

# Method

# 메소드

```
class 클래스의_이름
```

```
{
```

메소드의 속성을 수식하는  
한정자를 둘 수 있습니다.

메소드에 입력하는  
매개 변수의 목록입니다.

한정자 반환 형식 메소드의\_이름( 매개 변수\_목록 )

```
{
```

```
// 실행하고자 하는 코드 1
```

```
// 실행하고자 하는 코드 2
```

```
// ...
```

```
// 실행하고자 하는 코드 n
```

```
return 메소드의_결과;
```

```
}
```

```
}
```

메소드 결과의 데이터 형식은  
메소드의 반환 형식과 일치해야 합니다.

```
class Calculator
```

```
{
```

```
    public static int Plus( int a, int b )
```

```
    {
```

```
        Console.WriteLine("Input : {0}, {1}", a, b);
```

```
        int result = a + b;
```

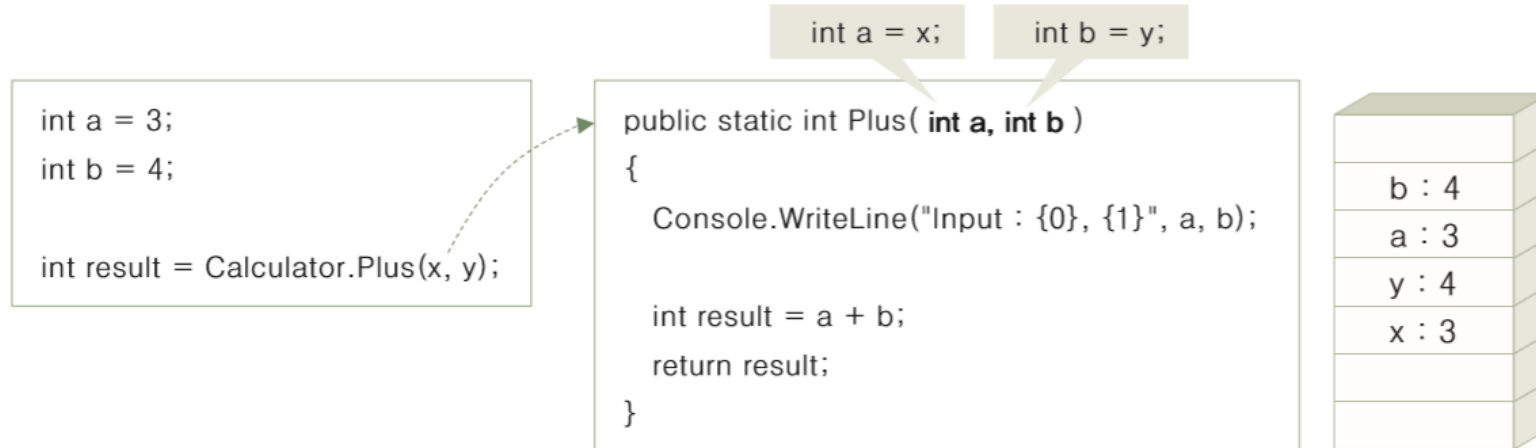
```
        return result;
```

```
    }
```

```
}
```

## 매개변수

## ❖ 매개변수가 전달되는 과정



## ❖ 값에 의한 전달:

- 메소드 호출 시 데이터를 복사해 매개 변수에 전달

## ❖ 참조에 의한 전달:

- 메소드 호출 시 직접 원본 변수의 값을 바꾸는 방법 (ref 키워드)

## ❖ 출력 전용에 의한 전달:

- 컴파일러를 통해 결과를 할당하지 않는 버그 가능성 제거 (out 키워드, 권장)

## ❖ 두 가지 이상의 결과가 필요한 메소드 - ref, out 이용 가능

## 메소드 오버로딩

- ❖ 하나의 메소드 이름에 여러 개의 구현을 올리는 것
- ❖ 매개 변수의 수와 형식을 분석해 호출할 메소드 결정

```
int Plus(int a, int b)
```

```
{
```

```
    return a + b;
```

```
}
```

```
double Plus(double a, double b)
```

```
{
```

```
    return a + b;
```

```
}
```

```
int    result1 = Plus( 1, 2 );
```

```
double result2 = Plus( 3.1, 2.4 );
```

int Plus(int, int)를 호출합니다.

double Plus(double, double)를 호출합니다.

매개 변수 수와  
형식

- ❖ 이름에 대한 고민을 덜어준다.

- ❖ 코드의 일관성 제공

## 가변길이 매개변수

- ❖ 형식은 같으나 매개 변수의 개수만 유연하게 달라질 수 있는 경우에 적합.
- ❖ params 키워드와 배열 이용

```
int Sum( params int[] args )  
{  
    int sum = 0;  
  
    for(int i=0; i<args.Length; i++)  
    {  
        sum += args[i];  
    }  
    return sum;  
}
```

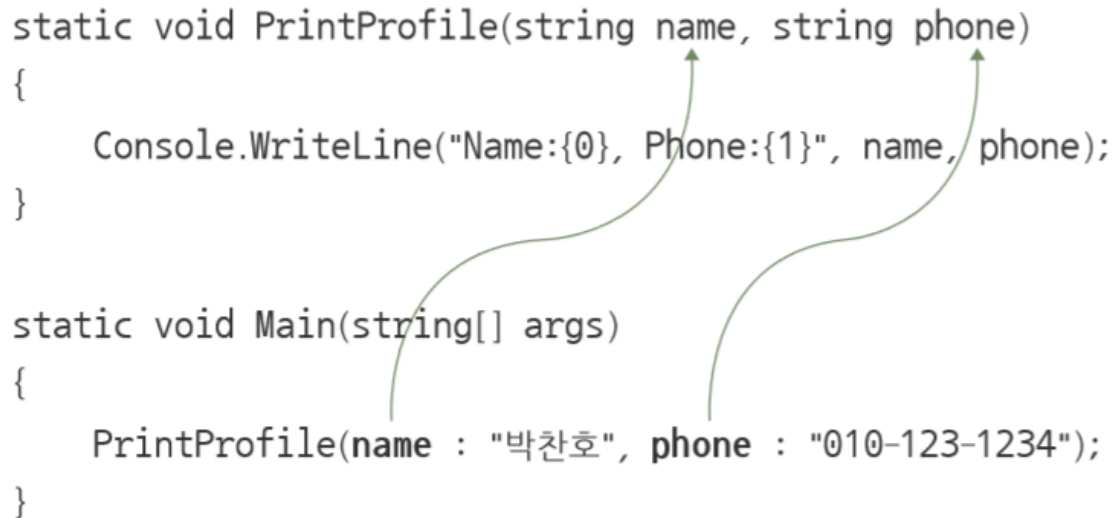
Sum() 메소드에 입력한 모든  
메소드는 args 배열에 담깁니다.

```
int total = 0;  
  
total = Sum( 1, 2 );  
total = Sum( 1, 2, 3 );  
total = Sum( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 );
```

## 명명된 매개변수 / 선택적 매개변수

### ❖ 메소드를 호출할 때 매개 변수의 이름에 근거해서 데이터를 할당하는 기능 (가독성에 도움)

#### ■ 매개 변수 이름: 값



```
static void PrintProfile(string name, string phone)
{
    Console.WriteLine("Name:{0}, Phone:{1}", name, phone);
}

static void Main(string[] args)
{
    PrintProfile(name : "박찬호", phone : "010-123-1234");
}
```

The diagram illustrates the process of passing arguments by name. In the `Main` method, the call `PrintProfile(name : "박찬호", phone : "010-123-1234");` uses named arguments. Curved arrows show the mapping from the named arguments `name` and `phone` in the `Main` call to the corresponding parameters `name` and `phone` in the `PrintProfile` method signature.

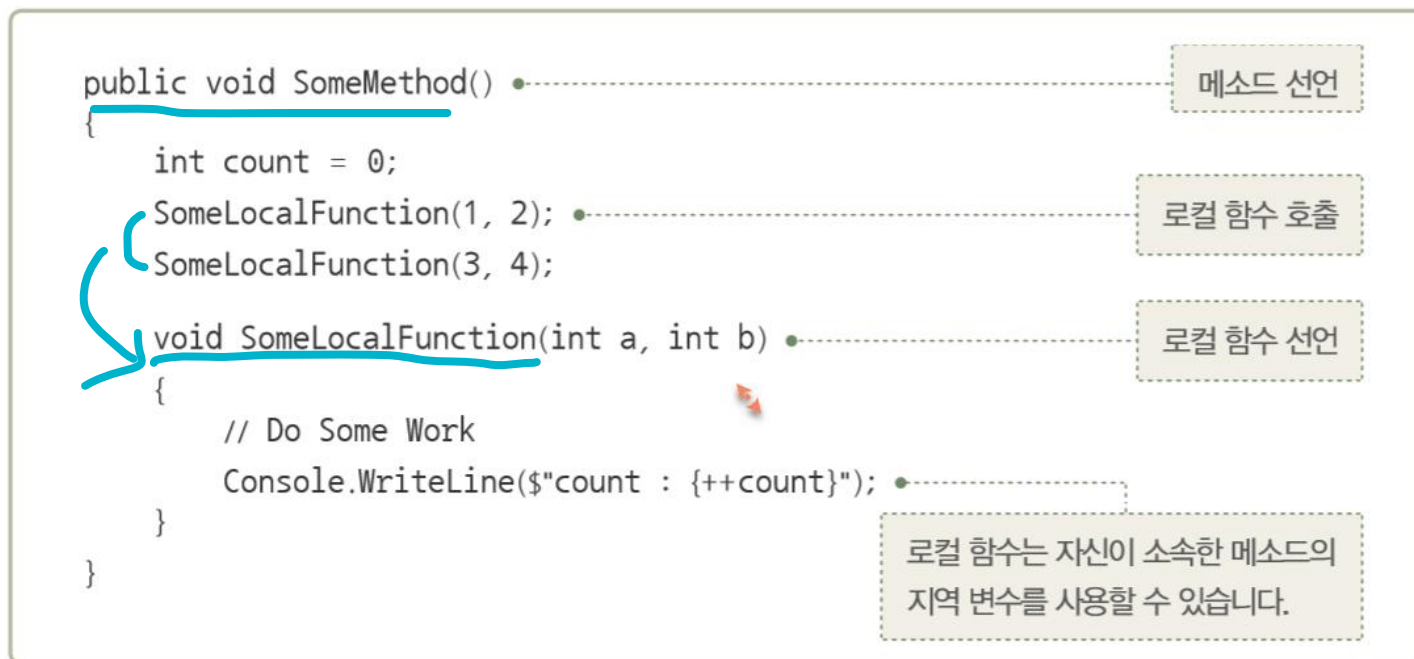
### ❖ 메소드의 매개 변수는 기본 값을 가질 수 있다.

- 필요에 따라 데이터를 할당하거나 할당하지 않을 자유
- 위치-필수 매개 변수(있다면) 다음.

## 로컬함수

## ❖ 메소드 내에 선언하고, 그 안에서만 사용하는 특별한 함수

- 클래스의 멤버가 아니므로 함수라고 명명
- 자신이 존재하는 지역에 선언된 변수 사용
- 메소드 밖에서는 다시 쓸 일 없는 반복적인 작업을 하나의 이름 아래 묶어 놓는 데 제격
- C# 7.0에서 추가





Visual C#