 - เช่น ต้องการเปลี่ยนรายละเอียดการทำงานของ method ที่สืบทอดมา
- การ masking ของ derived class
 - ปิดบัง data member โดยการสร้าง data member ที่มีชื่อและชนิดเดียวกัน
 - ปิดบัง function member โดยการสร้าง function member ที่มีชื่อและพารามิเตอร์แบบเดียวกันทั้งหมด (ไม่นับรวมชนิดการ return)
 - ปิดบังโดยใช้ modifier 'new'
 - สามารถปิดบัง static member ได้

การเข้าถึงสมาชิกของ Base Class โดยปกติ

```
class BaseClass
{
    public string Field1 = "Field1 of BaseClass";
}
class DerivedClass : BaseClass
{
}

class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        Console.WriteLine($"Field1 = {DC.Field1}");
    }
}
```

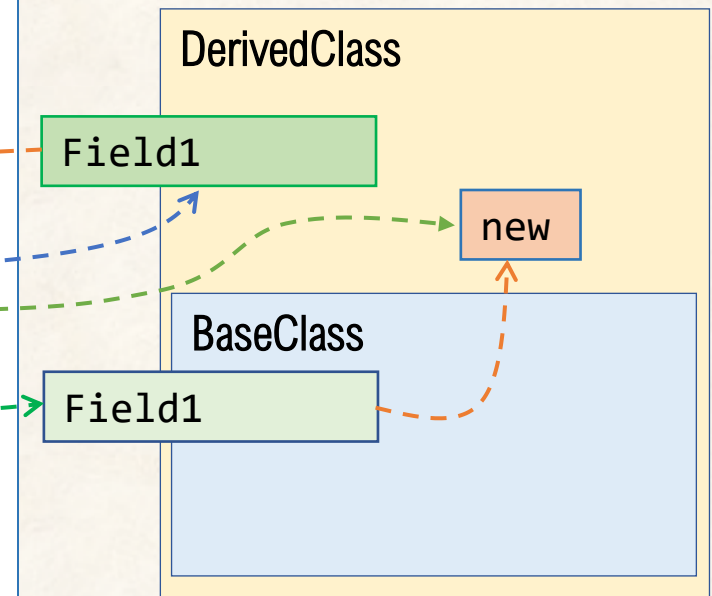


การปิดบัง (Masking) สมาชิกของ Base Class

```
class BaseClass
{
    public string Field1 = "Field1 of BaseClass";
}

class DerivedClass : BaseClass
{
    new public string Field1 = "Field1 of DerivedClass";
}

class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        Console.WriteLine($"Field1 = {DC.Field1}");
    }
}
```



```
C:\Windows\system32\cmd.exe
Field1 = Field1 of DerivedClass
Press any key to continue . . .
```

การเข้าถึงสมาชิกของ Base Class ที่ถูก masking

- ในกรณีที่ derived class ได้ทำการ masking สมาชิกของ base class ไปแล้ว แต่ต้องการเข้าถึงสมาชิกนั้น
 - สามารถทำได้โดยใช้ base access expression

```
Type Variable = { base.Field;
```

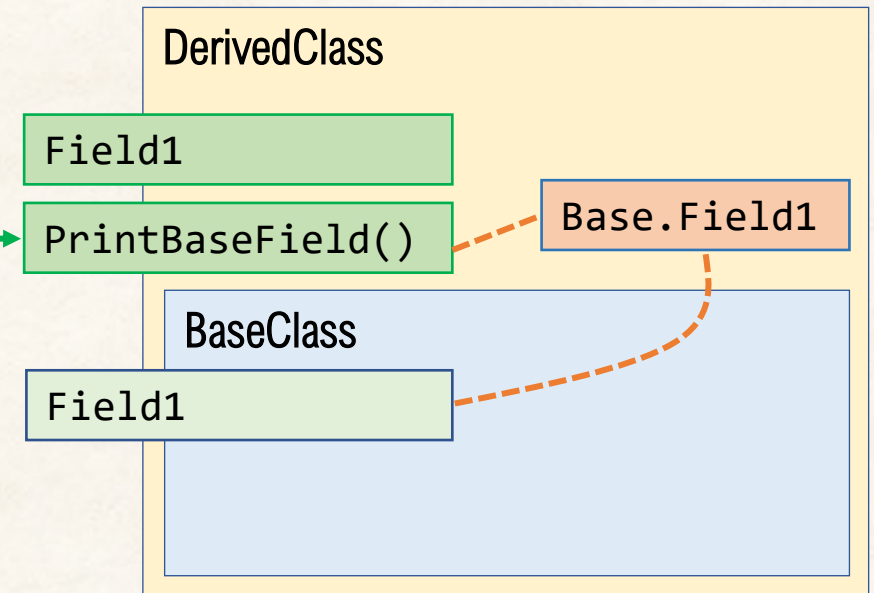
```
base.Field = Variable;
```

```
base.Method();
```

Base access expression

การเข้าถึงสมาชิกของ Base Class ที่ถูก masking

```
class BaseClass
{
    public string Field1 = "Field1 of BaseClass";
}
class DerivedClass : BaseClass
{
    new public string Field1 = "Field1 of DerivedClass";
    public void PrintBaseField1()
    {
        Console.WriteLine($"Field1 = {base.Field1}");
    }
}
```



การเข้าถึงสมาชิกของ Base Class ที่ถูก masking

C:\Windows\system32\cmd.exe

```
Field1 = Field1 of DerivedClass  
Field1 = Field1 of BaseClass  
Press any key to continue . . .
```

```
class Program  
{  
    static void Main()  
    {  
        DerivedClass DC = new DerivedClass();  
        Console.WriteLine($"Field1 = {DC.Field1}");  
        DC.PrintBaseField1();  
    }  
}
```

```
class BaseClass  
{  
    public string Field1 = "Field1 of BaseClass";  
}  
class DerivedClass : BaseClass  
{  
    new public string Field1 = "Field1 of DerivedClass";  
    public void PrintBaseField1()  
    {  
        Console.WriteLine($"Field1 = {base.Field1}");  
    }  
}
```

ข้อสังเกต ถ้าต้องใช้วิธีนี้บ่อยๆ แสดงว่าออกแบบระบบ
คลาสได้ไม่ดี ให้พิจารณาออกแบบระบบการสืบทอด
คลาสใหม่อีกครั้ง

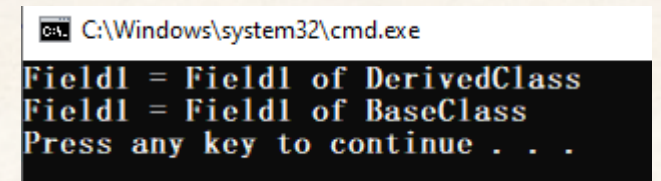
Using References to a Base Class

- สมาชิกทั้งหมดใน derived class คือสมาชิกที่สืบทอดมาจาก base class และสมาชิกที่สร้างขึ้นเอง
 - Reference ที่อ้างถึง instance ของ derived class จะรวมสมาชิกทั้งก่อน
 - ใน instance ของ derived class เราสามารถหา reference ไปยังส่วนที่เป็น base class ได้
- เราสามารถหา reference ไปยังส่วนที่เป็น base class ด้วย cast operator
 - วาง cast operator ไว้หน้าชื่อ instance ของ derive class
 - ใส่ชื่อของ base class ไว้ในวงเล็บ เพื่อบอกคอมไพเลอร์รู้ว่าต้องการอ้างถึง base class ตัวไหน
 - ในกรณีมีการสืบทอดหลายชั้น ก็สามารถเลือก base class ในชั้นที่ต้องการได้

Using References to a Base Class

```
class BaseClass
{
    public string Field1 = "Field1 of BaseClass";
}
class DerivedClass : BaseClass
{
    new public string Field1 = "Field1 of DerivedClass";
}
```

```
class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        Console.WriteLine($"Field1 = {DC.Field1}");
        BaseClass BC = (BaseClass)DC;
        Console.WriteLine($"Field1 = {BC.Field1}");
    }
}
```

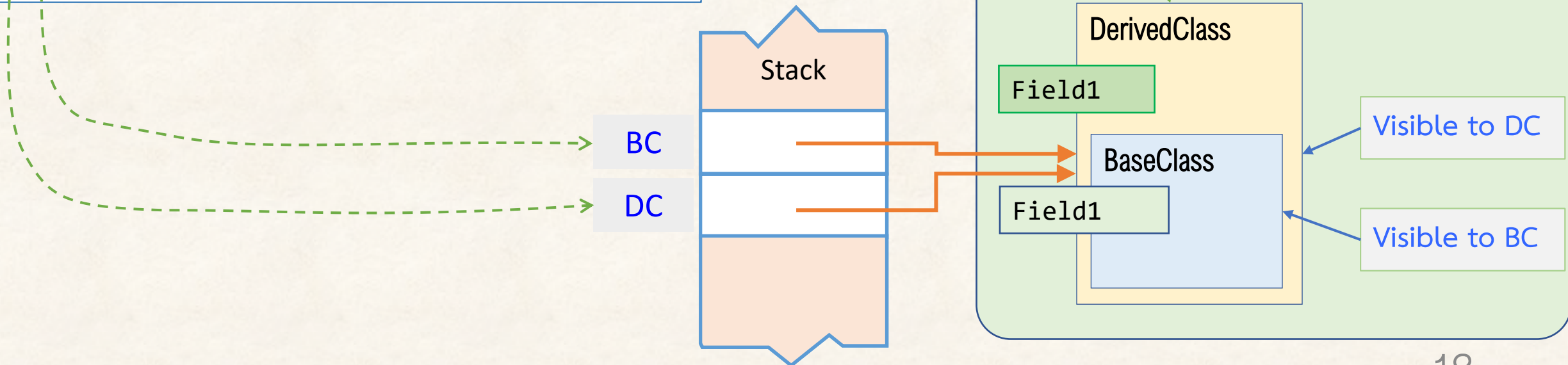


C:\Windows\system32\cmd.exe

```
Field1 = Field1 of DerivedClass
Field1 = Field1 of BaseClass
Press any key to continue . . .
```

Using References to a Base Class

```
class Program
{
    static void Main()
    {
        DerivedClass DC = new DerivedClass();
        Console.WriteLine($"Field1 = {DC.Field1}");
        BaseClass BC = (BaseClass)DC;
        Console.WriteLine($"Field1 = {BC.Field1}");
    }
}
```



Virtual and Override Methods

Next