

การเขียนโปรแกรม ด้วยภาษา C#

Members ชนิดอื่น ๆ ของ Class

ส่วนประกอบของ Class

Data members

- ✓ Fields
- ✓ Constants

Function Members

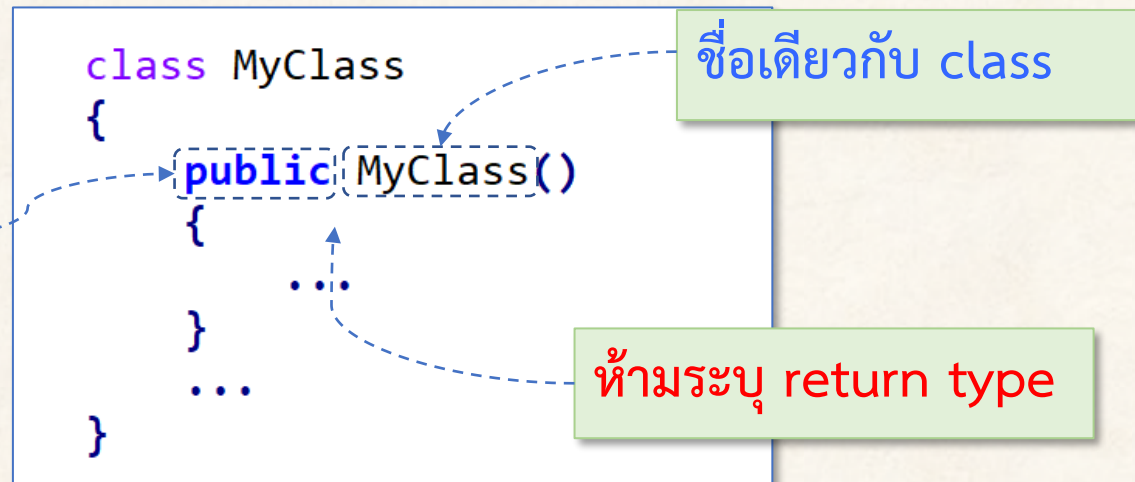
- ✓ Methods
- ✓ Properties
- Constructors
- Destructors
- Operators
- Indexers
- Events

Constructors

Instance constructors

- Instance constructors เป็น method ที่มีชื่อเดียวกับคลาส
 - ถูกเรียกใช้ทันทีเมื่อสร้าง instance ใหม่
 - เป็นที่ที่เหมาะสมสำหรับการ initial ค่า state ต่างๆ ให้แก่ instance
 - ถ้าต้องการให้สร้าง instance จากภายนอกคลาส ให้กำหนด modifier เป็น public
 - constructor ไม่สามารถส่งค่ากลับ (ไม่ต้องใส่ชนิด return แม้กระทั่ง void)

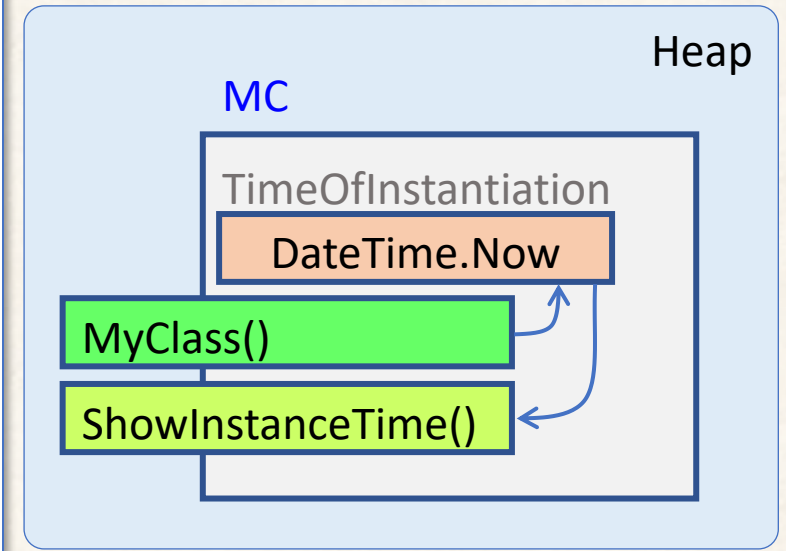
ระบุเป็น public
สามารถสร้าง instance ได้จาก
ภายนอก MyClass



ตัวอย่างการใช้ Instance constructors

```
class MyClass
{
    DateTime TimeOfInstantiation;
    public MyClass()
    {
        TimeOfInstantiation = DateTime.Now;
    }
    public void ShowInstanceTime()
    {
        Console.WriteLine($"Created at {TimeOfInstantiation}");
    }
}

class Program
{
    static void Main()
    {
        MyClass MC = new MyClass();
        MC.ShowInstanceTime();
    }
}
```



Constructors with Parameters

- Constructors มีฐานะเป็น method สามารถรับ parameters ได้เช่นเดียวกับ method ทั่วไป
 - ใช้รูปแบบของพารามิเตอร์เช่นเดียวกับ method อื่น ๆ
 - สามารถทำ overloaded ได้
- ในขณะที่สร้าง instance คอมไพเลอร์จะเลือก constructor ที่มี signature ตรงกับ constructor ตัวหนึ่งขึ้นมาทำงาน

Constructors with Parameters

```
class MyCar
{
    static int id;
    string NamePlate;
    public MyCar()
    {
        id++;
        NamePlate = $"Mycar-{id:d4}";
    }
    public MyCar(string nameplate)
    {
        id++;
        this.NamePlate = nameplate;
    }
    public void ShowCarDetails()
    {
        Console.WriteLine($"ID {id}, Name: {NamePlate}");
    }
}
```

```
class Program
{
    static void Main()
    {
        MyCar mC1 = new MyCar();
        mC1.ShowCarDetails();
        MyCar mC2 = new MyCar();
        mC2.ShowCarDetails();
        MyCar mC3 = new MyCar("ABC-1234");
        mC3.ShowCarDetails();
    }
}
```

```
C:\Windows\system32\cmd.exe
ID 1, Name: Mycar-0001
ID 2, Name: Mycar-0002
ID 3, Name: ABC-1234
Press any key to continue . . .
```

Default Constructors

- ถ้าไม่ได้กำหนด Constructor ใด ๆ ในคลาสเลย Compiler จะสร้าง default constructor ให้โดยอัตโนมัติ
 - เป็น Constructor ที่ไม่มี parameters
 - เป็น Constructor ที่ไม่มี คำสั่งใด ๆ ใน method body
- ถ้ามีการประกาศ constructor แม้เพียง 1 ตัว Compiler จะไม่สร้าง default constructor ให้
 - ข้อควรระวัง.. ถ้าสร้าง instance ด้วยรายการ parameter ที่ว่างเปล่า อาจทำให้เกิด error ได้

Default Constructors

error CS7036: There is no argument given that corresponds to the required formal parameter 'nameplate' of 'MyCar.MyCar(string)'

```
class MyCar
{
    static int id;
    string NamePlate;
    public MyCar(string nameplate)
    {
        id++;
        this.NamePlate = nameplate;
    }
    public void ShowCarDetails()
    {
        Console.WriteLine($"ID {id}, Name: {NamePlate}");
    }
}
```

```
class Program
{
    static void Main()
    {
        MyCar mC1 = new MyCar();
        mC1.ShowCarDetails();
        MyCar mC2 = new MyCar("ABC-1234");
        mC2.ShowCarDetails();
    }
}
```

Static Constructors

- Constructor สามารถประกาศเป็นแบบ static ได้เช่นเดียวกับ method ทั่วไป
 - instance constructors ใช้สำหรับ initial ค่าในระดับ instance
 - static constructor ใช้สำหรับ initial ค่าในระดับ class
 - โดยปกติ static constructor มีหน้าที่หลักคือการ initial ค่าให้ static fields ของคลาส
 - static constructor ไม่สามารถเข้าถึง ส่วนประกอบใน instance ของคลาสได้
- constructor ระดับคลาสจะถูกเรียกทำงานเป็นลำดับแรก
 - ทำงานก่อนที่ instance ใด ๆ ของคลาสจะถูกสร้างขึ้น
 - ทำงานก่อนการเข้าถึง static member อื่น ๆ ทั้งหมด

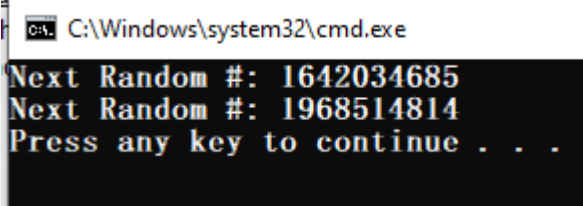
Static Constructors

- Static constructor มีลักษณะเหมือนกับ constructor อื่น ๆ
 - ต้องมีชื่อเดียวกันกับชื่อคลาส
 - ต้องไม่มีการ return ค่าใด ๆ กลับจาก static constructor
- ส่วนที่ Static constructor มีลักษณะต่างจาก constructor อื่น ๆ
 - การประกาศ static constructor จะใช้คีย์เวิร์ด static
 - คลาสสามารถมี static constructor ได้เพียงแค่ตัวเดียว (และต้องไม่มี parameters)
 - ไม่สามารถมี access modifier (เช่น public, private, ...)

Static Constructors

```
class RandomNumber
{
    private static Random RandomKey;
    static RandomNumber()
    {
        RandomKey = new Random();
    }
    public int GetRandomNumber()
    {
        return RandomKey.Next();
    }
}
```

```
class Program
{
    static void Main()
    {
        RandomNumber a = new RandomNumber();
        RandomNumber b = new RandomNumber();
        Console.WriteLine($"Next Random #: {a.GetRandomNumber()}");
        Console.WriteLine($"Next Random #: {b.GetRandomNumber()}");
    }
}
```



```
C:\Windows\system32\cmd.exe
Next Random #: 1642034685
Next Random #: 1968514814
Press any key to continue . . .
```

Object Initializers

- นอกจากส่งค่าเริ่มต้นผ่านทาง constructors แล้ว เราสามารถส่งค่าให้ instance ของคลาสด้วยวิธีที่เรียกว่า object initializer
 - ทำได้โดยการใส่รายการ parameter ไว้ในวงเล็บปีกกา ต่อท้ายการสร้าง instance ด้วยคำสั่ง new

```
new TypeName = {FieldOrProperty = InitExpression, ... }
```

```
new TypeName(ArgumetesList) = {FieldOrProperty = InitExpression, ... }
```


ตัวอย่าง Object Initializers

```
public class Point
{
    public int X = 0;
    public int Y = 0;
    public Point()
    {
    }
    public Point(int x, int y)
    {
        X = x;
        Y = y;
    }
}
```

```
class Program
{
    static void Main()
    {
        Point point1 = new Point();
        Point point2 = new Point(1,2);
        Point point3 = new Point{ X = 3, Y = 4 };
        Console.WriteLine($"point1 = {point1.X},{point1.Y}");
        Console.WriteLine($"point2 = {point2.X},{point2.Y}");
        Console.WriteLine($"point3 = {point3.X},{point3.Y}");
    }
}
```

```
C:\Windows\system32\cmd.exe
point1 = 0,0
point2 = 1,2
point3 = 3,4
Press any key to continue . . .
```


ส่วนประกอบของ Class

Data members

- ✓ Fields
- ✓ Constants

Function Members

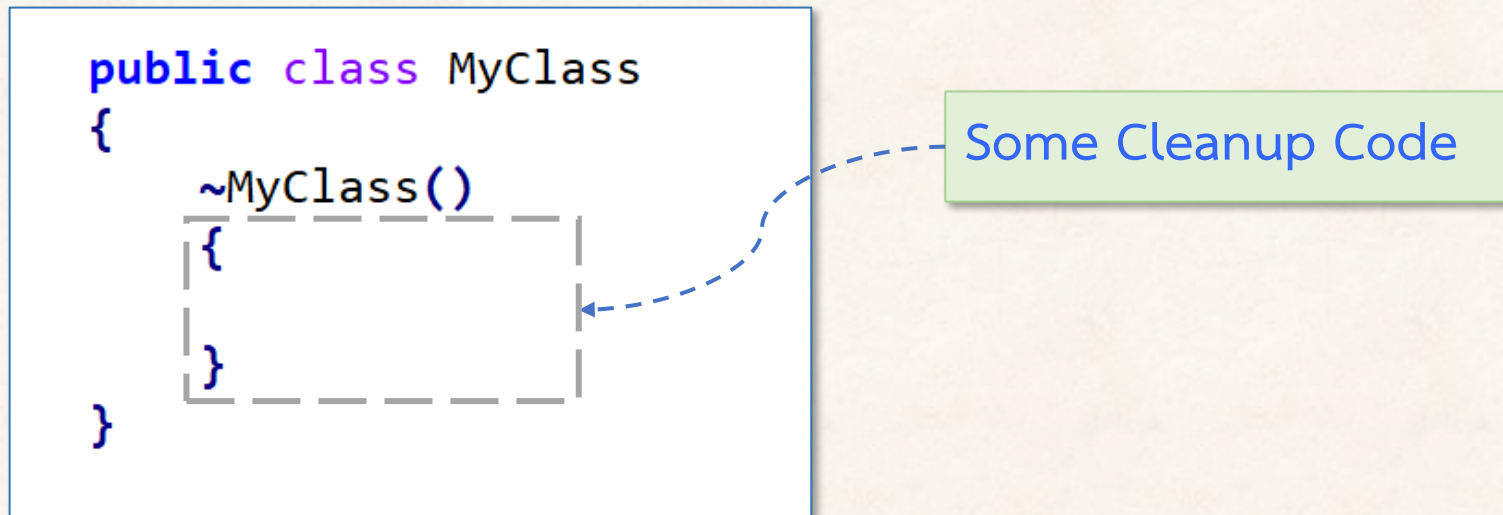
- ✓ Methods
- ✓ Properties
- ✓ Constructors
- Destructors
- Operators
- Indexers
- Events

Destructor

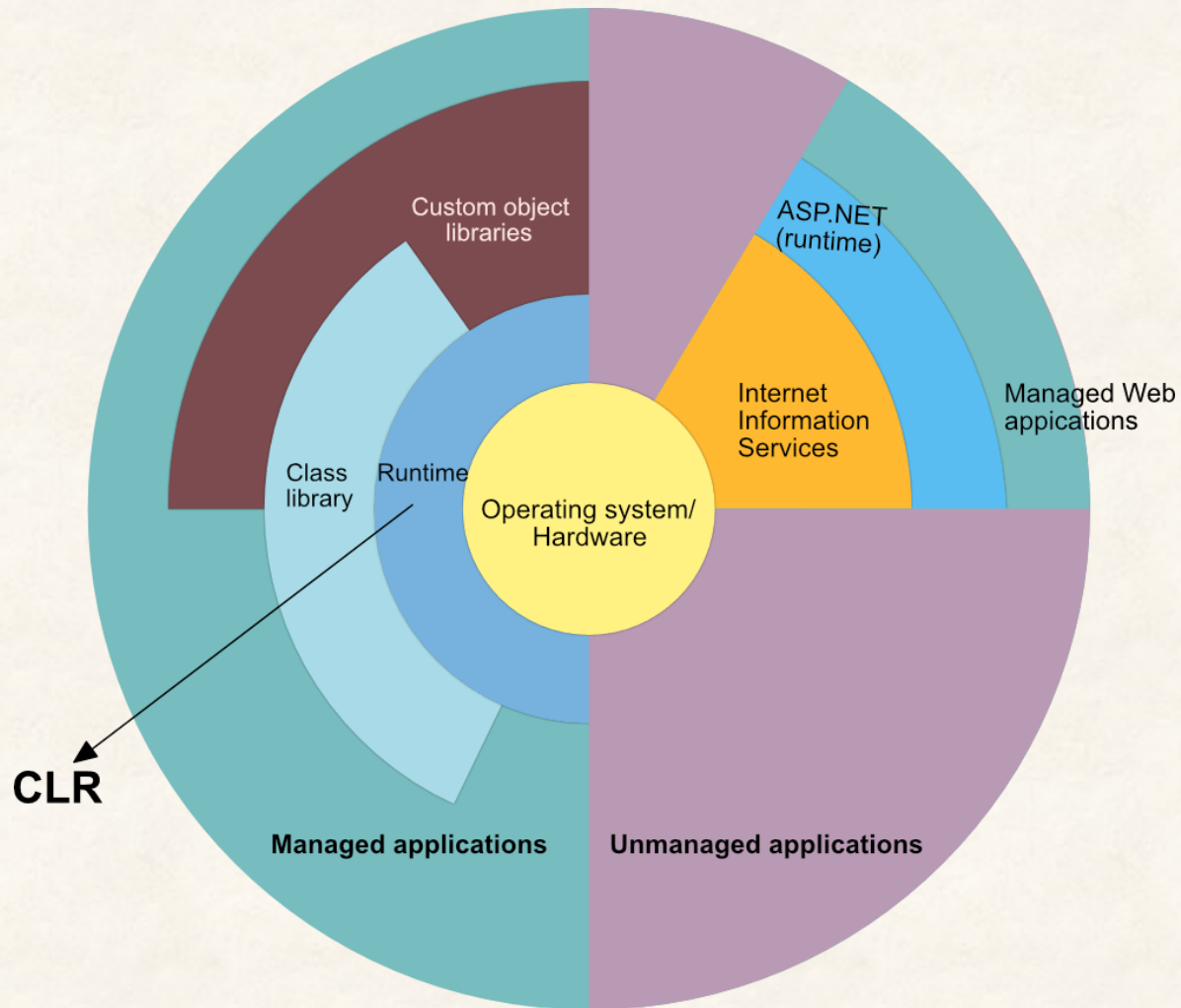
- Destructor จะถูกเรียกมาทำงานเป็น method ท้ายสุด
 - ใช้เพื่อสละสิ่งที้อาจจะหลงเหลืออยู่จากการ initial หรือจองทรัพยากร
 - ในคลาสใด ๆ สามารถมี destructor ได้เพียงตัวเดียว
 - Destructor ไม่มี parameter
 - Destructor ไม่มี accessibility modifier
 - Destructor ทำงานบน instance เท่านั้น ไม่มี static destructor
 - ไม่สามารถเรียก destructor แบบเดียวกับที่เรียก method อื่นๆ
 - Destructor มีชื่อเดียวกับคลาส แต่เพิ่มอักขระขึ้นต้นด้วย ~ (tilde อ่านออกเสียงว่า TIL-duh)

Destructor

- ถ้าไม่จำเป็นก็ไม่ต้องเขียน Destructor
- Destructor ไม่ควรพยายามใช้ทรัพยากรใดๆ (เพราะอาจจะถูกทำลายไปแล้ว)
- ใน C# นั้น Destructor จะถูกเรียกโดย Garbage Collector



Destructor



- ใน managed application นั้น จะมี CLR เป็นตัวกลางระหว่าง application และ Operating System
- การใช้ Destructor ส่วนใหญ่ มักจะใช้ในการคืนทรัพยากรที่จองไว้ และใน CLR จะมีความสามารถในการตรวจสอบและคืนทรัพยากรโดยอัตโนมัติ
- มีทางเลือกที่ดีกว่าคือการใช้งาน `Dispose()`
 - จะเรียนหลังจากเรื่อง Interface

ส่วนประกอบของ Class

Data members

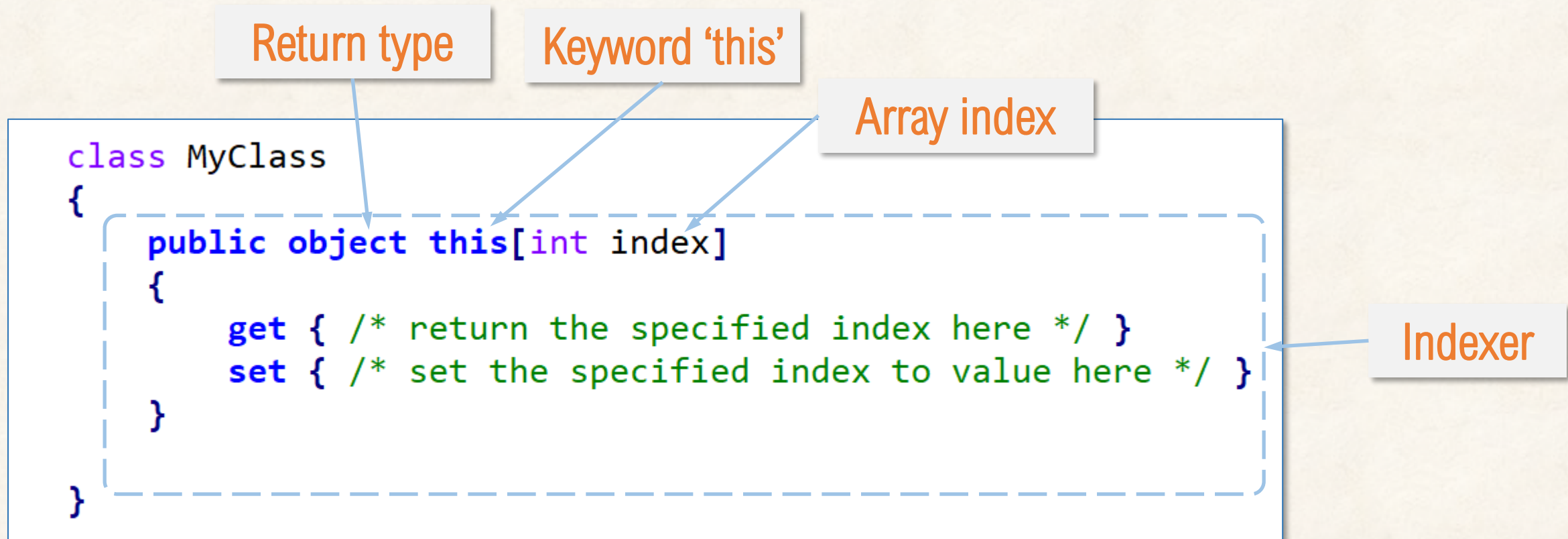
- ✓ Fields
- ✓ Constants

Function Members

- ✓ Methods
- ✓ Properties
- ✓ Constructors
- ✓ Destructors
- Operators
- Indexers
- Events

Indexers

- Indexer มีลักษณะคล้าย properties (มี accessors ชื่อ get และ set)



Indexers vs Properties

- Indexer มีลักษณะคล้าย properties

- Indexer และ properties ไม่มีการจองหน่วยความจำ

- Indexer และ properties มีไว้เพื่อให้สามารถเข้าถึง data member ภายในคลาส

- Indexer ต่างจาก properties

- Property แต่ละตัวสามารถเข้าถึงข้อมูลได้เพียง 1 ตัวเท่านั้น

- Indexer หนึ่งตัวสามารถเข้าถึงข้อมูลได้หลายตัว

อาจจะมองได้ว่า Indexers ก็คือ properties ที่เข้าถึงข้อมูลได้คราวละหลายตัว ด้วยการอ้าง index (แบบเดียวกับ array) โดยที่ index สามารถเป็น type ใดก็ได้

Indexers vs Properties

- Indexer มีลักษณะคล้าย properties
 - Indexer สามารถมี accessor เพียงตัวเดียวหรือทั้งคู่ก็ได้ (get, set)
 - Indexer ต้องเป็น instance member เท่านั้น ไม่สามารถประกาศเป็น static ได้
 - ในส่วนของ code สำหรับ Indexer นั้นสามารถมีคำสั่งใด ๆ ก็ได้
 - ในส่วนของ accessor 'get' ต้องมีคำสั่ง return และส่งกลับค่าใน type ที่ประกาศไว้

การประกาศ Indexer

ต่างจาก property

- ใช้ 'this' แทนชื่อ
- มี parameter ใน []

คล้าย property

- มี return type
- มี accessors (get, set)

```
ReturnType this[Type param1, Type param2, ...]
{
    get
    {
        return ValueInTypeOfReturnType;
    }
    set
    {
    }
}
```

คีย์เวิร์ด 'this'

- คีย์เวิร์ด 'this' ใช้ใน class เมื่อต้องการอ้างถึง instance ที่กำลังใช้งาน
- ใช้ได้เฉพาะในพื้นที่ (block) ต่อไปนี้เท่านั้น
 - instance constructors
 - instance methods
 - instance accessor ของ properties และ indexer
- ไม่สามารถใช้คีย์เวิร์ด 'this' ในส่วนที่เป็น static
- ใช้คีย์เวิร์ด 'this' เพื่อ
 - จำแนกระหว่าง instance member และ local variable ในกรณีที่ต้องตรงกัน
 - ใช้เป็น actual parameter ในขณะที่เรียกใช้ method

คีย์เวิร์ด 'this'

```
class Program
{
    static void Main()
    {
        Circle circle = new Circle();
        circle.SetRadius(10.0);
        Console.WriteLine($"Radius = {circle.GetRadius():N2}");
    }
}
```

```
class Circle
{
    double radius = 0.0d;
    public void SetRadius(double radius)
    {
        this.radius = radius;
    }
    public double GetRadius()
    {
        return radius ;
    }
}
```

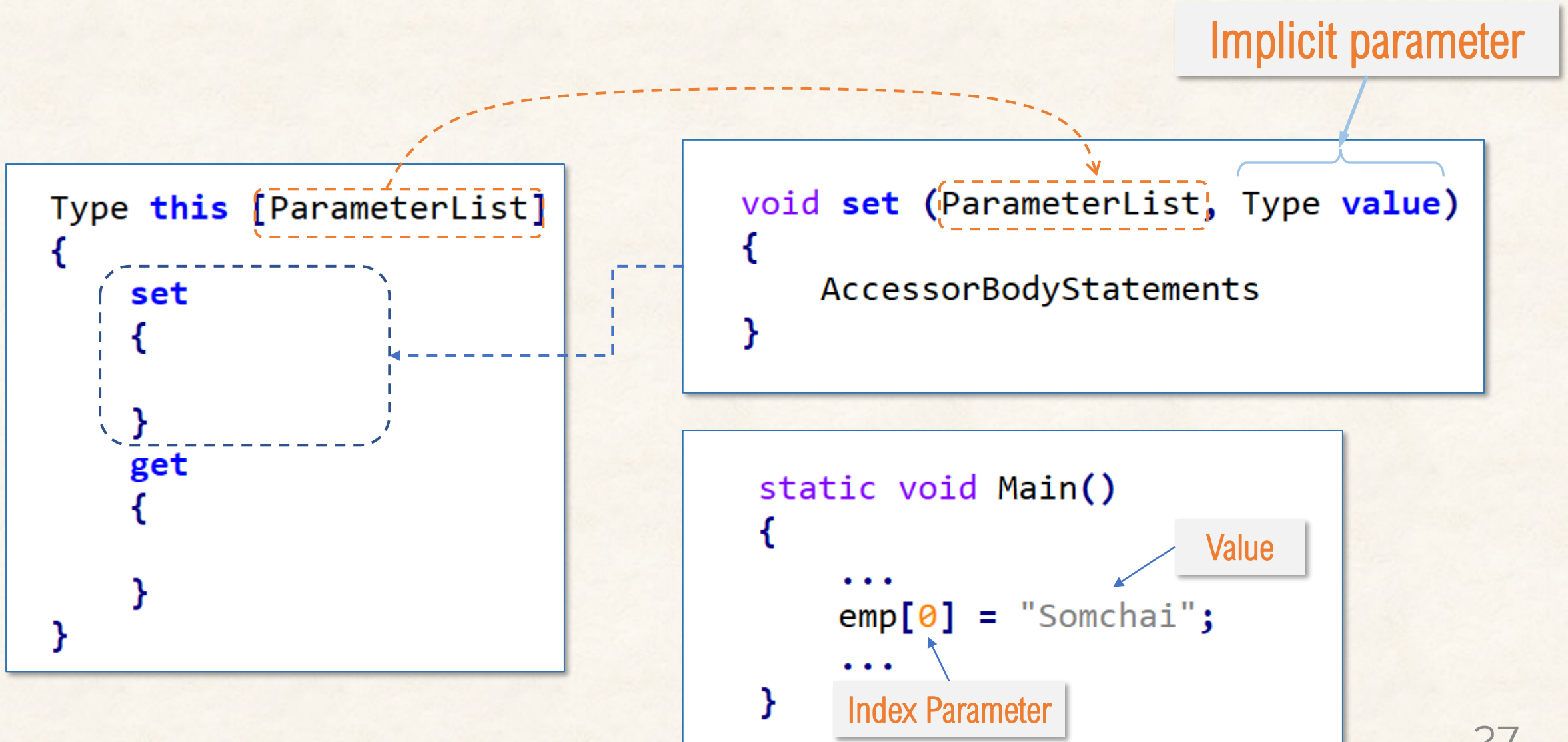
C:\Windows\system32\cmd.exe

```
Radius = 10.00
Press any key to continue . . .
```

The Indexer 'set' Accessor

- เมื่อต้องการกำหนดค่าให้ data member ผ่านทาง Indexers เราต้องรู้ค่าสองอย่าง
 - Parameter (ชื่อตัวแปร) ที่เก็บค่าต้นทาง
 - Index ที่จะนำค่าไปเก็บยังปลายทาง
- set มีชนิด return เป็น void
- มีรายการ parameter เหมือนกับที่ประกาศใน Indexer
- สามารถมี implicit value parameter ชนิดเดียวกับที่ประกาศใน Indexer

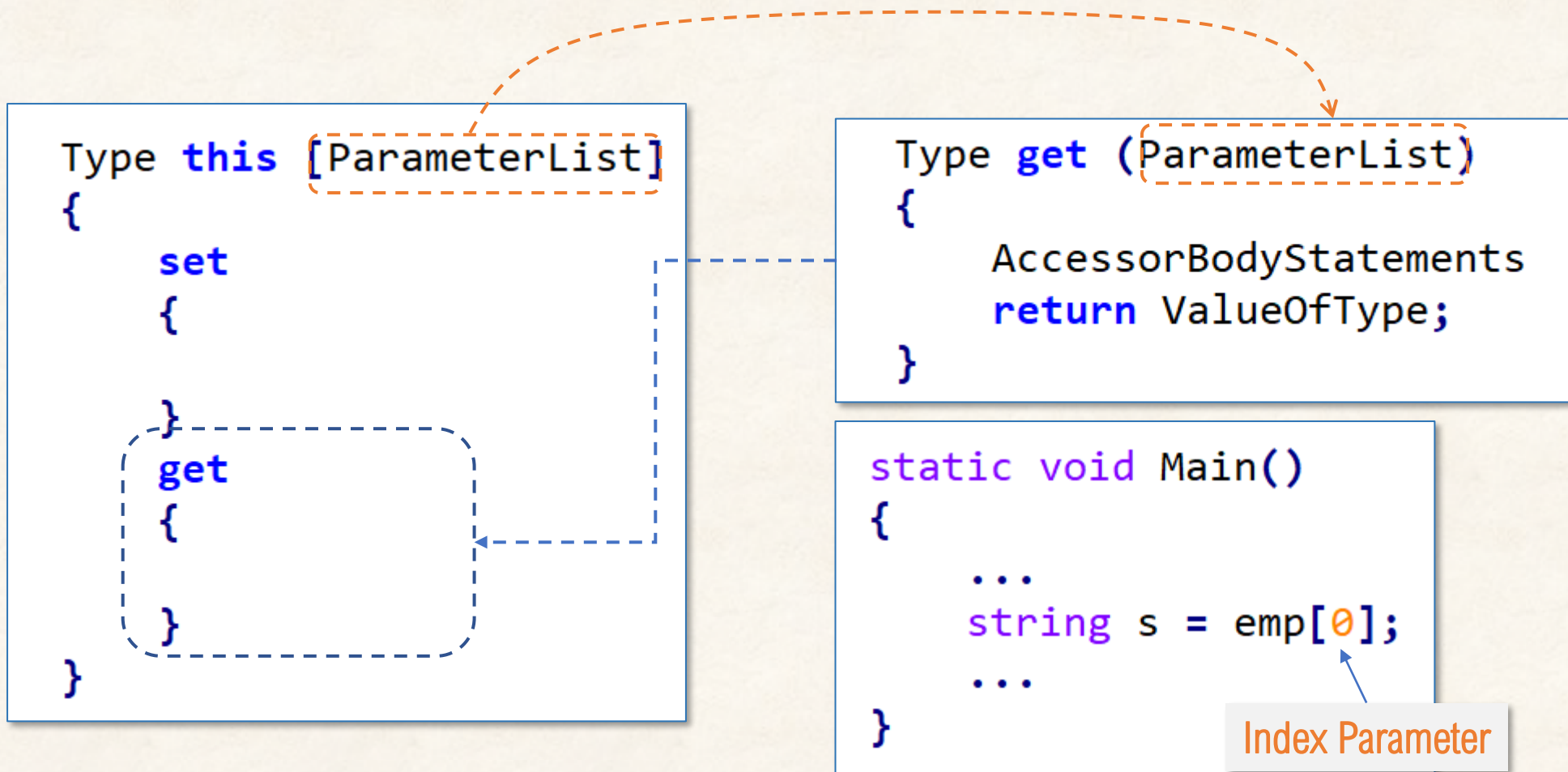
The Indexer 'set' Accessor



The Indexer 'get' Accessor

- เมื่อต้องการอ่านค่าจาก data member ผ่านทาง Indexers เราต้องกำหนด
 - ตัวแปรที่เก็บค่าปลายทาง
 - Index ที่จะอ่านค่าจาก indexer ต้นทาง
- get มีชนิด return ตามที่ระบุไว้ใน type ของ Indexer
- มีรายการ parameter เหมือนกับที่ประกาศใน Indexer

The Indexer 'get' Accessor

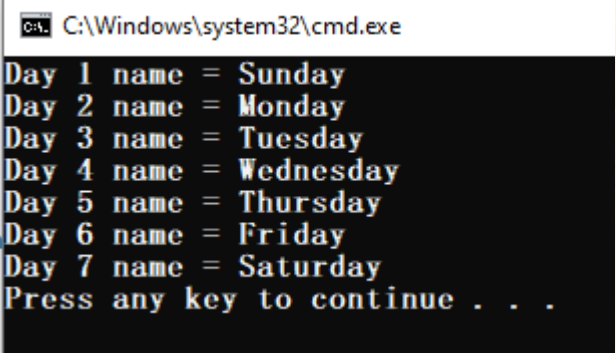


ตัวอย่าง Indexer

```
class Program
{
    static void Main()
    {
        DayOfWeek dow = new DayOfWeek();
        dow[0] = "Sunday";
        dow[1] = "Monday";
        dow[2] = "Tuesday";
        dow[3] = "Wednesday";
        dow[4] = "Thursday";
        dow[5] = "Friday";
        dow[6] = "Saturday";

        for (int i = 0; i < 7; i++)
        {
            Console.WriteLine($"Day {i+1} name = {dow[i]}");
        }
    }
}
```

```
class DayOfWeek
{
    private string[] val = new string[7];
    public string this[int index]
    {
        get { return val[index]; }
        set { val[index] = value; }
    }
}
```



C:\Windows\system32\cmd.exe

```
Day 1 name = Sunday
Day 2 name = Monday
Day 3 name = Tuesday
Day 4 name = Wednesday
Day 5 name = Thursday
Day 6 name = Friday
Day 7 name = Saturday
Press any key to continue . . .
```

Indexer Overloading

- เราสามารถสร้าง Indexer หลายตัวที่มีชื่อเดียวกันได้
 - เรียกว่า Indexer overloading
 - รายการ Parameter list ต้องต่างกัน

```
class MyClass
{
    public string this [ int index ]
    {
        get { ... }
        set { ... }
    }
    public string this [ int index1, int index2 ]
    {
        get { ... }
        set { ... }
    }
    public int this [ float index1 ]
    {
        get { ... }
        set { ... }
    }
    ...
}
```

ส่วนประกอบของ Class

Data members

- ✓ Fields
- ✓ Constants

Function Members

- ✓ Methods
- ✓ Properties
- ✓ Constructors
- ✓ Destructors
- Operators
- ✓ Indexers
- Events

คำถาม?