06.메소드로 코드 간추리기

6.1 메소드란?

-코드뭉치를 하나의 이름아래 묶은 것 ( 메소드, 함수, 프로시저, 서브루틴, 서브프로그램 다 비슷한말)

Class Calculator

{

Public static int~

}

메소드를 동작시키기 위해서는 호출해야한다.

매개변수를 넘김 -> (메소드) 매개변수를 가공! -> 결과출력

---------------------------------

class Calculator

{

public static int Plus(int a, int b)

{ return a + b; }

public static int Minus(int a, int b)

{ return a - b; }

static void Main(string[] args)

{

int result = Calculator.Plus(3, 4);

Console.WriteLine(result);

result = Calculator.Minus(5,2);

Console.WriteLine(result);

}

}

--------------------------

6.2 return에 대하여

-점프문의 한 종류임

-프로그램의 흐름을 호출자에게로 돌려놓음.

-메소드의 어느 위치에서나 호출 가능

-----------------------------------

class MainApp

{

static int Fibonacci(int n)

{

if(n < 2)

return n;

else

return Fibonacci(n - 1) + Fibonacci(n - 2);

}

static void PrintProfile(string name, string phone)

{

if (name == "")

{

Console.WriteLine("이름을 입력해 주세요. : ");

return;

}

Console.WriteLine("Name:{0}, Phone:{1}", name, phone);

}

static void Main(string[] args)

{

Console.WriteLine("10번째 피보나치 수 : {0}",Fibonacci(10));

PrintProfile("", "123-4567");

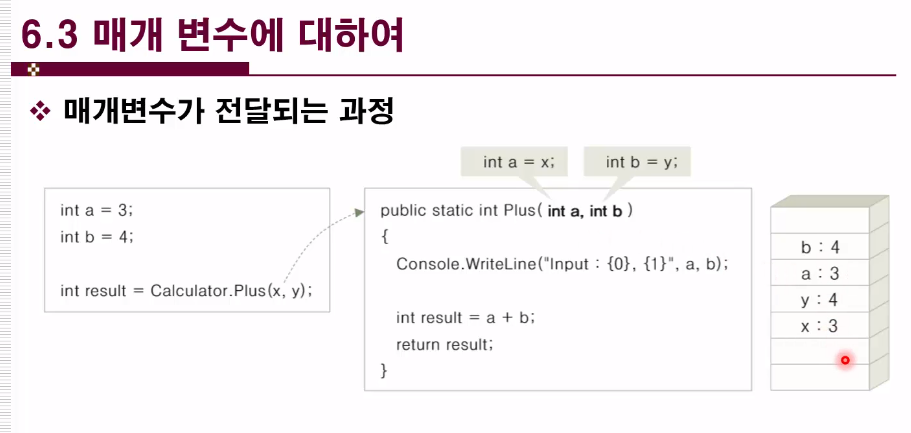
PrintProfile("박소은", "456-7890");

}

}

6.3 매개변수에 대하여

-값에 의한 전달: 메소드 호출 시 데이터를 복사해 매개변수에 전달.



class Program

{

public static void Swap(int a, int b)

{

int temp = b;

b = a;

a = temp;

}

static void Main(string[] args)

{

int x = 3;

int y = 4;

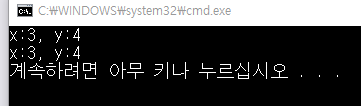
Console.WriteLine("x:{0}, y:{1}", x, y);

Swap(x, y);

Console.WriteLine("x:{0}, y:{1}", x, y);

}

}

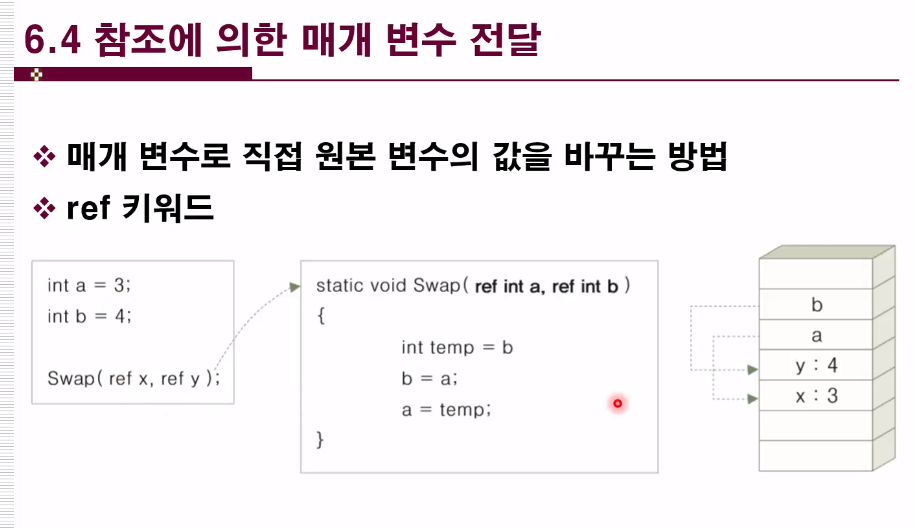


원본 변수의 값을 바꾸지 못하고 값만 바꿈.

6.4 참조에 의한 매개변수 전달

-매개변수로 직접 원본 변수의 값을 바꾸는 방법

-ref 키워드



class Program

{

public static void Swap(ref int a, ref int b)

{

int temp = b;

b = a;

a = temp;

}

static void Main(string[] args)

{

int x = 3;

int y = 4;

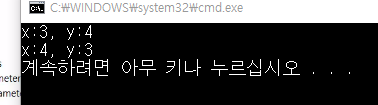
Console.WriteLine("x:{0}, y:{1}", x, y);

Swap(ref x, ref y);

Console.WriteLine("x:{0}, y:{1}", x, y);

}

}



6.5 메소드의 결과를 참조로 반환하기

-메소드 호출자가 반환 결과를 참조로 다룰 수 있다. Ref

1. Ref 한정자로 메소드 선언

2. Return문이 반환하는 변수에 ref키워드 사용

6.6 출력전용 매개변수

-두가지 이상의 결과가 필요한 메소드

-ref 키워드를 이용한 방법

Void Divide( int a, int bm ref int quotient, ref int remainder)

{ ~ }

ref라는 키워드 보다는 out 키워드를많이 사용함(권장)

컴파일러를 통해 결과를 할당하지 않는 버그를 만들 가능성을 제거함

class Program

{

static void Divide(int a, int b, out int quotient, out int remainder)

{

quotient = a / b;

remainder = a % b;

}

static void Main(string[] args)

{

int a = 20;

int b = 3;

Divide(a, b, out int c, out int d);

Console.WriteLine("a:{0}, b:{1}, a/b:{2}, a%b:{3}", a, b, c, d);

}

}

6.7 메소드 오버로딩

-하나의 메소드 이름에 여러 개의 구현을 올리는것

-매개변수의 수와 형식을 분석해 호출할 메소드 결정

-이름에 대한 고민을 덜어주고, 코드를 일관성 있게 제공함

class Program

{

static int Plus(int a, int b)

{

Console.WriteLine("Calling int Plus(int,int)...");

return a + b;

}

static int Plus(int a, int b,int c)

{

Console.WriteLine("Calling int Plus(int,int,int)...");

return a + b+c;

}

static double Plus(double a, double b)

{

Console.WriteLine("Calling int Plus(double,double)...");

return a + b;

}

static double Plus(int a, double b)

{

Console.WriteLine("Calling int Plus(int,double)...");

return a + b;

}

static void Main(string[] args)

{

Console.WriteLine(Plus(1, 2));

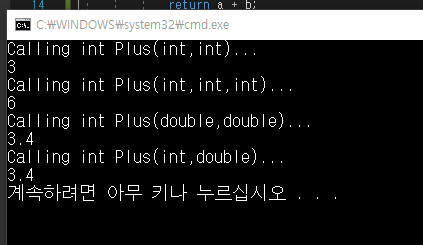
Console.WriteLine(Plus(1, 2,3));

Console.WriteLine(Plus(1.0, 2.4));

Console.WriteLine(Plus(1, 2.4));

}

}



6.8 가변길이 매개 변수

-형식은 같으나 매개변수의 개수만 유연하게 달라질 수 있는 경우에 적합.

Int total = 0;

Total = sum(1,2);

Total = sum(1,2,3);

Total = sum(1,2,3,4,5,6,7,8,9);

..이런경우에

-params 키워드와 배열이용

class Program

{

static int Sum(params int[] args)

{

Console.Write("Summing...");

int sum = 0;

for(int i = 0; i<args.Length; i++)

{

if (i > 0)

Console.Write(",");

Console.Write(args[i]);

sum += args[i];

}

Console.WriteLine();

return sum;

}

static void Main(string[] args)

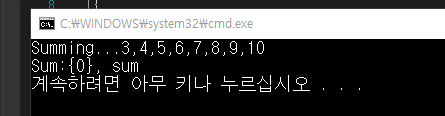
{

int sum = Sum(3, 4, 5, 6, 7, 8, 9, 10);

Console.WriteLine("Sum:{0}, sum");

}

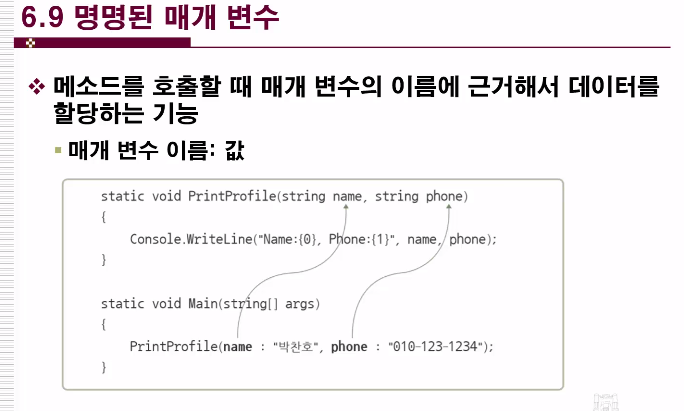
}



6.9 명명된 매개 변수

- 메소드를 호출할 때 매개변수의 이름에 근거해서 데이터를 할당하는 기능

- 매개 변수 이름: 값(이름에근거해서 호출함)



-가독성에 도움이 됨

class Program

{

static void PrintProfile(string name, string phone)

{

Console.WriteLine("Name:{0}, Phone:{1}", name, phone);

}

static void Main(string[] args)

{

PrintProfile(name: "박소은", phone: "010-123-1233");

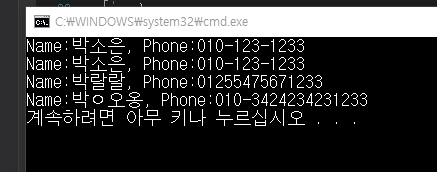
PrintProfile( phone: "010-123-1233", name: "박소은");

PrintProfile("박랄랄","01255475671233");

PrintProfile("박ㅇ오옹", phone: "010-3424234231233");

}

}



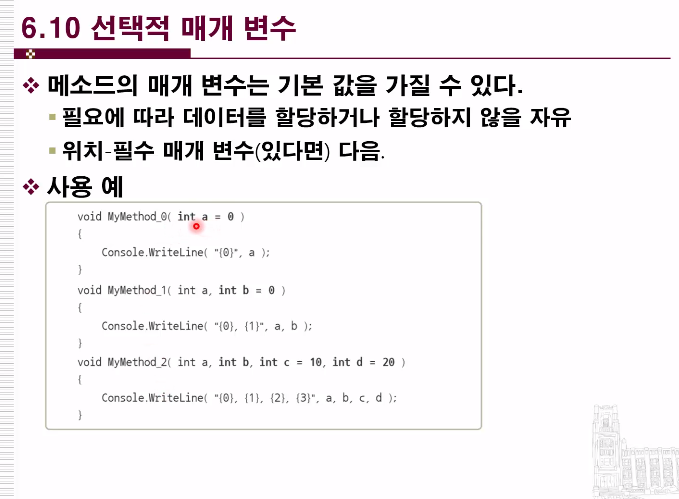
-순서에 상관없이, 아무 에러없이 정상적으로 출력됨

6.10 선택적 매개 변수

-메소드의 매개변수는 기본 값을 가질 수 있다.

-필요에 따라 데이터를 할당하거나 할당하지 않을 자유

-위치 – 필수 매개 변수(있다면) 다음.



class Program

{

static void PrintProfile(string name, string phone="")

{

Console.WriteLine("Name:{0}, Phone:{1}", name, phone);

}

static void Main(string[] args)

{

PrintProfile("태연");

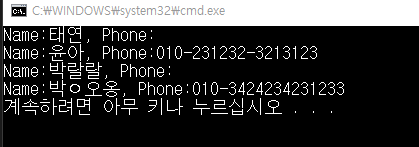
PrintProfile("윤아","010-231232-3213123");

PrintProfile(name:"박랄랄");

PrintProfile(name:"박ㅇ오옹", phone: "010-3424234231233");

}

}



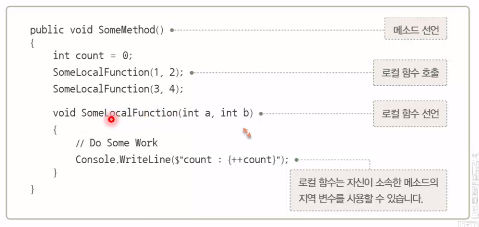
6.11 로컬 함수

#메소드 내에 선언하고, 그 안에서만 사용하는 특별한 함수

-클래스의 멤버가 아니라서 함수라고 명명

-자신이 존재하는 지역에 선언된 변수 사용

-메소드 밖에서는 다시 쓸 일 없는 반복적인 작업을 하나의 이름아래 묶어 놓는데 제격



class Program

{

static string ToLowerString(string input)

{

var arr = input.ToCharArray();

for (int i = 0; i < arr.Length; i++)

{

arr[i] = ToLowerChar(i);

}

//입력받은 string을 한자씩 한자씩 아스키값을 가지고 대문자를 소문자로 변경

char ToLowerChar(int i)

{

if (arr[i] < 65 || arr[i] > 90) //

return arr[i];

else

return (char)(arr[i] + 32);

}

return new string(arr);

}

static void Main(string[] args)

{

Console.WriteLine(ToLowerString("Hello"));

Console.WriteLine(ToLowerString("GOOD MORNING"));

Console.WriteLine(ToLowerString("This is C#"));

}

}

