윈도우프로그래밍

8. 메서드(1)

2023. 4. 28.

심미나

목 차

참고자료: 수업교재(C# 프로그래밍, 2판)

I. 메서드 기본 형태

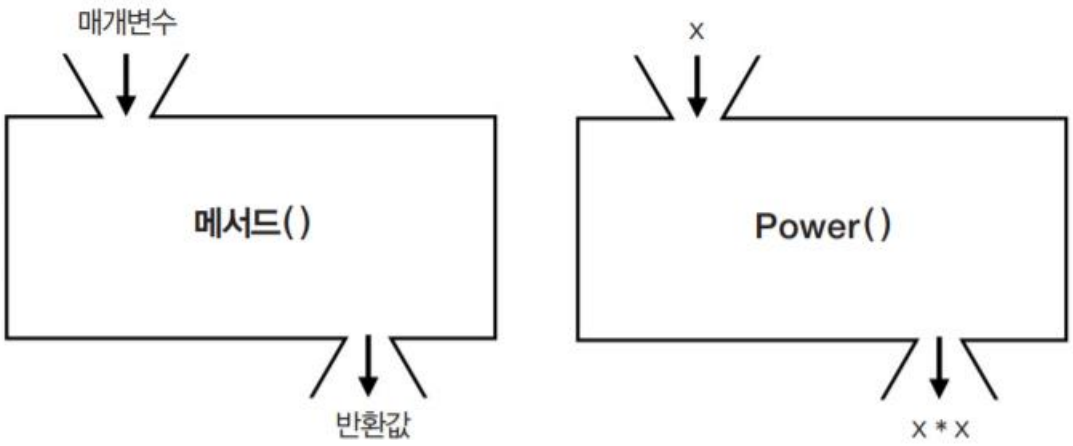
메서드 기본 형태

메서드의 기본 형태

- 메서드의 기본 형태

메서드 기본 형식

```
[접근제한자] [반환형] [메서드 이름] ([매개변수])

{
        [메서드 코드]
}
```



메서드 기본 형태

메서드의 기본 형태

- (예제 6-1) 인스턴스 메서드 생성과 사용(교재 266p)
  - Test 클래스에 매개변수로 넣은 숫자를 제곱하는 Power 메서드 생성하기
  - 코드6-1. 인스턴스 메서드 생성과 사용

```
01  namespace InstanceMethod
02  {
03      class Program
04      {
05          class Test
06          {
07              public int Power(int x)
08              {
09                  return x * x;
10              }
11          }
12
13          static void Main(string[] args)
14          {
15              Test test = new Test();
16              Console.WriteLine(test.Power(10));
17              Console.WriteLine(test.Power(20));
18          }
19      }
20  }
```

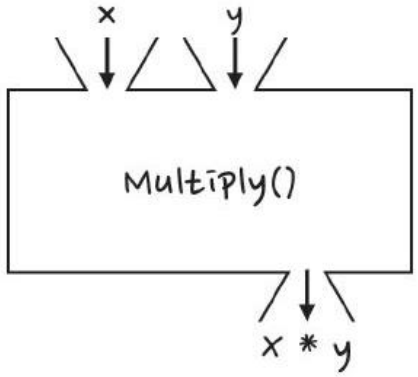
실행 결과

```
100
400
```

메서드 기본 형태

메서드의 기본 형태

- 두 개의 매개변수를 갖는 메서드
  - 메서드는 매개변수를 여러 개 가질 수 있음

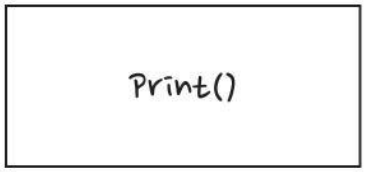



```
class Program
{
    class Test
    {
        public int Multi(int x, int y)
        {
            return x * y;
        }
    }

    static void Main(string[] args)
    {
        Test test = new Test();
        Console.WriteLine(test.Multi(52, 273));
        Console.WriteLine(test.Multi(103, 32));
    }
}
```

메서드 기본 형태

메서드의 기본 형태

- 아무것도 반환하지 않는 메서드
  - 메서드는 아무것도 반환하지 않을 수 있음




```
class Program
{
    class Test
    {
        public void Print()
        {
            Console.WriteLine("Print() 메서드가 호출
되었습니다.");
        }
    }


    static void Main(string[] args)
    {
        Test test = new Test();
        test.Print();
        test.Print();
        test.Print();
    }
}
```

II. 매개변수와 반환

매개변수와 반환

반환 메서드 형태

- 반환 메서드 형태
  - 반환값을 갖는 메서드의 일반적 형태

```
public 자료형 메서드(자료형 매개변수, 자료형 매개변수)
{
    자료형 output = 초기값;

    // output에 값을 계산

    return output;
}
```

매개변수와 반환

반환 메서드 형태

- (예제 6-2) 매개 변수와 반환(1)(교재 270p)
  - 코드6-4. 매개변수와 반환(1) : 매개변수 min부터 max까지 더하는 메서드

```
01  namespace SumMethod
02  {
03      class Program
04      {
05          class Test
06          {
07              public int Sum(int min, int max)
08              {
09                  int output = 0;
10                  for (int i = min; i <= max; i++)
11                  {
12                      output += i;
13                  }
14                  return output;
15              }
16
17              static void Main(string[] args)
18              {
19                  Test test = new Test();
20                  Console.WriteLine(test.Sum(1, 100));
21              }
22      }
23  }
```

실행 결과

```
5050
```

매개변수와 반환

반환 메서드 형태

- (예제 6-3) 매개 변수와 반환(2)(교재 270p)
  - 코드6-5. 매개변수와 반환(2) : 매개변수 min부터 max까지 곱하는 메서드

```
01  namespace MultiplyMethod
02  {
03      class Program
04      {
05          class Test
06          {
07              public int Multiply(int min, int max)
08              {
09                  int output = 1;
10                  for (int i = min; i <= max; i++)
11                  {
12                      output *= i;
13                  }
14                  return output;
15              }
16          }
17
18          static void Main(string[] args)
19          {
20              Test test = new Test();
21              Console.WriteLine(test.Multiply(1, 10));
22          }
23      }
24  }
```

실행 결과

```
3628800
```

him

III. 클래스 메서드

클래스 메서드

- Main() 메서드 - 클래스 메서드
  - 기본 콘솔 프로젝트로 생성되는 Program 클래스의 Main() 메서드는 클래스 메서드

```
class Program
{
    static void Main(string[] args)
    {

    }
}
```

클래스 메서드

- 클래스 메서드 생성 방법

클래스 메서드 기본 형식

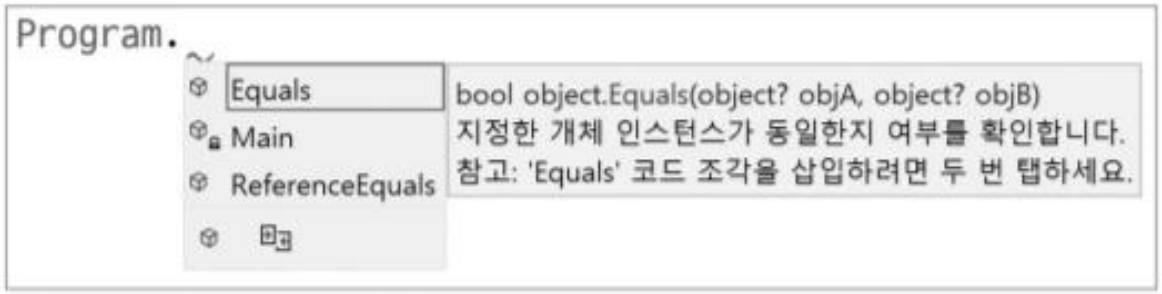
```
[접근제한자]  static [반환형]  [메서드 이름]  ([매개변수])

{
      [메서드 코드]
}
```

  - (비교) 클래스 변수 생성 시, [접근제한자] static [자료형] [변수명]

클래스 메서드

반환 메서드 형태

- 클래스 메서드 사용 방법
  - Program 클래스 내부의 Main() 메서드를 직접 사용할 경우, “클래스명.” 바로 사용



클래스 메서드

반환 메서드 형태

- (예제 6-4) 클래스 메서드 생성과 사용(교재 273p)
  - 코드6-7. 메서드 생성과 사용 : 클래스 메서드 MyMath.Abs() 생성하고 호출하기

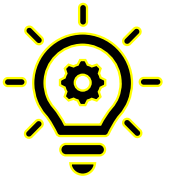
```
01  namespace ClassMethod
02  {
03      class Program
04      {
05          class MyMath
06          {
07              public static int Abs(int input)
08              {
09                  if (input < 0)
10                  {
11                      return -input;
12                  }
13                  else
14                  {
15                      return input;
16                  }
17              }
18          }
19
20          static void Main(string[] args)
21          {
22              Console.WriteLine(MyMath.Abs(52));
23              Console.WriteLine(MyMath.Abs(-273));
24          }
25      }
26  }
```

실행 결과

```
52
273
```

클래스 메서드

(참고) 클래스 메서드에서 사용할 수 있는 것



- 클래스 메서드에서는 메모리에 올라가지 않은 인스턴스 변수, 인스턴스 메서드는 사용 못 함
  - 클래스 메서드에서 인스턴스 변수 사용은 오류가 발생
  - 클래스 메서드에서 클래스 변수만 사용 가능

```
class Program
{
    public int instanceVariable = 10;

    static void Main(string[] args)
    {
        Console.WriteLine(instanceVariable);
    }
}
```

오류 발생

```
class Program
{
    public static int instanceVariable = 10;

    static void Main(string[] args)
    {
        Console.WriteLine instanceVariable);
    }
}
```

추가해주었습니다.

IV. 오버로딩

오버로딩

오버로딩(Overloading)

- 오버로딩
  - 이름은 같고, 매개변수는 다른 메서드를 만드는 것
  - Math.Abs() 메서드의 형태(1)

```
Math.Abs()
```

▲ 1/7 ▼ decimal Math.Abs(**decimal value**)

System.Decimal 숫자의 절대 값을 반환합니다.

value: *System.Decimal.MinValue보다 크거나 같지만 System.Decimal.MaxValue보다 작거나 같은 숫자입니다.*

  - Math.Abs() 메서드의 형태(2)

```
Math.Abs()
```

▲ 3/7 ▼ float Math.Abs(**float value**)

단정밀도 부동 소수점 수의 절대 값을 반환합니다.

value: *System.Single.MinValue보다 크거나 같지만 System.Single.MaxValue보다 작거나 같은 숫자입니다.*

오버로딩

오버로딩(Overloading)

- (예제 6-5) 메서드 오버로딩(교재 275p)
  - 코드6.10. : Abs() 메서드를 다양한 자료형에서 동작할 수 있도록 오버로딩하기

```
01  namespace Overloading
02  {
03      class Program
04      {
05          class MyMath
06          {
07              public static int Abs(int input)
08              {
09                  if (input < 0) { return -input; }
10                  else { return input; }
11              }
12              public static double Abs(double input)
13              {
14                  if (input < 0) { return -input; }
15                  else { return input; }
16              }
17              public static long Abs(long input)
18              {
19                  if (input < 0) { return -input; }
20                  else { return input; }
21              }
22          }
```

shim

오버로딩

오버로딩(Overloading)

- (예제 6-5) 메서드 오버로딩(교재 275p)(계속)
  - 코드6.10. : Abs() 메서드를 다양한 자료형에서 동작할 수 있도록 오버로딩하기

```
23    static void Main(string[] args)
24          {
25              // int
26              Console.WriteLine(MyMath.Abs(52));
27              Console.WriteLine(MyMath.Abs(-273));
28
29              // double
30              Console.WriteLine(MyMath.Abs(52.273));
31              Console.WriteLine(MyMath.Abs(-32.103));
32
33              // long
34              Console.WriteLine(MyMath.Abs(21474836470));
35              Console.WriteLine(MyMath.Abs(-21474836470))
36          }
37      }
38  }
```

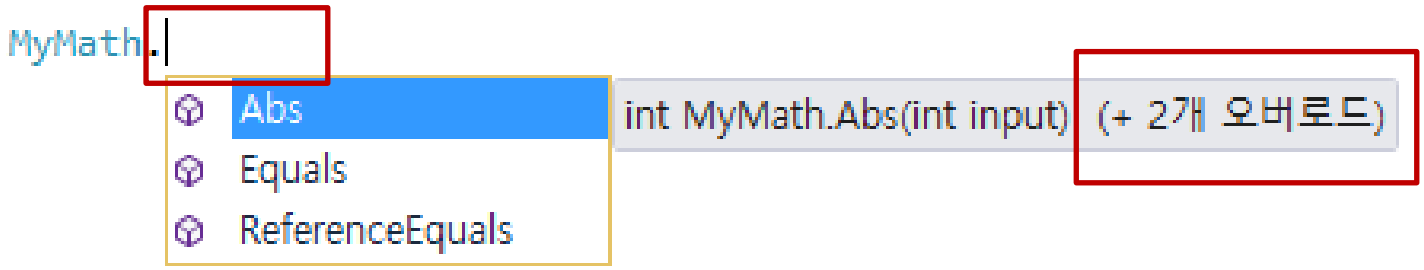
실행 결과

```
52
273
52.273
32.103
21474836470
21474836470
```

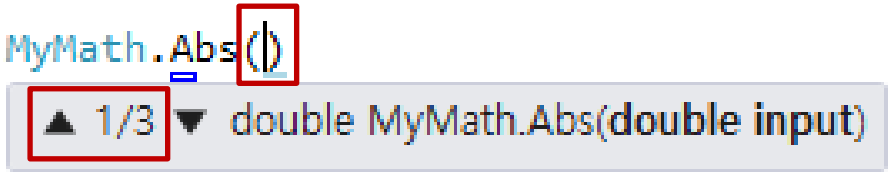
오버로딩

오버로딩(Overloading)

- 자동 완성 기능으로 출력 결과
  - 오버로딩 용어 확인

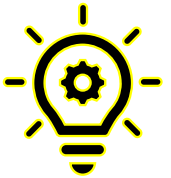


  - 3개의 메서드가 오버로딩됨



오버로딩

(참고) 오버로딩과 반환값의 자료형



- 이름은 같으나 반환값이 다른 경우, 오버로딩 불가
  - 오버로딩은 이름이 같고, 매개변수가 다른 경우에 가능한 것

```
class TestWorld
{
    public int Test(int input) { }
    public double Test(int input) { }
    public double Test(int input) { }
}
```

  - 오버로딩 오류 메시지

```
class TestWorld
{
    참조 0개
    public int Test(int input) { }
    참조 0개
    public double Test(int input) { }
    참조 0개
    public double Test(int input) { }
}
```

double TestWorld.Test(int input)

CS0111: 'Program.TestWorld' 형식은 동일한 매개 변수 형식을 가진 'Test' 멤버를 미리 정의합니다.

CS0161: 'Program.TestWorld.Test(int)' 코드 경로 중 일부만 값을 반환합니다.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

V. 접근 제한자

접근 제한자

접근 제한자

- 가장 기본적인 접근제한자 설정

변수 & 메서드 기본형식

```
[접근제한자] [자료형] [변수 이름]
[접근제한자] [반환형] [메서드 이름] ([매개변수])
{
      [메서드 코드]
}
```

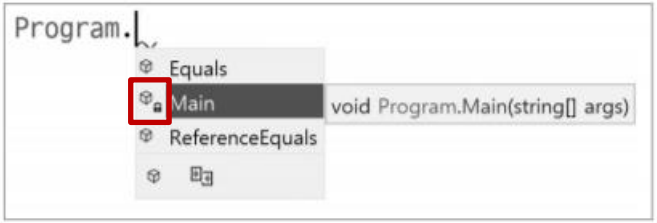
  - 대표적 접근제한자: Public, Private

접근 제한자

private 접근 제한자

- 접근 제한자 생략 시, 자동으로 private 접근 제한자 설정
  - Main() 메서드는 기본적으로 private 메서드

```
01  static void Main(string[] args)
02  {
03
04  }
```



  - 자물쇠가 채워져 있는 Main() 메서드 (Main( ) 메서드의 그림 옆 표시)
    : private 접근 제한자 적용 되었다는 의미
    즉, 자신의 클래스 내부에서만 해당 메서드 사용 가능

접근 제한자

private 접근 제한자

- 다른 클래스에서 Program 클래스의 Main( ) 메서드 호출
  - 외부 클래스에서의 접근

```
class Test
{
    public void TestMethod()
    {
        Program.Main(new string[] { "" });
    }
}

class Program
{
    static void Main(string[] args)
    {

    }
}
```

오류 발생

접근 제한자

private 접근 제한자

- 다른 클래스에서 Program 클래스의 Main( ) 메서드 호출
  - 접근 제한자로 인한 접근불가 상태이므로 오류 발생

```
public void TestMethod()
{
    Program.Main(new string[] { "" });
}
```

void Program.Main(string[] args)

CS0122: '보호 수준 때문에 'Program.Main(string[])'에 액세스할 수 없습니다.

잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

접근 제한자

private 접근 제한자

```
class Program
{
    class Test
    {
        public void TestMethod()
        {
            Program.Main(new string[] { "" });
        }
    }

    public void TestMethod()
    {
        Program.Main(new string[] { "" });
    }

    static void Main(string[] args)
    {

    }
}
```

자신의 클래스 내부에 있는 클래스의 메서드 (내부 클래스 메서드)

자신의 클래스 내부에 있는 메서드

접근 제한자

public 접근 제한자

- 다른 클래스에서 Main( ) 메서드를 호출
  - Public 접근 제한자를 붙인 Main() 메서드

```
class Test
{
   public void TestMethod()
   {
      Program.Main(new string[] { "" });
   }
}

class Program
{
   public static void Main(string[] args)
   {
                  접근 제한자를 추가했습니다.
   }
}
```

  - public 접근 제한자가 걸린 변수 또는 메서드는 모든 곳에서 접근 가능!

X. 실습 및 과제

과제

자율학습

- 세부 과제
  - (교재 p.265~282) 예제 6-1 ~ 6-5
    - 금주는 자율학습으로 진행
    - 메서드 전체 완료 후 8강, 9강 통합한 과제로 제출 예정

- 제출 시 주의사항
  - 과제 제출 없으며 각자 복습차원에서 이해하기

감사합니다

mnshim@sungkyul.ac.kr