

초보자도 할 수 있는 C# Programming

Fast Campus
Online

한창민 강사.

7-1강

쉽게 이해하는 클래스 문법

- Check
 - 075_Check(예외 처리는 제외)

```
학생 ID를 입력하세요? 32
국어 점수를 입력하세요? 12
수학 점수를 입력하세요? 43
영어 점수를 입력하세요? 54
```

```
학생 ID를 입력하세요? 21
국어 점수를 입력하세요? 34
수학 점수를 입력하세요? 25
영어 점수를 입력하세요? 90
```

```
학생 ID를 입력하세요? 4
국어 점수를 입력하세요? 90
수학 점수를 입력하세요? 87
영어 점수를 입력하세요? 32
```

```
학생 ID: 1
학생 ID: 23
학생 ID: 67
학생 아이디를 입력하세요? (0)나가기 2
학생 아이디가 없어요. 다시 입력하세요
학생 ID: 1
학생 ID: 23
학생 ID: 67
학생 아이디를 입력하세요? (0)나가기 1
국어 점수: 54
수학 점수: 67
영어 점수: 90
총점: 211
평균: 70.33334
```

```
학생 ID: 1
학생 ID: 23
학생 ID: 67
학생 아이디를 입력하세요? (0)나가기 23
국어 점수: 56
수학 점수: 8
영어 점수: 90
총점: 154
평균: 51.33333
```

```
학생 ID: 1
학생 ID: 23
학생 ID: 67
학생 아이디를 입력하세요? (0)나가기
```

• Check

- 076_Check(예외 처리는 제외)

- 두수를 입력 받아 더하기 연산을 10번까지 저장 가능한 프로그램
- 반드시 배열을 사용해서 연산 저장
- 연산을 끝내고 나갈 때 기존 연산 자료를 한번에 보여주기

```
static int InputNumber(int count) ...
```

```
static void PrintResult(int a, int b) ...
```

```
static bool CheckEnd() ...
```

첫번째 수를 입력 해 주세요?1

두번째 수를 입력 해 주세요?1

1 + 1 = 2

추가로 계산할까요(1: OK, 0: NO, 단 총 10번까지 가능)1

첫번째 수를 입력 해 주세요?2

두번째 수를 입력 해 주세요?2

2 + 2 = 4

추가로 계산할까요(1: OK, 0: NO, 단 총 10번까지 가능)1

첫번째 수를 입력 해 주세요?43

두번째 수를 입력 해 주세요?554

43 + 554 = 597

추가로 계산할까요(1: OK, 0: NO, 단 총 10번까지 가능)0

1 + 1 = 2

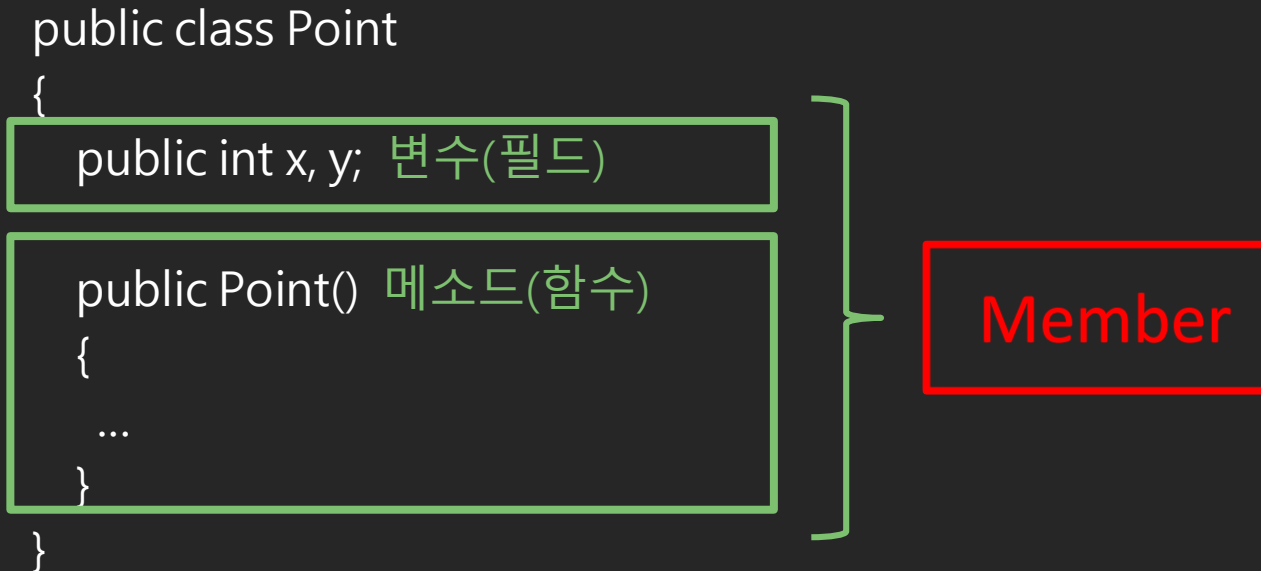
2 + 2 = 4

43 + 554 = 597

- 클래스 개념

- 클래스는 사용자가 직접 만든 틀
- 변수(필드)와 함수(메소드)를 하나의 단위로 결합
- 상속, 다형성, 파생 클래스 클래스의 특수화 메커니즘

```
public class Point
{
    public int x, y; 변수(필드)
    public Point() 메소드(함수)
    {
        ...
    }
}
```



Member

- 접근 한정자 → 소스코드 (`_077_class`)
 - 클래스의 멤버에 액세스(접근) 할 수 있는 수준

- `public`
 - 액세스가 제한되지 않음
- `protected`
 - 이 클래스 또는 이 클래스에서 파생된 클래스로만 액세스가 제한됨
- `internal`
 - 현재 어셈블리(.exe, .dll 등)로만 액세스가 제한됨
- `protected internal`
 - 포함하는 클래스, 포함하는 클래스에서 파생된 클래스 또는 동일한 어셈블리 내의 클래스로만 액세스가 제한됨
- `private`
 - 이 클래스로만 액세스가 제한됨
- `private protected`
 - 포함하는 클래스 또는 동일한 어셈블리 내의 포함하는 유형으로부터 파생된 클래스로만 액세스가 제한됨

- new 키워드

- 클래스의 객체 생성

- 클래스명 객체명 = new 생성자

- AA aa = new AA();

- 생성자, 소멸자

- 생성자: 객체 생성할 때 호출(new)

- 소멸자: 객체가 소멸될 때 호출(GC)

- 생성자

- 클래스 명 동일
- 리턴형 X
- 접근 한정자 **public**
- 멤버 변수 초기화
- 객체 생성시 자동으로 처리하는 것들

- 소멸자

- 클래스 명 동일
- 리턴형 X
- 접근 한정자 X
- '~' 키워드로 구성

- this 키워드

- 객체 자신을 참조하는 키워드
- 사용처
 - 함수의 파라미터 이름과 멤버 변수 이름이 동일
 - 클래스 내부에서 멤버변수를 접근

➡ 소스코드 (_079_this)

```
class AA
{
    int a;

    public AA(int a) {
        this.a = a;
    }
}
```

- static 키워드

- 클래스의 멤버를 객체 생성 없이 사용 가능
- 클래스 static 필드(변수)
- 클래스 static 메서드(함수)
- static 메서드 내부에 사용하는 변수는 반드시 static

➡ 소스코드 (_080_static)

```
class AA
{
    public static int a;
    public static int b;

    public static void Print() {
        Console.WriteLine("a: {0}", a);
        Console.WriteLine("b: {0}", b);
    }
}
```

- static 키워드

- 클래스의 멤버를 객체 생성 없이 사용 가능
- 클래스 static 필드(변수)
- 클래스 static 메서드(함수)
- static 메소드 내부에 사용하는 변수는 반드시 static

➡ 소스코드 (_080_static)

```
class AA
{
    public static int a;
    public static int b;

    public static void Print() {
        Console.WriteLine("a: {0}", a);
        Console.WriteLine("b: {0}", b);
    }
}
```

- 클래스의 메소드 활용
 - 클래스가 메소드의 파라미터
 - 클래스가 메소드의 리턴형

➡ 소스코드 (_081_class_parameter)

```
static void CopyRefClass(AA aa) {  
    AA refAA = aa;  
    refAA.a = 100;  
    refAA.b = 10000;  
}
```

```
static AA CopyDeepClass(AA aa) {  
    AA tempAA = new AA();  
  
    tempAA.a = aa.a;  
    tempAA.b = aa.b;  
    tempAA.a = 0;  
  
    return tempAA;  
}
```