

윈도우 프로그래밍

7. 클래스

2023. 4. 14
심미나



목 차

- I. 클래스 개요
- II. 클래스 사용
- III. 클래스 생성
- IV. 클래스의 변수
- V. 추상화
- VI. 실습 및 과제

I. 클래스 개요

클래스 개요



사용자 정의 자료형

- 클래스

- 객체 지향 언어, 원하는 새로운 자료형 정의
- (예시) 주차관리시스템
 - 필요한 변수?

```
int carNumber;    // 차량번호
int inTime;       // 입차시간
int outTime;      // 출차시간
```

- 3대를 관리해야 한다면?
- 관리 차량이 여러 대라면?

```
int[] CarNumbers = new int[10];
int[] inTimes = new int[10];
int[] outTimes = new int[10];
```

```
int carNumberA;
int inTimeA;
int outTimeA;
int carNumberB;
int inTimeB;
int outTimeB;
int carNumberC;
int inTimeC;
int outTimeC;
```

클래스 개요



사용자 정의 자료형

- 클래스

- 하나의 객체가 갖는 모든 속성을 한꺼번에 묶어서 처리
- (예시) 주차관리시스템
 - 차량번호, 입차시간, 출차시간은 모두 하나의 자동차(객체)가 갖는 속성
 - 따라서 3개 속성을 한꺼번에 묶어서 처리할 수 있도록 int 자료형 3개를 저장하는 새로운 자료형을 사용자가 정의 → 객체지향언어의 클래스 기능!

```
class Car
{
    int carNumber;
    int inTime;
    int outTime;
}

static void Main(string[] args)
{
    Car[] cars = new Car[10];
}
```

클래스 개요



사용자 정의 자료형

- 클래스

- 속성을 나타내는 변수와 더불어 행위를 나타내는 메서드 포함
- (예시) 주차관리시스템 - 메서드 사용 코드 포함

```
class Car
{
    int carNumber;
    DateTime inTime;
    DateTime outTime;
```

```
    public void SetInTime()
    {
        this.inTime = DateTime.Now;
    }
```

```
    public void SetOutTime()
    {
        this.outTime = DateTime.Now;
    }
}
```

```
static void Main(string[] args)
{
    Car car = new Car();
    car.SetInTime();
    car.SetOutTime();
}
```

입차시간 기록

출차시간 기록

클래스 개요



클래스와 인스턴스

- 클래스와 인스턴스

Car car = new Car()
클래스 인스턴스 new 키워드 생성자

- ① 클래스: 사용자 정의 자료형
- ② 인스턴스(객체): 클래스 자료형을 변수로 선언한 것
- ③ 생성자: 클래스 이름과 같은 메서드(클래스 이름 뒤에 괄호가 붙은 것)
 - 클래스 이름 대문자로 시작(관례)

II. 클래스 사용

클래스 사용



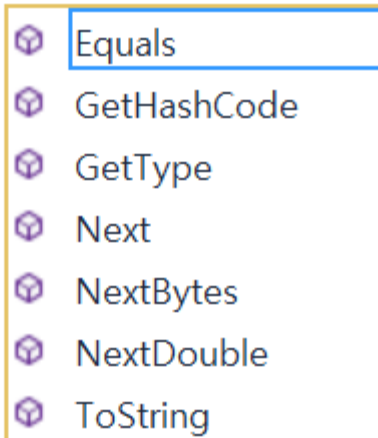
Random 클래스

- Random 클래스
 - 임의의 숫자 생성시 사용
 - (인스턴스 생성 방법)

```
Random random = new Random( )
```

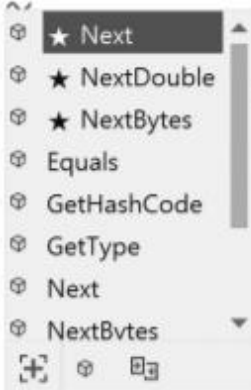
- (예시) Random 클래스의 인스턴스에서 사용 가능한 메서드

random.|



```
Random random = new Random();
```

random.



int Random.Next(int minValue, int maxValue) (+ 2 오버로드)
★ 이 컨텍스트를 기반으로 한 IntelliCode 제안

클래스 사용



Random 클래스

- (예제 5-1) Random 클래스를 사용한 임의의 정수 생성(교재 208p)
 - 코드5-1. Random 클래스를 사용한 임의의 정수 생성

```
01 static void Main(string[] args)
02 {
03     Random random = new Random();
04     Console.WriteLine(random.Next(10, 100));
05     Console.WriteLine(random.Next(10, 100));
06     Console.WriteLine(random.Next(10, 100));
07     Console.WriteLine(random.Next(10, 100));
08     Console.WriteLine(random.Next(10, 100));
09 }
10
11
12
13
```

지정된 최대값보다 작은 음수가 아닌 난수를 반환

실행 결과

86
48
76
49

클래스 사용



Random 클래스



- (예제 5-1) Random 클래스를 사용한 임의의 정수 생성(교재 208p)

Random.Next 메서드

네임스페이스: `System`

어셈블리: `System.Runtime.Extensions.dll`

임의의 정수를 반환합니다.

오버로드

`Next()`

음수가 아닌 임의의 정수를 반환합니다.

`Next(Int32)`

지정된 최댓값보다 작은 음수가 아닌 임의의 정수를 반환합니다.

`Next(Int32, Int32)`

지정된 범위 내의 임의의 정수를 반환합니다.

<https://docs.microsoft.com/ko-kr/>

클래스 사용



Random 클래스

- (예제 5-2) Random 클래스를 사용한 임의의 실수 생성(교재 209p)
 - 코드5-2. Random 클래스를 사용한 임의의 실수 생성

```
01 static void Main(string[] args)
02 {
03     Random random = new Random();
04     Console.WriteLine(random.NextDouble());
05     Console.WriteLine(random.NextDouble());
06     Console.WriteLine(random.NextDouble());
07     Console.WriteLine(random.NextDouble());
08 }
09
10
11
12
13
```

0.0과 1.0 사이의 난수(임의의 수)
를 반환

실행 결과

```
0.385116411086692
0.294557459789588
0.83222587864484
0.910290311514535
```



(참고) 원하는 범위의 실수 난수 생성

- 0.0부터 10.0 사이의 난수를 구할 때는 다음과 같이 10을 곱하기

```
static void Main(string[] args)
{
    Random random = new Random();
    Console.WriteLine(random.NextDouble() * 10);
    Console.WriteLine(random.NextDouble() * 10);
    Console.WriteLine(random.NextDouble() * 10);
    Console.WriteLine(random.NextDouble() * 10);
}
```

```
3.66736765190371
5.76678149205017
0.73619941749433
4.62893074593923
```

클래스 사용



Random 클래스

- Random 클래스 메서드 사용
 - 인스턴스(random)를 생성한 후 인스턴스 뒤에 .을 찍고 메서드를 사용
 - random.Next(10)
 - random.Next(10, 100)
 - random.NextDouble()
 - **인스턴스 멤버**: 인스턴스 뒤에 .을 찍고 사용하는 멤버라 함. 이때,
 - 해당 멤버가 변수이면, **인스턴스 변수**
 - 해당 멤버가 메서드이면, **인스턴스 메서드**
 - 해당 멤버가 속성이면, **인스턴스 속성**이라고 함!

클래스 사용



List 클래스

- List 클래스

- 배열과 유사하나, 크기가 가변적인 배열을 만듦

- 배열은 크기가 고정

- (예) `int[] intArray = new int[10];` 은 10개의 공간 갖게 됨

- `long[] longArray = new long[10];`

- `string[] stringArray = new string[10];`

- 제네릭(Generic): 클래스 선언 시 어떤 자료형인지 알려주기 위해 사용

- 클래스명 뒤에 `< >`로 적용

- (예시) List 클래스의 인스턴스 생성

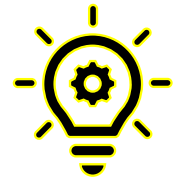
```
static void Main(string[] args)
{
    // 변수를 선언합니다.
    List<int> list = new List<int>();
}
```

ClassUses

클래스 사용



(참고) List 클래스 참조 추가



- 참조 오류

```
List<int> list = new List<int>();
```

CS0246: 'List<>' 형식 또는 네임스페이스 이름을 찾을 수 없습니다. using 지시문 또는 세요.
잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

- 참조오류 보조기능(마우스 가져다 대기)

```
// 변수를 선언합니다.  
List<int> list = new List<int>();
```

using System.Collections.Generic;

System.Collections.Generic.List
'List'에 대한 클래스 생성(C)
새 형식 생성(T)...

- (**Ctrl** + **.**) 단축키 누르거나 파란 글상자 선택)

```
List<int> list = new List<int>();  
list.
```

~

- ★ Add
- ★ Count
- ★ Clear
- ★ AddRange
- ★ ForEach
- Add
- AddRange
- AsReadOnly

void List<int>.Add(int item)
개체를 List<T>의 끝 부분에 추가합니다.
★ 이 컨텍스트를 기반으로 한 IntelliCode 제안

클래스 사용



List 클래스

- (예제 5-3) 리스트 요소 추가(교재 213p)

- 코드5-6. 리스트 요소 추가

```
01 static void Main(string[] args)
02 {
03     // 변수를 선언합니다.
04     List<int> list = new List<int>();
05
06     // 리스트에 요소를 추가합니다.
07     list.Add(52);
08     list.Add(273);
09     list.Add(32);
10     list.Add(64);
11
12     // 반복을 수행합니다.
13     foreach (var item in list)
14     {
15         Console.WriteLine("Count: " + list.Count + "\titem: " + item);
16     }
17 }
```

실행 결과

```
Count: 4      item: 52
Count: 4      item: 273
Count: 4      item: 32
Count: 4      item: 64
```

Add() 메서드 : 리스트에 요소 추가

Count() 메서드 : 리스트의 요소개수 가져옴

클래스 사용



List 클래스

실행 결과

- (예제 5-4) 리스트 요소 제거(교재 214p)
 - 코드5-8. 리스트 요소 제거

```
Count: 3      item: 273
Count: 3      item: 32
Count: 3      item: 64
```

```
01 static void Main(string[] args)
02 {
03     // 변수를 선언합니다.
04     List<int> list = new List<int>() { 52, 273, 32, 64 };
05
06     // 반복을 수행합니다.
07     list.Remove(52);
08
09     // 반복을 수행합니다.
10     foreach (var item in list)
11     {
12         Console.WriteLine("Count: " + list.Count + "\titem: " + item);
13     }
14
15 }
```

리스트 생성과 동시에 요소 추가 가능

Remove() 메서드 : 리스트에서 요소 제거

클래스 사용



Math 클래스

- Math 클래스
 - 수학과 관련된 변수 또는 메서드 제공
 - Math클래스는 인스턴스를 만들지 않고 클래스명.멤버 형태로 바로 사용
 - (예시) Math 클래스의 메서드



클래스 사용



Math 클래스

- Math 클래스
 - (예시) Math 클래스의 메서드

메서드 이름	설명
Abs(숫자)	절대 값을 구합니다.
Ceiling(숫자)	지정된 숫자보다 크거나 같은 최소 정수를 구합니다.
Floor(숫자)	지정된 숫자보다 작거나 같은 최대 정수를 구합니다.
Max(숫자, 숫자)	두 개의 매개변수 중에서 큰 값을 구합니다.
Min(숫자, 숫자)	두 개의 매개변수 중에 작은 값을 구합니다.
Round(숫자)	반올림합니다.

클래스 사용



Math 클래스

- (예제 5-5) Math 클래스 활용 (교재 215p)
 - 코드5-9. Math 클래스 활용

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine(Math.Abs(-52273));
04     Console.WriteLine(Math.Ceiling(52.273));
05     Console.WriteLine(Math.Floor(52.273));
06     Console.WriteLine(Math.Max(52, 273));
07     Console.WriteLine(Math.Min(52, 273));
08     Console.WriteLine(Math.Round(52.273));}
09
10
11
12
13
```

실행 결과

52273

53

52

273

52

52

클래스 사용



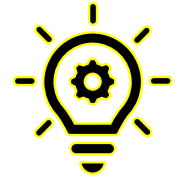
Math 클래스

- Math 클래스의 메서드 사용
 - 클래스 멤버: 클래스명 뒤에 .을 찍고 사용하는 멤버를 말함. 이때,
 - 해당 멤버가 변수이면, 클래스 변수
 - 해당 멤버가 메서드이면, 클래스 메서드
 - 해당 멤버가 속성이면, 클래스 속성이라고 함

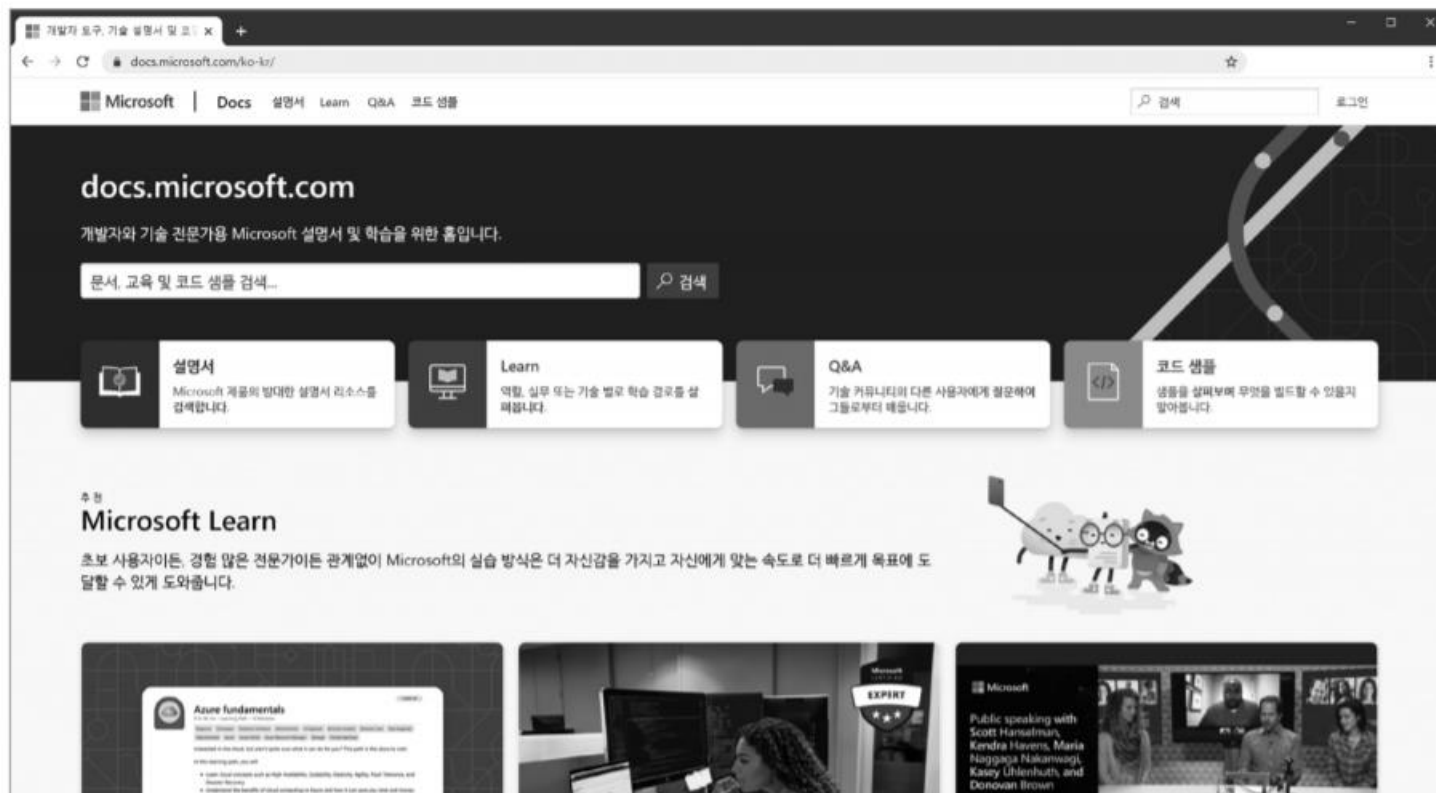
클래스 사용



(참고) 클래스 검색



- MSDN 온라인 문서 (<https://docs.microsoft.com/ko-kr/>)
(<https://docs.microsoft.com/en-us/>)

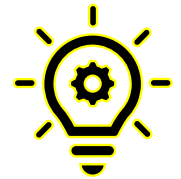


클래스 사용



(참고) 클래스 검색

- 구글 검색



Google

MSDN C# Math

전체 이미지 동영상 뉴스 쇼핑 더보기

검색결과 약 295,000개 (0.40초)

docs.microsoft.com > ... > .NET API 브라우저 > Math 클래스 (System) | Microsoft

C# 복사. `/// <summary> /// The following class`

정의 · 예제 · 필드 · 메서드

버전

.NET Core 3.1

검색

Math

> 필드

> 메서드

> MathF

> MemberAccessException

> Memory<T>

> MemoryExtensions

> MethodAccessException

MidpointRounding

> MissingFieldException

> MissingMemberException

> MissingMethodException

> ModuleHandle

> MTAThreadAttribute

> MulticastDelegate

> MulticastNotSupportedException

Math 클래스

네임스페이스: System

어셈블리: System.Runtime.Extensions.dll

삼각, 로그 및 기타 일반 수학 함수에 대한 상수 및 정적 메서드를 제공합니다.

C# 복사

`public static class Math`

상속 Object → Math

예제

다음 예제에서는 Math 클래스의 여러 수학 및 삼각 함수를 사용 하 여 사다리꼴의 내부 각도를 계산 합니다.

C# 복사

`/// <summary>
/// The following class represents simple functionality of the trapezoid.
/// </summary>
using System;`

윈도우

III. 클래스 생성

클래스 생성



클래스의 생성

- 클래스 생성 형식
 - class 키워드 사용
 - 클래스 이름의 첫 글자는 주로 대문자 사용

class 기본 형식

```
class [클래스 이름]
{

}
```

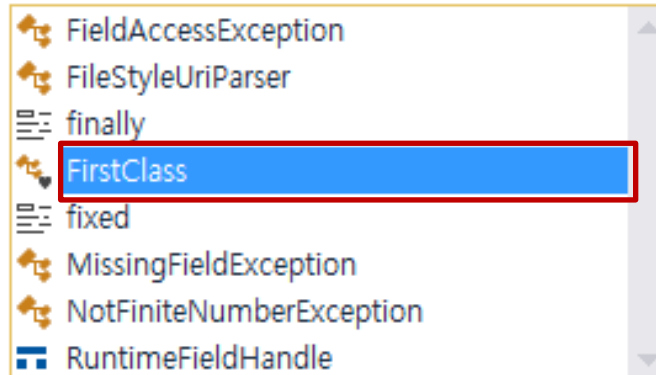
클래스 생성



하나의 파일에 여러 개의 클래스 생성

- 가장 쉬운 클래스 생성 방법
 - 파일 하나에 여러 개의 클래스를 생성하기
 - C# 콘솔 프로젝트 생성시 C# 파일(Program.cs)이 기본 생성되면 이 파일 내부에 클래스 추가
 - 생성된 클래스 확인하기

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



```
using System;

namespace ClassBasic
{
    class FirstClass
    {
    }

    class SecondClass
    {
    }

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

클래스 생성



클래스 내부에 클래스 생성

- 클래스 내부에 클래스 생성
 - 하나의 클래스 내부에 여러 개의 클래스 생성 가능 (예: Program 클래스에 추가)

```
class Program
```

```
{
```

```
    class FirstClass
```

```
    {
```

```
    }
```

```
    class SecondClass
```

```
    {
```

```
    }
```

```
    static void Main(string[] args)
```

```
    {
```

```
    }
```

```
}
```

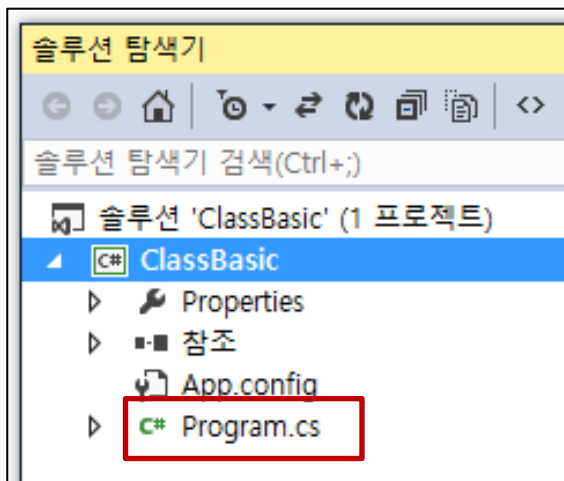
→ Program 클래스에 추가된 클래스는
Main 메서드 내부에서 사용 가능

클래스 생성



서로 다른 파일에 클래스 생성

- 서로 다른 파일에 클래스 생성
 - 파일 하나에 클래스 하나를 넣고, 파일 이름과 맞추어 만드는 것이 일반적
 - 파일의 이름과 클래스 이름이 달라도 상관없음
- 일반적으로 Program.cs 파일에는 하나의 Program 클래스가 존재함



```
using System;

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

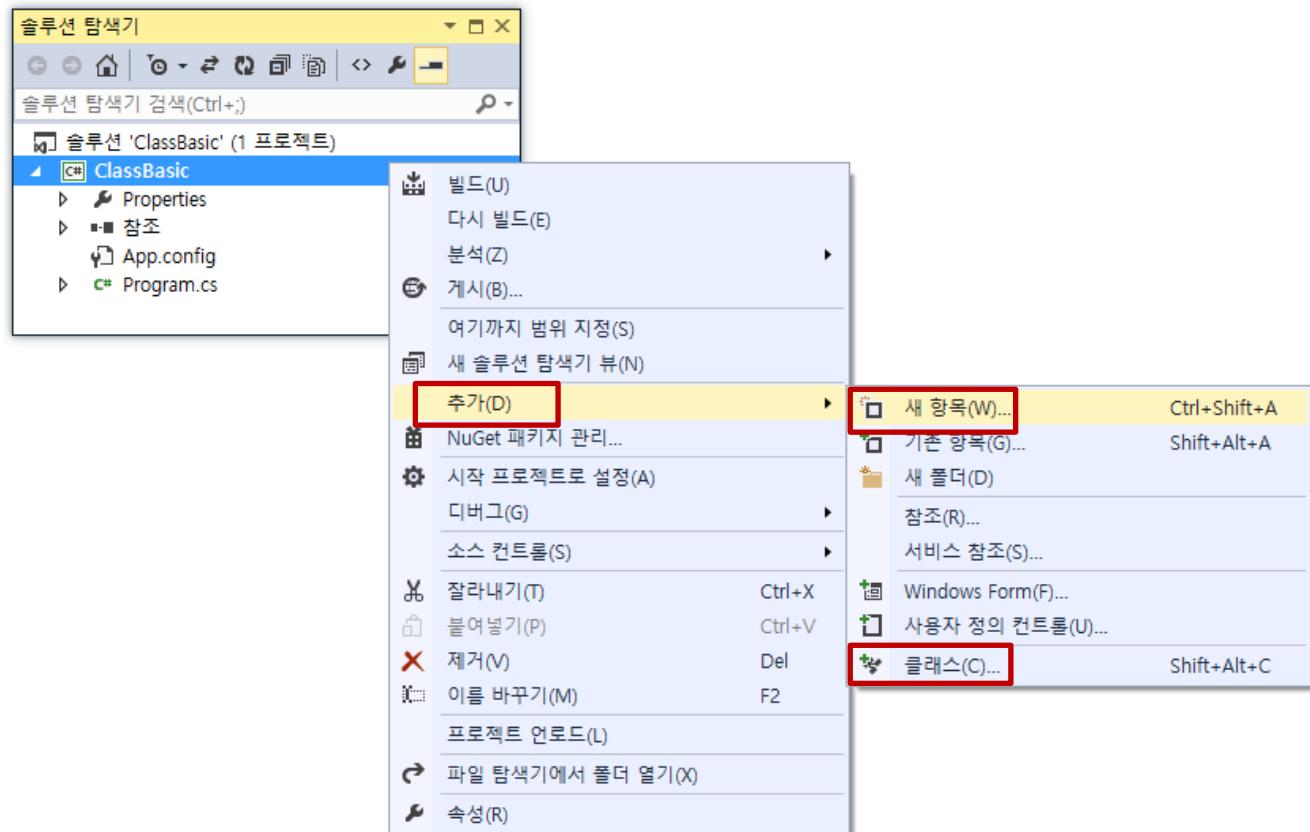
클래스 생성



서로 다른 파일에 클래스 생성

- 서로 다른 파일에 클래스 생성

- ① 마우스 오른쪽 클릭 [추가] - [새 항목] 또는 [클래스] 클릭
- 새항목 추가 메뉴



클래스 생성

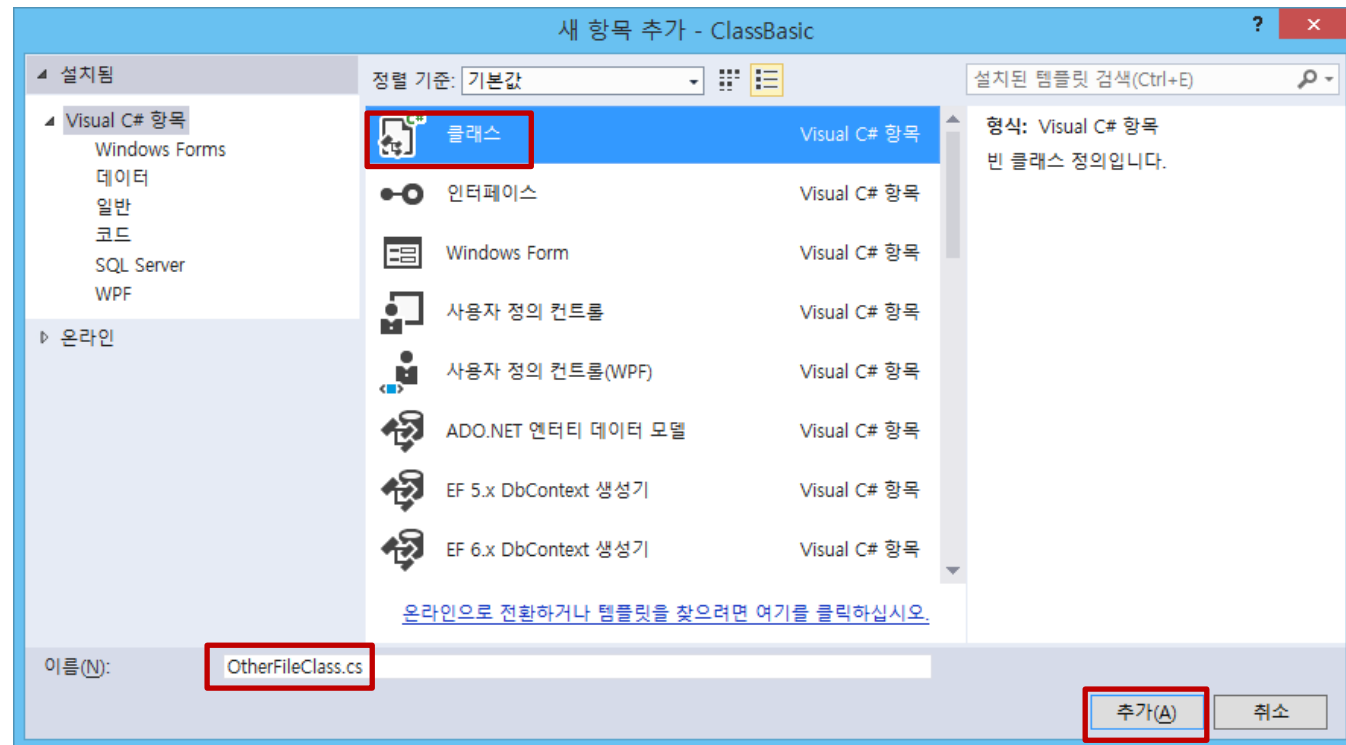


서로 다른 파일에 클래스 생성

- 서로 다른 파일에 클래스 생성

② 새 파일 대화상자 실행되면, 클래스 이름을 입력하고 [추가] 버튼 클릭

- 클래스 추가 방법

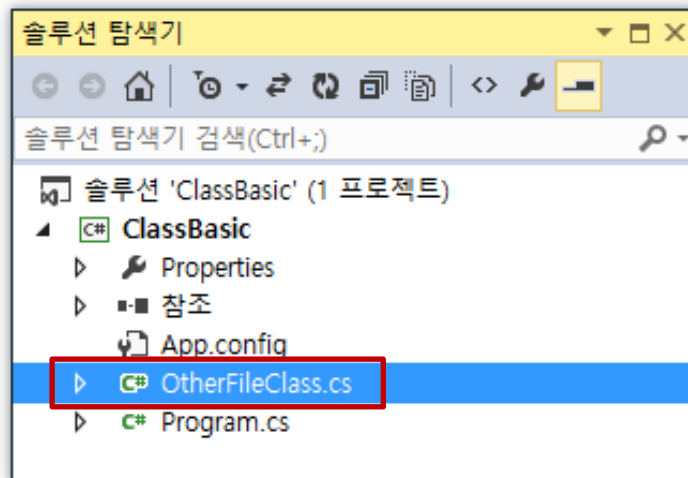


클래스 생성



서로 다른 파일에 클래스 생성

- 서로 다른 파일에 클래스 생성
 - 선언된 클래스 확인
 - 생성된 클래스 파일



클래스 생성



서로 다른 파일에 클래스 생성

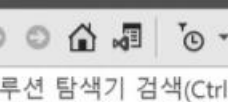
- 서로 다른 파일에 클래스 생성
 - 선언된 클래스 확인: 파일 이름과 같은 이름의 클래스가 선언됨을 확인

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ClassBasic
{
    class OtherFileClass
    {
    }
}
```



- ```
static void Main(string[] args)
{
 Product product = new Product();
}
```

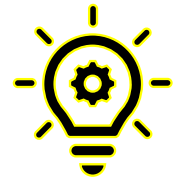
- 

- 
- The screenshot shows the Visual Studio Solution Explorer. The top bar is labeled '솔루션 탐색기' (Solution Explorer). Below it is a toolbar with icons for navigation and actions. The main area shows the project '솔루션 'ClassBasic' (1/1개 프로젝트)' (Solution 'ClassBasic' (1/1 project)). Under this project, there is a folder 'ClassBasic' containing four files: '종속성' (Dependencies), 'C# OtherFileClass.cs', 'Product.cs', and 'C# Program.cs'. The 'Product.cs' file is currently selected and highlighted.

# 클래스 생성



## (참고) 클래스 이름 충돌



- 클래스 이름 충돌: C#의 기본 클래스 Math와 별도로 선언한 클래스 Math가 충돌

```
class Program
{
 class Math { }

 static void Main(string[] args)
 {
 Console.WriteLine(Math.Abs(-10))
 }
}
```

```
참조 0개
class Program
{
 참조 1개
 class Math { }

 참조 0개
 static void Main(string[] args)
 {
 Console.WriteLine(Math.Abs(-10));
 }
}
```

CS0117: 'Program.Math'에는 'Abs'에 대한 정의가 포함되어 있지 않습니다.  
잠재적 수정 사항 표시 (Alt+Enter 또는 Ctrl+.)

- 클래스 이름 지정 시 특별한 목적이 없는 한 기존 클래스 이름과 다르게 선언

## IV. 클래스의 변수

# 클래스의 변수



## 인스턴스 변수

- 인스턴스 변수 생성 방법

- 인스턴스 변수는 [인스턴스].[변수이름]의 형태로 사용함
- 소문자로 시작하여 여러 개를 만들 수 있음
- 클래스 내부에 선언

---

[접근 제한자] [자료형] [이름]

---

- (예시) 인스턴스 변수 선언

```
class User
{
 public string name;
 public string password;
 public string address;
 public string phoneNumber;
 public DateTime regDate;
}
```

# 클래스의 변수



## 인스턴스 변수

- (예제 5-6) 인스턴스 변수 생성과 사용(교재 226p)
  - 코드5-12. 인스턴스 변수 생성과 사용

실행 결과

감자 : 2000원

```
01 namespace InstanceVariable
02 {
03 class Program
04 {
05 class Product
06 {
07 public string name;
08 public int price;
09 }
10
11 static void Main(string[] args)
12 {
13 Product product = new Product();
14
15 product.name = "감자";
16 product.price = 2000;
17
18 Console.WriteLine(product.name + " : " + product.price + "원");
19 }
20 }
21 }
```

Product 인스턴스 생성 후  
Product. 입력하고 자동완성기능 사용하면  
해당 인스턴스 변수 확인 가능(name, price)

# 클래스의 변수



## 인스턴스 변수

- 인스턴스 변수를 생성할 때 초기화
  - 인스턴스 변수 생성 시 초기화

---

```
class Product
{
 public string name = "default";
 public int price = 1000;
}
```

---

# 클래스의 변수



## 인스턴스 변수

- 인스턴스 변수를 생성할 때 초기화
  - (예시) 인스턴스 변수를 생성과 동시에 초기화

```
class Program
{
 class Product
 {
 public string name;
 public int price;
 }

 static void Main(string[] args)
 {
 Product productA = new Product() { name = "감자", price = 2000 };
 Product productB = new Product() { name = "고구마", price = 3000 };
 }
}
```



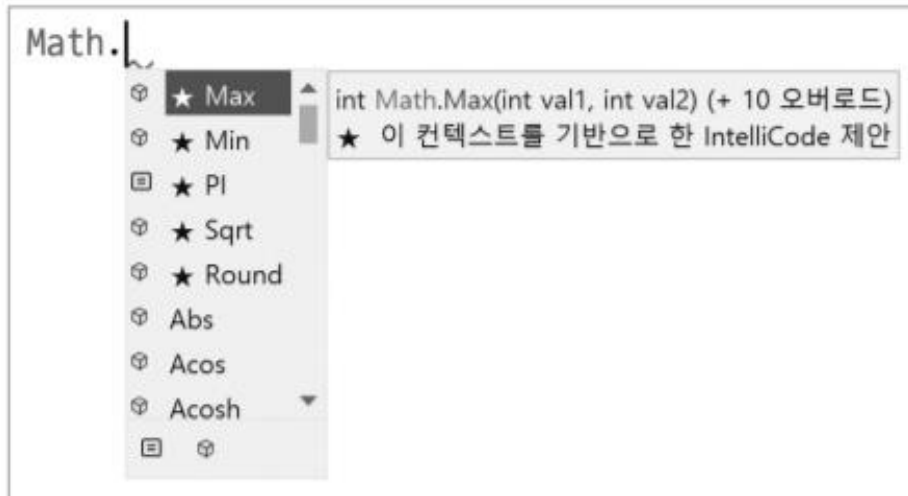
# 클래스의 변수



## 클래스 변수

- 클래스 변수와 클래스 메서드
    - 클래스 이름으로 곧바로 사용하는 변수와 메서드
    - 클래스 변수 생성 방법
      - static 키워드 사용
- [접근 제한자] static [자료형] [이름]

- Math 클래스의 클래스 메서드



# 클래스의 변수



## 클래스 변수

- (예제 5-7) 클래스 변수 생성과 사용(교재 229p)
  - MyMath 클래스에 클래스 변수 PI를 생성하고 사용하기
  - 코드5-7. 클래스 변수 생성과 사용

```
01 namespace ClassVariable
02 {
03 class Program
04 {
05 class MyMath
06 {
07 public static double PI = 3.141592;
08 }
09
10 static void Main(string[] args)
11 {
12 Console.WriteLine(MyMath.PI);
13 }
14 }
15 }
```

실행 결과

3.141592

## V. 추상화



## 추상화

- 클래스 기반의 객체 지향 프로그래밍 언어의 특징
  - 추상화, 캡슐화, 상속, 다형성
- 추상화
  - 프로그램에 사용되는 핵심적인 부분을 추출하는 것
  - (예시) 학생 추상화

```
class Student
{
 public string id;
 public string name;
 public int grade;
 public string major;
 public DateTime birthday;

 /* 계속해서 생각해보세요. */
}
```

## **VI. 실습 및 과제**

# 과제



## 자율학습

- 세부 과제
  - (교재 p. 203~231) 기본예제 5-1 ~ 5-7
  - (교재 p. 232~237) 응용예제 5-1 ~ 5-2
    - 자율적으로 학습
- 주의사항
  - 금주는 과제 제출 없습니다. 각자 복습하면서 학습내용을 이해해보세요.



# 감사합니다

mnshim@sungkyul.ac.kr

