

📖 2. 객체지향 개발

📄 1. Class & Instance & new

1). new를 통해 Instance 생성

new 연산자로 인스턴스(객체)를 만든다.

- (메모리 heap영역에) 데이터 저장 공간을 할당받고
-> 그 공간의 참조값(해시코드)를 객체에게 반환해준다.
-> 이어서 생성자를 호출한다.

클래스	객체변수	=	new	클래스();
자료형	참조값 저장 (인스턴스 핸들)		메모리(Heap) 할당 인스턴스 생성, 참조값 리턴(→객체)	생성자 호출

•

2). 예시

```
class Calculator //클래스(공장)
{
    int left, right;
    public void setOprands(int left, int right){
        this.left = left;
        this.right = right;
    }
    public void sum(){System.out.println(this.left + this.right);}
    public void avg(){System.out.println((this.left + this.right)/2);}
}

public static void main(String[]args){
    Calculator c1 = new Calculator();
    //인스턴스(제품)
    c1.메소드();
}
```

📖 2. 접근 한정자

1). 한정자

• 호출자의 위치	public	protected	internal	private
클래스 내부 끼리	✓	✓	✓	✓
파생 클래스 끼리 (Same 어셈)	✓	✓	✓	✗
비파생 클래스 끼리 (Same 어셈)	✓	✗	✓	✗
파생 클래스 끼리 (diff 어셈)	✓	✓	✗	✗
비파생 내부 끼리 (diff 어셈)	✓	✗	✗	✗

2). *왜 뭐야?* : Internal

㉔ unity 에서 인스펙터 창에서 변수를 숨기고 싶을때 사용 가능하기도 한다.

- 근데 보통 [HideInInspector] 사용해야한다.

㉔ 개발 관점

- public 이랑 비슷한데 같은 어셈블리끼리 공유가 된다.
- 개발에서는 namespace가 다르다? 그렇다면 어셈블리가 다른것.
 - 어셈블리가 뭘 개소리? ㅋㅋ;

📖 3. Static

1). 클래스 변수 & 인스턴스 변수?

클래스 내에서 멤버변수(클래스에 포함된 변수)는 선언된 위치에 따라 2가지로 구분됨.

1. 클래스 변수 : static variable
2. 인스턴스 변수 : instance variable

```
1 class Car {  
2     static int modelOutput; // 1. 클래스 변수  
3     String modelName;      // 2. 인스턴스 변수  
4 }
```

2). 예시

```
public class Enemy  
{  
    static int mEnemyCount = 0; 클래스 변수  
    string mEnemyName; 인스턴스 변수  
  
    public Enemy(string _enemyName){ mEnemyCount++; this.mEnemyName = _enemyName;}  
}  
.....  
  
static void Main(string[] args)  
{  
    Enemy 푸에르 = new Enemy("푸에르");  
    Enemy 에테르누스 = new Enemy("에테르누스");  
    Enemy 방패르누스 = new Enemy("방패르누스");  
  
    > 1. 클래스 변수 사용하기 <  
    int x = Enemy.mEnemyCount; // > 3  
  
    > 2. 인스턴스 변수 사용하기 <  
    string 적이름 = 푸에르.mEnemyName + 에테르누스.mEnemyName + 방패르누스.mEnemyName;
```

```
Console.WriteLine(적이름); // > 푸에르에테르누스방패르누스  
}
```

3). 정의

1. 클래스 변수(static variable)

- **Static** 키워드를 가진것
- 클래스 영역에 위치한 변수 중에서 static 키워드를 가지는 변수를 클래스 변수(static variable)라고 합니다.
- 오직 클래스를통해서 접근가능합니다

2. 인스턴스 변수(instance variable)

- 키워드 없는것 그냥우리가 일반적으로 쓸때
- 변수 중 static 키워드를 가지지 않는 변수는 인스턴스 변수(instance variable)라고 합니다.
- 오직 생성된 인스턴스를통해서 접근 가능합니다.

4). Static은 변수말고도 메소드에서도 사용 가능하다.

- http://www.tcpschool.com/java/java_member_method

4. 클래스의 this

1). this 란... 🔍

- 클래스를 이용해 구체적인 제품으로 만든 Instance를 가르키는 것
그 인스턴스 그 자신의 참조 변수

2). 왜 쓰는건가? 😊

Doc 이나, 블로그 돌아다니면서 알아본 결과..

㉠ 명시성 ㉡ (Class 작성 관점)

Class 메소드 작성시, **매개변수**와 **인스턴스의 멤버변수** 명시적으로 구분하기 위해 **this** 사용

- ex)

매개변수와 멤버변수의 이름이 같습니다.
이럴때 **this** 를 사용하여 구분할 수 있습니다.

```
/*클래스*/
class Foo {
    private int value;
    Foo(int value(매개변수) ){
        /*    value(멤버변수) = value(매개변수) ; */?? 이런식으로 작성해도 괜찮긴한데
        this.value(멤버변수) = value(매개변수) ; 이렇게 해야지 매개변수랑 멤버랑 확실하
    }
}
/*인스턴스*/
Foo fooInstance = new Foo(30);
```

- 그래서 매개변수와 멤버변수 구별안되는일을 피하기 위해
웬만해서 `void Foo(int _매개변수) {...}` 이런식으로 **언더바**를 작성한다.

⑥ 체이닝 📖 (Instance 사용 관점)

문장을 마치지 않고 메소드 호출을 이어나갈 수 있다.
가독성을 향상시킬 수 있다.

- 인스턴스의 **메소드 사용시** 다음을 비교해보자

None 체이닝	체이닝
<pre>1 // 너비는 10, 높이는 20인 박스를 그리 2 Box olBox3 = new OutlinedBox(); 3 olBox3.SetWidth(10); 4 olBox3.SetHeight(20);</pre>	<pre>1 // 너비는 10, 높이는 20인 박스를 그리 2 Box olBox3 = new OutlinedBox(); 3 olBox3.SetWidth(10).SetHeight(20);</pre>

- 🤖 와! 한줄로 점찍어서 작성어케함?
 - **this**를 이용한 메소드 체이닝

5. 구조체

1). 클래스 VS 구조체

- 타입차이에 따른 메모리 할당

	Class	Struct
타입	Reference	Value
메모리할당	Heap	Stack

stack

- 지역 변수를 저장하며, 실행 중인 함수를 찾아 계산을 수행함
- 변수들은 Stack으로 저장(후입선출)

heap

- 참조 타입들이 이 곳에 할당된다.
- 메모리 누수의 대상이 된다.

2). 클래스 써버리지 뭘.. 왜?

- 아까말했듯 클래스는 힙영역에 할당된다. 반대로 구조체는 Stack에 들어가므로 가비지컬렉터가 덜 일해도 된다.
메소드를 쓰지 않고 오직 데이터만 그룹 시키고 싶을때, 딱좋다~!
- 그리고 생성자 오버로딩에 사용되기도 한다.

3). 사용법

구조체가 가능한것

- ③ 프로퍼티 : **get, set**
 - 예시

```
class _className_  
{  
    _type_ _fieldName_  
    _접근한정자_ _type_ _프로퍼티명_{get; set;}
```

```

        _접근한정자_ _type_ _프로퍼티명_
    {
        get{return;}
        set{return;}
    }
}

```

- ⑥ 생성자
- ⑦ 이벤트
- ⑧ **System.Object** 매서드 `readonly override`

4). 참조

① C# 구조체를 써야하는 이유

② C# 구조체 (프로퍼티 & 생성자 & 이벤트)

- property & constructor & event

```

public struct Coords {
    private int _x, _y;
    public Coordinate(int x, int y) { this.x = x; this.y = y; }
    // 2. 생성자
    // _접근한정자_ _type_ _프로퍼티명_{get{...}; set{...};}
    public int x //1. property
    {
        get{ return _x; }
        set{ _x = value; CoordinatesChanged(_x);}
    }
    public int y //1. property
    {
        get{ return _y; }
        set{ _y = value; CoordinatesChanged(_y);}
    }

    public readonly override string ToString() => $"({x}, {y})";
    //4. readonly override
    public event Action<int> CoordinatesChanged;
    //3. 이벤트
}

```

- 좌표가 변경 될 때 발생하는 CoordinatesChanged event 가 포함되어 있습니다.

- 다음 예제는 이벤트를 처리를 보여줍니다. xyCoordinatesChanged

```
class Program
{
    static void Main(string[] args) {
        Coordinate point = new Coordinate();
        point.CoordinatesChanged += StructEventHandler;
        point.x = 10; point.y = 20;
    }

    static void StructEventHandler(int point) {
        Console.WriteLine("Coordinate changed to {0}", point);
    }
}
```

6. 생성자 & 생성자 위임

1). 생성자는 왜쓰나?

- 클래스를 가지고 객체를 생성한다고 인스턴스 변수가 초기화되지는 않는다
- 인스턴스 변수의 초기화하기 위해서는 생성자 함수를 정의한다. 마치 메소드 처럼
- public과 같은 한정자도, 리턴형이든 아~무것도 적지말고 작성하면 된다.

2). 생성자 위임

- 생성자 위임 사용안하면..

```
1 class Time {  
2     int h, m ,s;  
3     Time(int h, int m, int s) { this.h = h; this.m = m; this.s = s;}  
4 }  
5  
6 static void Main(string[] argv){  
7     //20초 시간 저장  
8     Time time = new Time(0,0,20);  
9 }
```

- 0시간 0분 20초를 달렸다? ✕ Time(0, 0, 20);
20초 달렸으면 : Time(20);
5분 0초 달렸으면 : Time(5, 0);
6시간을 달렸으면 : Time(6, 0, 0);
- 생성자 매개변수 순서에 상관없이 잘 작동하게끔 만들고 싶다..
 1. 매개변수가 일부 작성이 안되더라도 알아서 이해하게 만들기
 2. 매개변수의 순서를 고정하고 싶기

- 생성자 위임 사용하면..

```
1  class Time {
2      private int h, m, s;
3
4      public Time(int h, int m, int s) { this.h = h; this.m = m; this.s = s; }
5      public Time(int m, int s) : this(0, m, s) { }
6      public Time(int s) : this(0, 0, s) { }
7      public Time() : this(0, 0, 0) { }
8
9      public override string ToString()
10     {
11         String ret = $"{s}초";
12         if (m != 0) { ret = String.Format("{0}분 {1}", m, ret); }
13         if (h != 0) { ret = String.Format("{0}시 {1}", h, ret); }
14         return ret;
15     }
16 }
17 static void Main(string[] argv)
18 {
19     Time[] T = new Time[3];
20     T[0] = new Time(6, 24, 5);
21     T[1] = new Time(12, 30);
22     T[2] = new Time(35);
23     foreach(Time E in T) { Console.WriteLine(E.ToString()); }
24 }
```

📖 7. 객체 배열

절차

1. 객체배열 new
2. 객체배열에 있는 인스턴스 각각마다 new

```
List<Class> instanceArray = new List<Class>();
```

📖 8. 화살표 함수

C#은 함수형 프로그래밍도 제공한다.