Chapter 05 클래스 기본

Section 01 클래스 개요(1)

■ 사용자 정의 자료형

- 클래스 : 객체 지향 언어, 원하는 새로운 자료형 정의
- 예

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

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

Section 01 클래스 개요(2)

- 메서드 사용한 코드

```
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();
}
```

Section 01 클래스 개요(3)

■ 클래스와 인스턴스

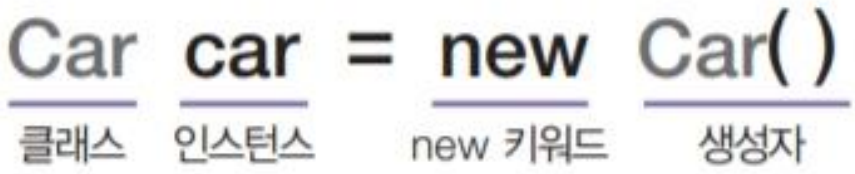


그림 5-1 클래스와 관련된 기본 용어

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

Section 02 클래스 사용(1)

■ Random 클래스

- 임의의 숫자 생성시 사용
- 인스턴스 생성 방법 : Random random = new Random ( )

그림 5-2 Random 클래스의 인스턴스에서 사용할 수 있는 메서드

Section 02 클래스 사용(2)

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

/5장/RandomInteger

```
static void Main(string[] args)
{
    Random random = new Random();
    Console.WriteLine(random.Next(10, 100));
    Console.WriteLine(random.Next(10, 100));
    Console.WriteLine(random.Next(10, 100));
    Console.WriteLine(random.Next(10, 100));
    Console.WriteLine(random.Next(10, 100));
}
```

실행 결과

```
86
48
76
49
```

Section 02 클래스 사용(2)

- **기본예제 5-2** Random 클래스를 사용한 임의의 실수 생성(교재 209p)

/5장/RandomDouble

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

실행 결과

```
0.385116411086692
0.294557459789588
0.83222587864484
0.910290311514535
```

■ 원하는 범위의 실수 난수 생성

```
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);
}
```

실행 결과

```
3.66736765190371
5.76678149205017
0.73619941749433
4.62893074593923
```

Section 02 클래스 사용(3)

■ List 클래스

- 배열과 유사 but, 크기가 가변적인 배열을 만들 수 있음
- 제네릭Generic : 클래스 선언 시 어떤 자료형인지 알려주는 것

```
// 배열 생성
int[] intArray = new int[10];
long[] longArray = new long[10];
string[] stringArray = new string[10];
```

코드 5-5 List 클래스의 인스턴스 생성 /5장/ClassUses

```
01 static void Main(string[] args)
02 {
03     // 변수를 선언합니다.
04     List<int> list = new List<int>();
05 }
```

List 클래스 참조 추가

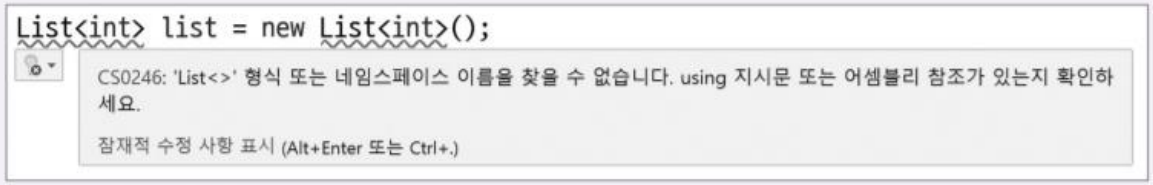


그림 5-7 참조 오류

- 마우스 가져다 대기

그림 5-8 참조 관련 오류 보조 기능

- Ctrl + . 단축키를 누르거나

  [그림 5-10] 파란 글상자 선택

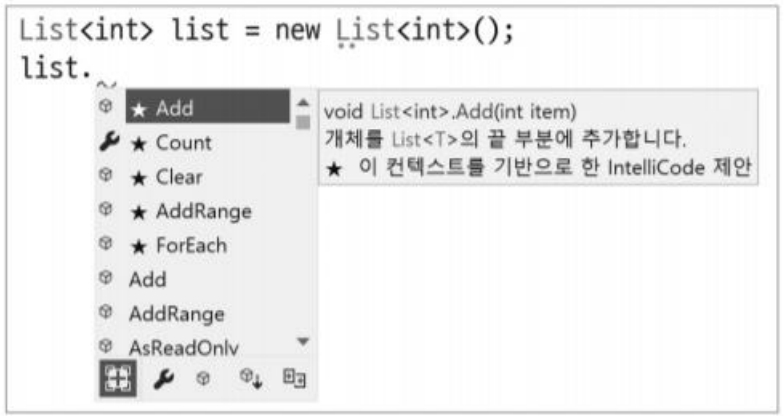


그림 5-9 List 인스턴스의 자동 완성 기능

Section 02 클래스 사용(4)

- **기본예제 5-3** 리스트 요소 추가(교재 213p) /5장/ListElementAdd

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

    // 리스트에 요소를 추가합니다.
    list.Add(52);
    list.Add(273);
    list.Add(32);
    list.Add(64);

    // 반복을 수행합니다.
    foreach (var item in list)
    {
        Console.WriteLine("Count: " + list.Count + "₩titem: " + item);
    }
}
```

실행 결과

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

Section 02 클래스 사용(4)

- **기본예제 5-4** 리스트 요소 제거(교재 214p) /5장/ListElementRemove

```
static void Main(string[] args)
{
    // 변수를 선언합니다.
    List<int> list = new List<int>() { 52, 273, 32, 64 };

    // 반복을 수행합니다.
    list.Remove(52);

    // 반복을 수행합니다.
    foreach (var item in list)
    {
        Console.WriteLine("Count: " + list.Count + "₩titem: " + item);
    }
}
```

실행 결과

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

RemoveAt(int index) : 지정한 인덱스에서 List<T> 항목을 제거
RemoveRange(int index, int count) : 지정한 인덱스에서 지정한 개수 만큼 항목 제거

■ List 인스턴스 생성과 동시에 요소 추가

코드 5-7 List 인스턴스 생성과 동시에 요소 추가

```
static void Main(string[] args)
{
    // 변수를 선언합니다.
    List<int> list = new List<int>() { 52, 273, 32, 64 };

    // 반복을 수행합니다.
    foreach (var item in list)
    {
        Console.WriteLine("Count: " + list.Count + "\titem: " + item);
    }
}
```

Section 02 클래스 사용(5)

■ Math 클래스

- 수학과 관련된 변수 또는 메서드 제공
- 인스턴스를 만들지 않고 사용

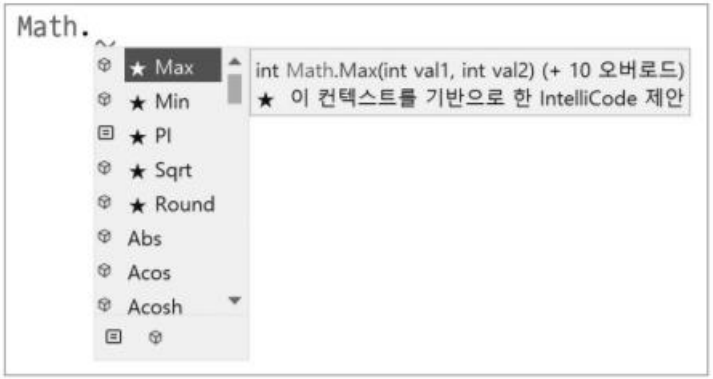


그림 5-10 클래스 메서드

클래스 멤버 : 클래스 이름 뒤에 점을 찍고 사용하는 멤버

- 클래스 변수
- 클래스 메서드
- 클래스 속성

표 5-1 Math 클래스의 메서드

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

Section 02 클래스 사용(6)

- **기본예제 5-5** Math 클래스 활용 (교재 216p) /5장/MathBasic

```
static void Main(string[] args)
{
    Console.WriteLine(Math.Abs(-52273));
    Console.WriteLine(Math.Ceiling(52.273));
    Console.WriteLine(Math.Floor(52.273));
    Console.WriteLine(Math.Max(52, 273));
    Console.WriteLine(Math.Min(52, 273));
    Console.WriteLine(Math.Round(52.273));
}
```

실행 결과

```
52273
53
52
273
52
52
```

표 5-1 Math 클래스의 메서드

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

Section 02 클래스 사용(7)

- **클래스 검색**
  - **https://docs.microsoft.com/ko-kr/documentation**

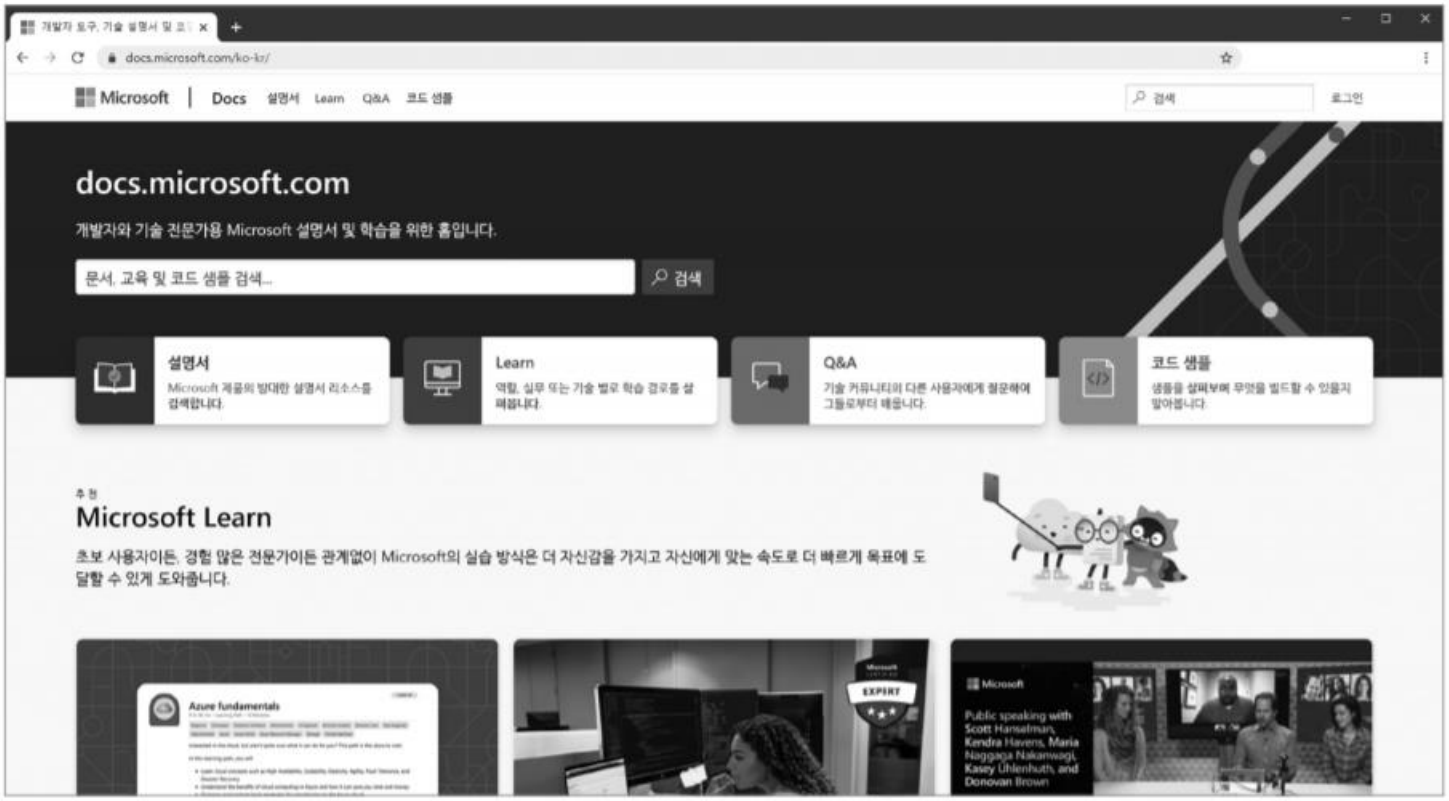


그림 5-11 마이크로소프트 기술 문서

Section 03 클래스 생성(1)

- 예

```
class [클래스 이름]
{

}
```

Section 03 클래스 생성(2)

■ 하나의 파일에 여러 개의 클래스 생성

- 클래스를 생성 가장 쉬운 방법 : 파일 하나에 여러 개의 클래스를 생성하는 것
- C# 콘솔 프로젝트를 생성하면 Program.cs 파일 기본 생성

```
using System;

namespace ClassBasic
{
    class FirstClass
    {
    }

    class SecondClass
    {
    }

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

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

FirstClass — class ClassBasic.FirstClass
FirstChanceExceptionEventArgs — System.Runtime.ExceptionServices
FirstDayOfWeek — Microsoft.VisualBasic
FirstWeekOfYear — Microsoft.VisualBasic

그림 5-14 생성된 클래스의 확인

Section 03 클래스 생성(3)

■ 클래스 내부에 클래스 생성

```
class Program
{
    class FirstClass
    {
    }

    class SecondClass
    {
    }

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

Section 03 클래스 생성(4)

■ 서로 다른 파일에 클래스 생성

- 파일 하나에 클래스 하나를 넣고, 파일의 이름과 맞추어 만드는 것이 일반적
- 파일의 이름과 클래스 이름이 달라도 상관없음

① 마우스 오른쪽 클릭 [추가] - [새 항목](또는 [New Item]) 또는 [클래스] 클릭

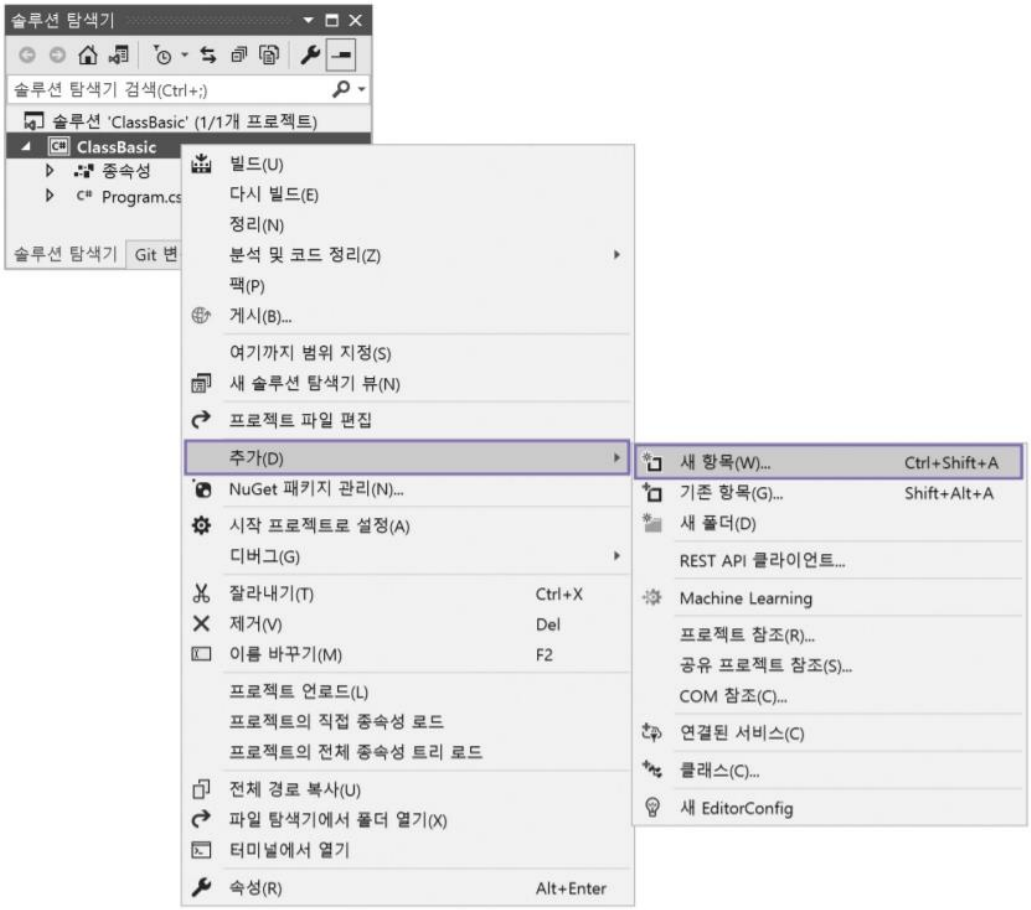


그림 5-16 새 항목 추가 메뉴

Section 03 클래스 생성(5)

② 새 파일 대화상자 실행, 클래스 이름 입력, [추가](또는 [Add]) 버튼 클릭

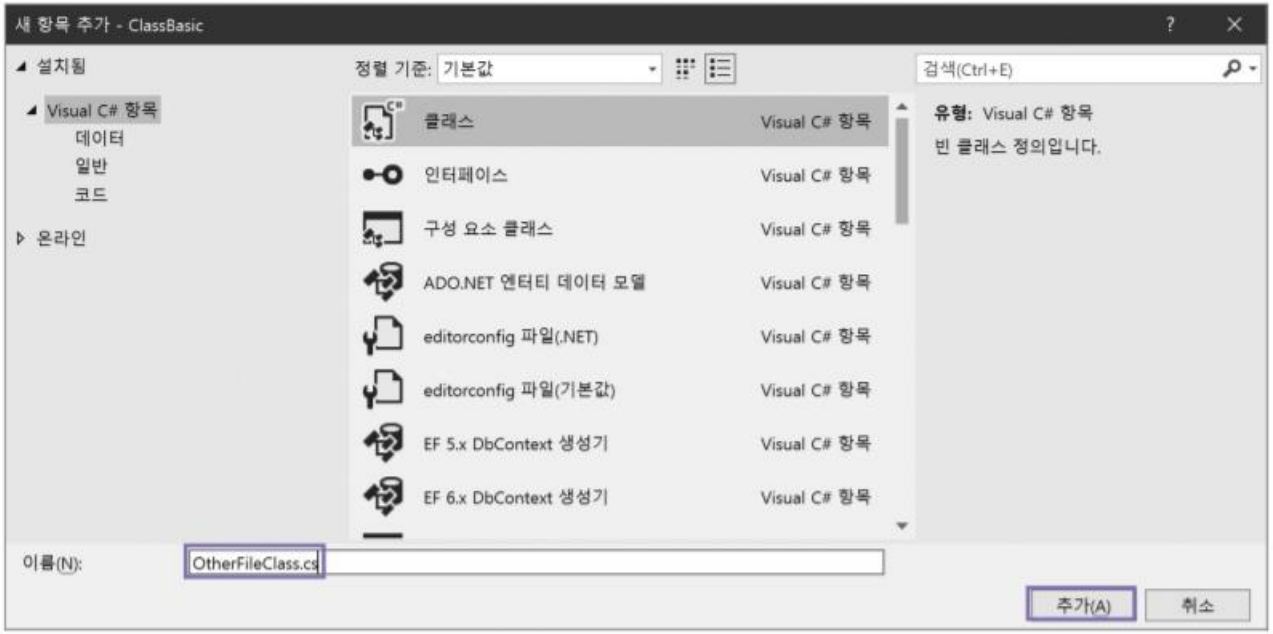


그림 5-17 클래스 추가 방법

Section 03 클래스 생성(6)

- 선언된 클래스 확인

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

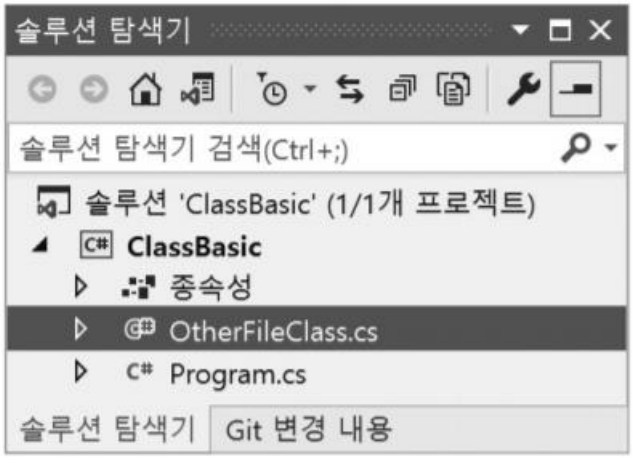


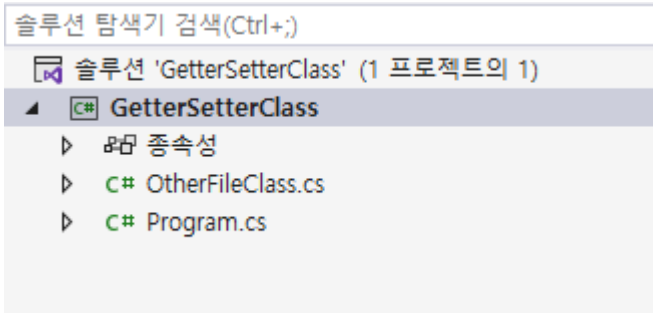
그림 5-18 생성된 클래스 파일

Section 03 클래스 생성(6) – 예)

```
using GetterSetterClass;

참조 0개
internal class Program
{
    참조 0개
    private static void Main(string[] args)
    {
        //Solution box = new Solution(-5, -10);
        OtherFileClass box = new OtherFileClass(-5, -10);
        Console.WriteLine("넓이 : " + box.Area());

        box.width = 2;
        box.height = 4;
        Console.WriteLine("넓이 : " + box.Area());
    }
}
```



NOTE(1)

■ 클래스 파일 빠르게 생성하기

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

- 발생된 오류에 마우스 대서 뜨는 붉은색 상자 클릭 또는 Ctrl + . 단축키 클릭

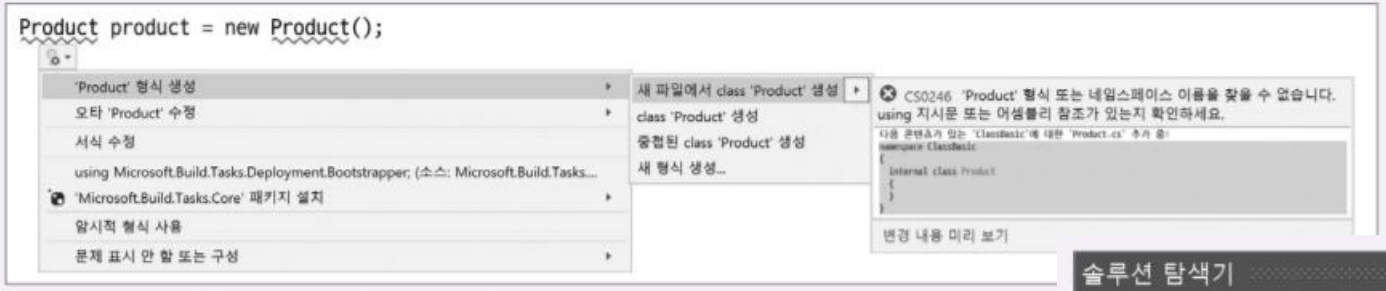


그림 5-19 빠른 클래스 파일 생성 방법

- [클래스 생성](또는 [Create Class]) 버튼 클릭

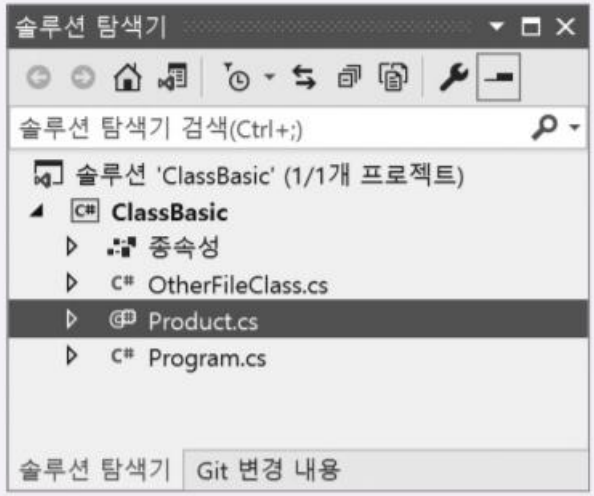


그림 5-20 생성된 클래스 파일 확인

NOTE(2)

■ 클래스 이름 충돌

코드 5-10 클래스 이름 충돌 /5장/NameCollision

```
class Program
{
    class Math { }

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

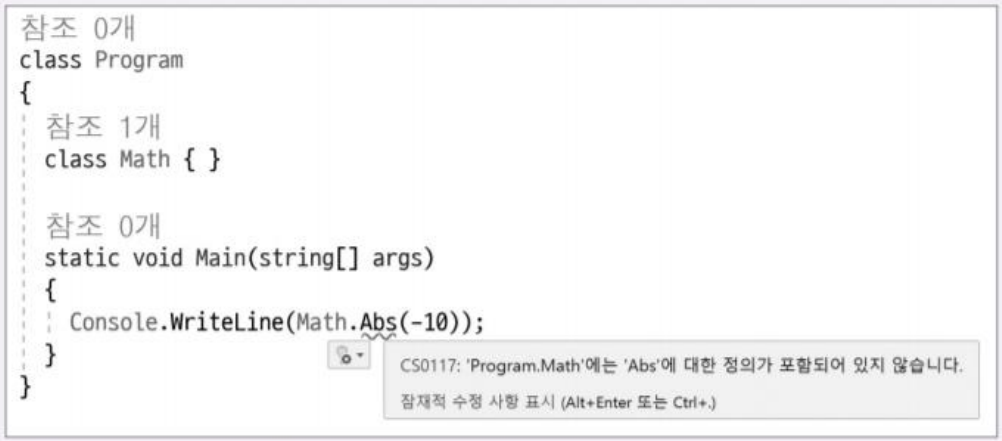


그림 5-21 클래스 이름 충돌

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

Section 04 C#에서 변수(인스턴스 변수, 클래스 변수, 지역 변수)

■ 선언 위치에 따른 변수의 종류

```
public class test
{
        int iv;             // 인스턴스 변수
        static int cv;      // 클래스 변수
        void method()
        {
                int lv;     // 지역 변수
        }
}
```

변수의 종류	선언위치	생성시기(메모리 할당 시기)
클래스 변수	클래스 영역	클래스가 메모리에 올라갈 때
인스턴스 변수		인스턴스가 생성될 때
지역 변수	클래스 이외의 영역 (메서드, 생성자, 초기화 블럭)	변수 선언문이 수행 되었을 때

■ 인스턴스 변수

- 인스턴스 변수는 인스턴스(객체)가 생성될 때 새로운 메모리 공간에 생성
- 인스턴스 변수의 값을 읽어오거나 저장하려면 인스턴스(객체)를 먼저 생성해야 한다.
- 인스턴스(객체) 별로 다른 값을 가질 수 있으므로, 각각의 인스턴스(객체)마다 고유의 값을 가져야할 때는 인스턴스 변수로 선언한다.

클래스는 참조형식으로 생성된 객체는 해당 메모리에 대한 참조만 갖는다.

b=a와 같이 객체 a가 다른 객체 b에 할당되면, 새 객체인 b도 동일한 메모리 참조

➔ 한 객체의 값을 변경하게 되면 다른 객체의 값 또한 변경됨

Section 04 C#에서 변수(인스턴스 변수, 클래스 변수, 지역 변수)

■ 클래스 변수

- 클래스 내부에 static 키워드와 함께 정의! 즉, 인스턴스 변수에 static만 붙여주면 됨
- 인스턴스 변수는 각각 고유한 값을 가지지만 클래스 변수는 모든 인스턴스(객체)가 공통된 값을 공유
- 한 클래스의 모든 인스턴스(객체)들이 공통적인 값을 가져야할 때 클래스 변수로 선언
- 클래스가 로딩될 때 생성되어(메모리에 딱 한번만 올라감) 프로그램이 종료 될 때 까지 유지
- 클래스 변수는 public 을 붙이면 같은 프로그램 내에서 어디서든 접근할 수 있는 전역 변수가 됨
- 인스턴스 변수의 접근법과 다르게 인스턴스를 생성하지 않고 *클래스이름.클래스변수명* 을 통해서 접근

Section 04 C#에서 변수(인스턴스 변수, 클래스 변수, 지역 변수)

■ 지역 변수

- 메서드 내에서 선언되며 메서드 내에서만 사용할 수 있는 변수
- 메서드가 실행될 때 메모리를 할당 받으며 메서드가 끝나면 소멸되어 사용할 수 없게 됨

■ 초기화

- 인스턴스 변수는 생성자를 통해 초기화
- 클래스 변수는 정적 생성자(static constructor)를 통해 초기화

■ 생명주기

- 인스턴스 변수는 해당 인스턴스(객체)가 소멸할 때 함께 소멸
- 클래스 변수는 프로그램이 종료될 때까지 유지

Section 04 클래스의 변수(1)

■ 인스턴스 변수

- 인스턴스 변수 생성 방법 : 소문자로 시작
- 사용 형태 : 인스턴스.변수이름

```
[접근 제한자] [자료형] [이름]
```

- 예

코드 5-11 인스턴스 변수 선언 /5장/InstanceVariables

```
01 class User
02 {
03     public string name;
04     public string password;
05     public string address;
06     public string phoneNumber;
07     public DateTime regDate;
08 }
```

Section 04 클래스의 변수(2)

- **기본예제 5-6** 인스턴스 변수 생성과 사용(교재 226p) /5장/InstanceVariable

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

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

        product.name = "감자";
        product.price = 2000;

        Console.WriteLine(product.name + " : " + product.price + "원");
    }
}
```

실행 결과

```
감자 : 2000원
```

Section 04 클래스의 변수(3)

- 인스턴스 변수를 생성할 때 초기화

코드 5-14 인스턴스 변수를 생성할 때 초기화 /5장/InstanceVariables

```
01 class Product
02 {
03     public string name = "default";
04     public int price = 1000;
05 }
```

Section 04 클래스의 변수(4)

- 인스턴스 변수를 생성할 때 초기화

코드 5-15 인스턴스 변수를 생성과 동시에 초기화 /5장/InstanceVariables

```
01 class Program
02 {
03     class Product
04     {
05         public string name;
06         public int price;
07     }
08
09     static void Main(string[] args)
10     {
11         Product productA = new Product() { name = "감자", price = 2000 };
12         Product productB = new Product() { name = "고구마", price = 3000 };
13     }
14 }
```

Section 04 클래스의 변수(5)

■ 클래스 변수

- 클래스 변수와 클래스 메서드 : 클래스 이름으로 곧바로 사용하는 변수와 메서드

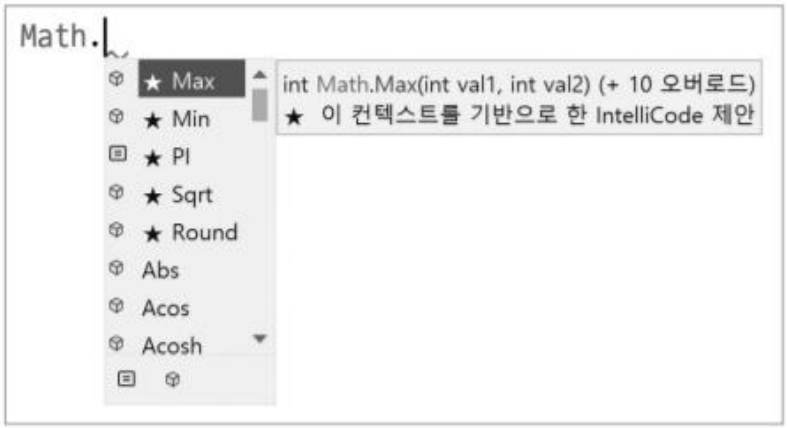


그림 5-23 Math 클래스의 클래스 메서드

- 클래스 변수 생성 방법

```
[접근 제한자] static [자료형] [이름]
```

Section 04 클래스의 변수(6)

- **기본예제 5-7** 클래스 변수 생성과 사용(교재 229p) /5장/ClassVariable

```
class Program
{
    class MyMath
    {
        public static double PI = 3.141592;
    }

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

실행 결과

```
3.141592
```

Section 05 추상화

- 클래스 기반의 객체 지향 프로그래밍 언어의 특징
  - 추상화, 캡슐화, 상속, 다형성
  - 추상화 : 프로그램에 사용되는 핵심적인 부분을 추출하는 것

코드 5-17 학생 추상화 /5장/InstanceVariables

```
01 class Student
02 {
03     public string id;
04     public string name;
05     public int grade;
06     public string major;
07     public DateTime birthday;
08
09     /* 계속해서 생각해보세요. */
10 }
```

Section 06 함께 하는 응용 예제(1)

- **응용예제 5-1** 모델 클래스와 List 클래스(교재 233p) /5장/ModelClassWithList

```
class Program
{
    class Student
    {
        public string name;
        public int grade;
    }

    static void Main(string[] args)
    {
        List<Student> list = new List<Student>();
        list.Add(new Student() { name = "윤인성", grade = 1 });
        list.Add(new Student() { name = "연하진", grade = 2 });
        list.Add(new Student() { name = "윤아린", grade = 3 });
        list.Add(new Student() { name = "윤명월", grade = 4 });
        list.Add(new Student() { name = "구지연", grade = 1 });
        list.Add(new Student() { name = "김연화", grade = 2 });

        foreach (var item in list)
        {
            Console.WriteLine(item.name + " : " + item.grade);
        }
    }
}
```

실행 결과

```
윤인성 : 1
연하진 : 2
윤아린 : 3
윤명월 : 4
구지연 : 1
김연화 : 2
```

Section 06 함께 하는 응용 예제(2)

- **응용예제 5-2** List 클래스 요소 제거와 역 반복문(교재 234p)

/5장/ElementRemoveWithReverse

표 5-2 리스트 요소 제거에 사용하는 메서드

메서드 이름	설명
Remove(object element)	특정 요소를 리스트에서 제거합니다(객체를 지정).
RemoveAt(int index)	특정 위치에 있는 요소를 리스트에서 제거합니다(인덱스를 지정).

① foreach 반복문으로 요소 제거

② for 반복문으로 요소 제거

실행 결과

```
윤인성 : 1
윤아린 : 3
구지연 : 1
```

③ 역 for 반복문을 사용한 요소 제거

실행 결과

```
윤인성 : 1
구지연 : 1
```

Section 06 함께 하는 응용 예제(2)

- **응용예제 5-2** List 클래스 요소 제거와 역 반복문(교재 234p)

/5장/ElementRemoveWithReverse

```
class Program
{
    class Student
    {
        public string name;
        public int grade;
    }

    static void Main(string[] args)
    {
        List<Student> list = new List<Student>();
        list.Add(new Student() { name = "윤인성", grade = 1 });
        list.Add(new Student() { name = "연하진", grade = 2 });
        list.Add(new Student() { name = "윤아린", grade = 3 });
        list.Add(new Student() { name = "윤명월", grade = 4 });
        list.Add(new Student() { name = "구지연", grade = 1 });
        list.Add(new Student() { name = "김연화", grade = 2 });

        for (int i = list.Count - 1; i >= 0; i--)
        {
            if (list[i].grade > 1)
            {
                list.RemoveAt(i);
            }
        }

        foreach (var item in list)
        {
            Console.WriteLine(item.name + " : " + item.grade);
        }
    }
}
```

■ 프로젝트 생성

- [파일]-[새로만들기(N)]-[프로젝트(P)] 클릭

그림 5-24(1) 새 프로젝트 대화상자

Section 07 윈도 폼: 윈도 폼 기본 익히기(2)

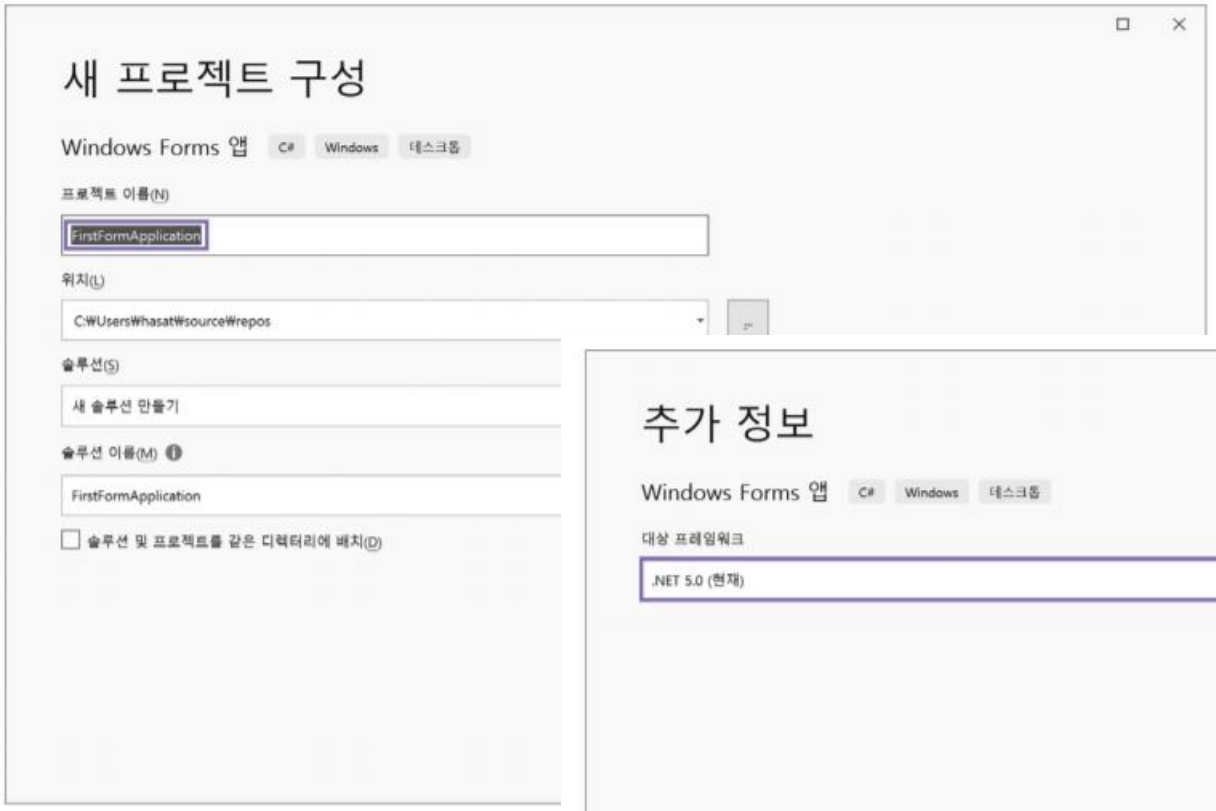


그림 5-24(2) 프로젝트 이름 지정

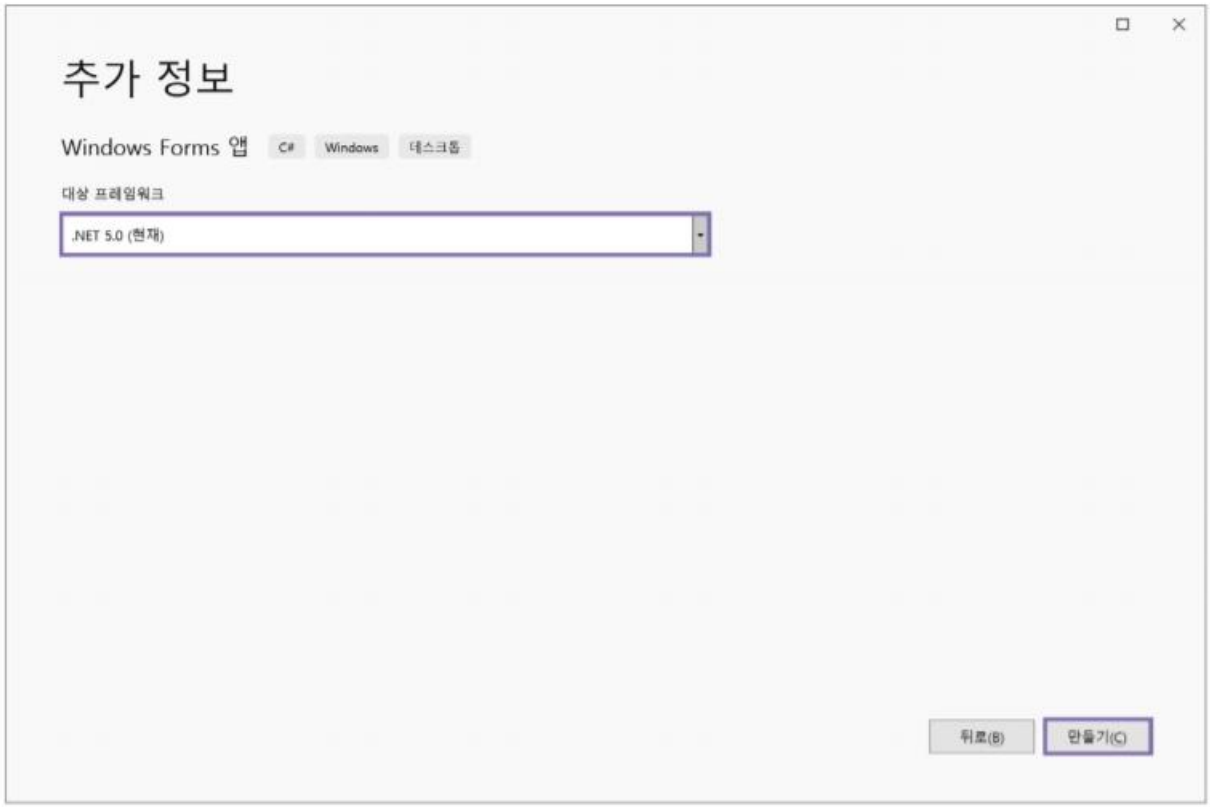


그림 5-24(3) 대상 프레임워크 선택 화면

Section 07 윈도 폼: 윈도 폼 기본 익히기(3)

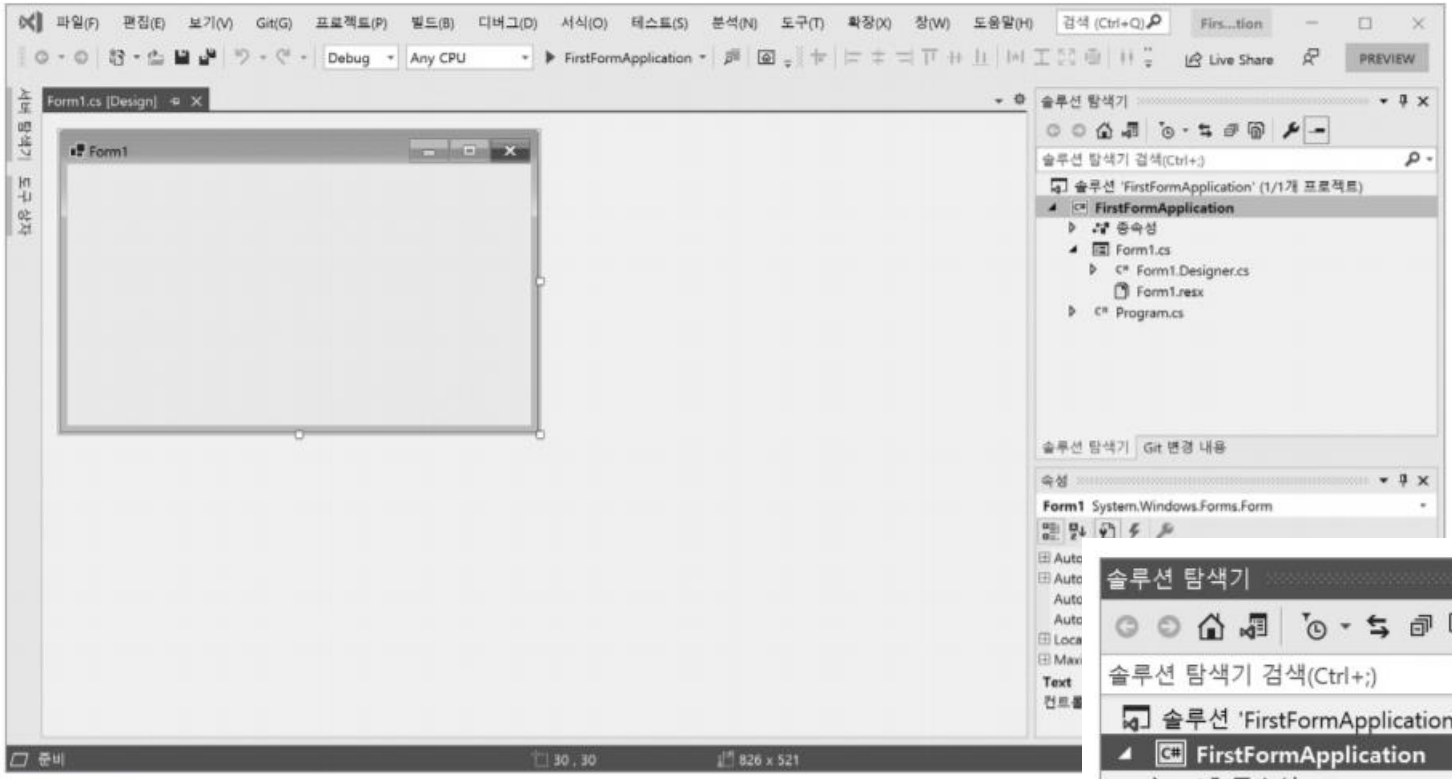


그림 5-25 생성된 프로젝트

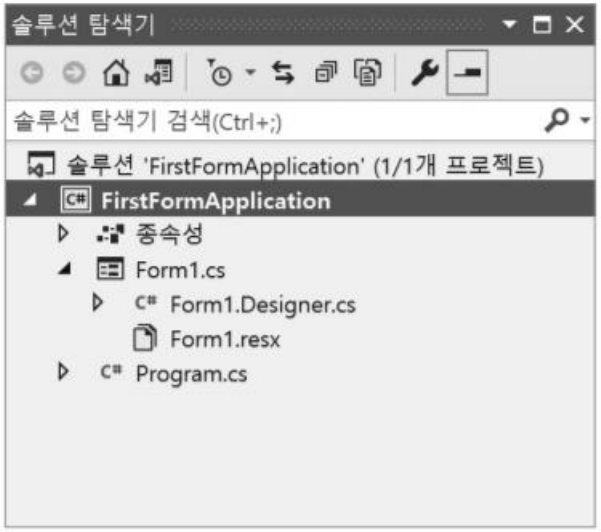


그림 5-27 윈도 폼 응용 프로그램 프로젝트의 기본 구조

Section 07 윈도 폼: 윈도 폼 기본 익히기(4)

- Form1 클래스

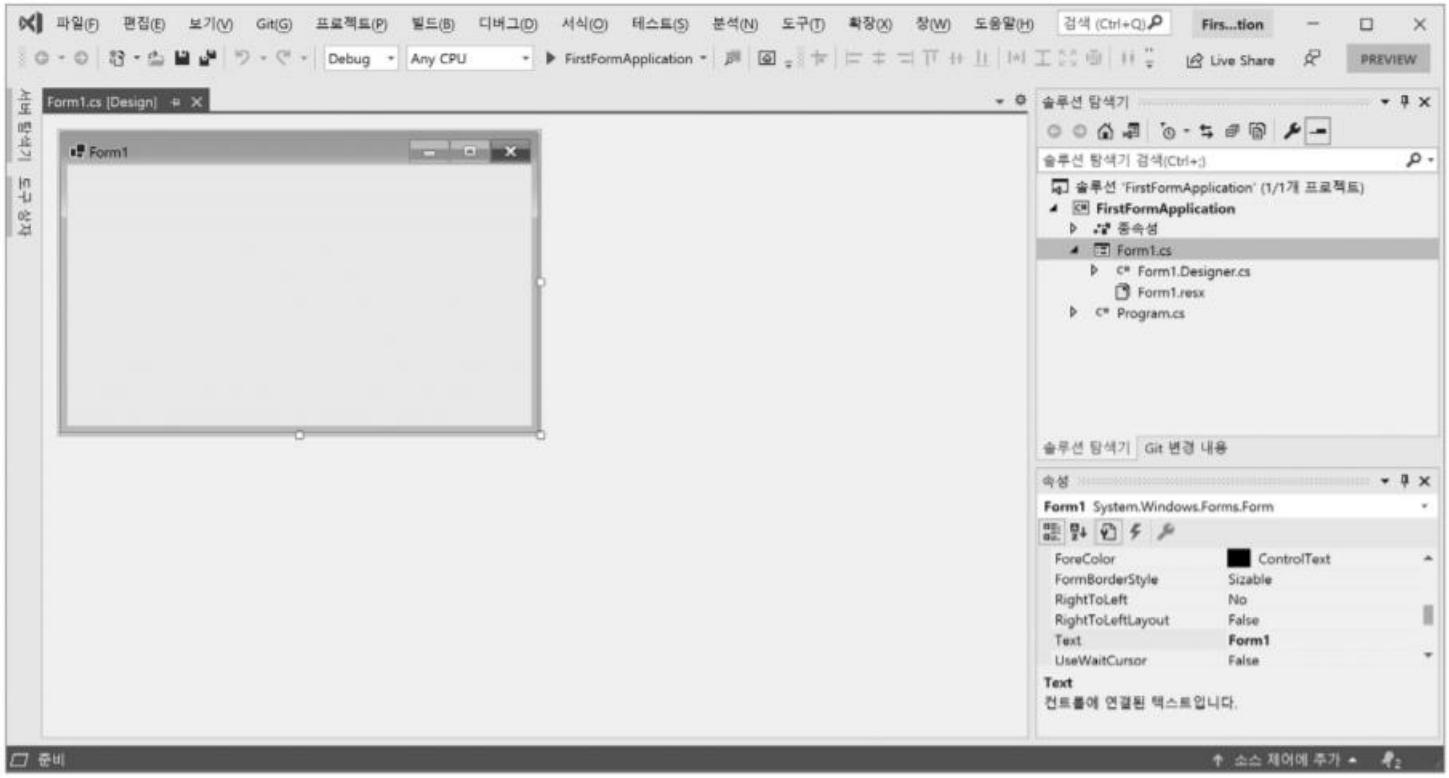


그림 5-28 Form1.cs 파일을 열었을 때의 화면

Section 07 윈도 폼: 윈도 폼 기본 익히기(5)

- Form1.cs 파일에서 마우스 오른쪽 버튼 클릭, [코드 보기] 클릭

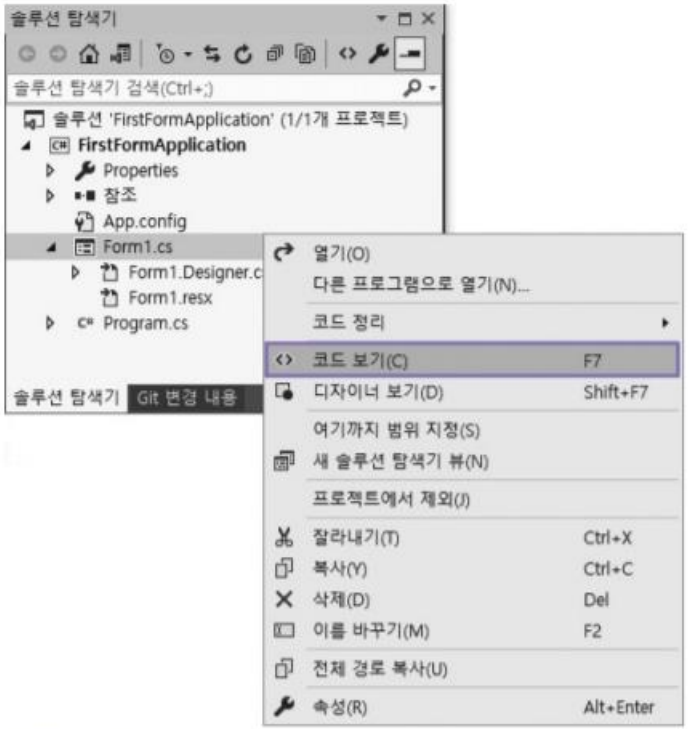


그림 5-29 Form1.cs 파일의 코드 보기

Section 07 윈도 폼: 윈도 폼 기본 익히기(6)

- Form1.cs 파일(코드 작성)

코드 5-26 Form1.cs 파일

```
01 using System;
02 using System.Collections.Generic;
03 using System.ComponentModel;
04 using System.Data;
05 using System.Drawing;
06 using System.Linq;
07 using System.Text;
08 using System.Threading.Tasks;
09 using System.Windows.Forms;
10
11 namespace FirstFormApplication
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }
20 }
```

■ partial 클래스

코드 5-27 일반적인 클래스

```
class Example
{
    public int a;
    public int b;
}
```

코드 5-28 partial 클래스

```
partial class Example
{
    public int a;
}

partial class Example
{
    public int b;
}
```

Section 07 윈도 폼: 윈도 폼 기본 익히기(7)

■ 디자인 화면에서 요소 생성

- 도구 상자 열기 : 화면 왼쪽의 도구 상자 클릭 또는 메뉴의 [보기] - [도구 상자]

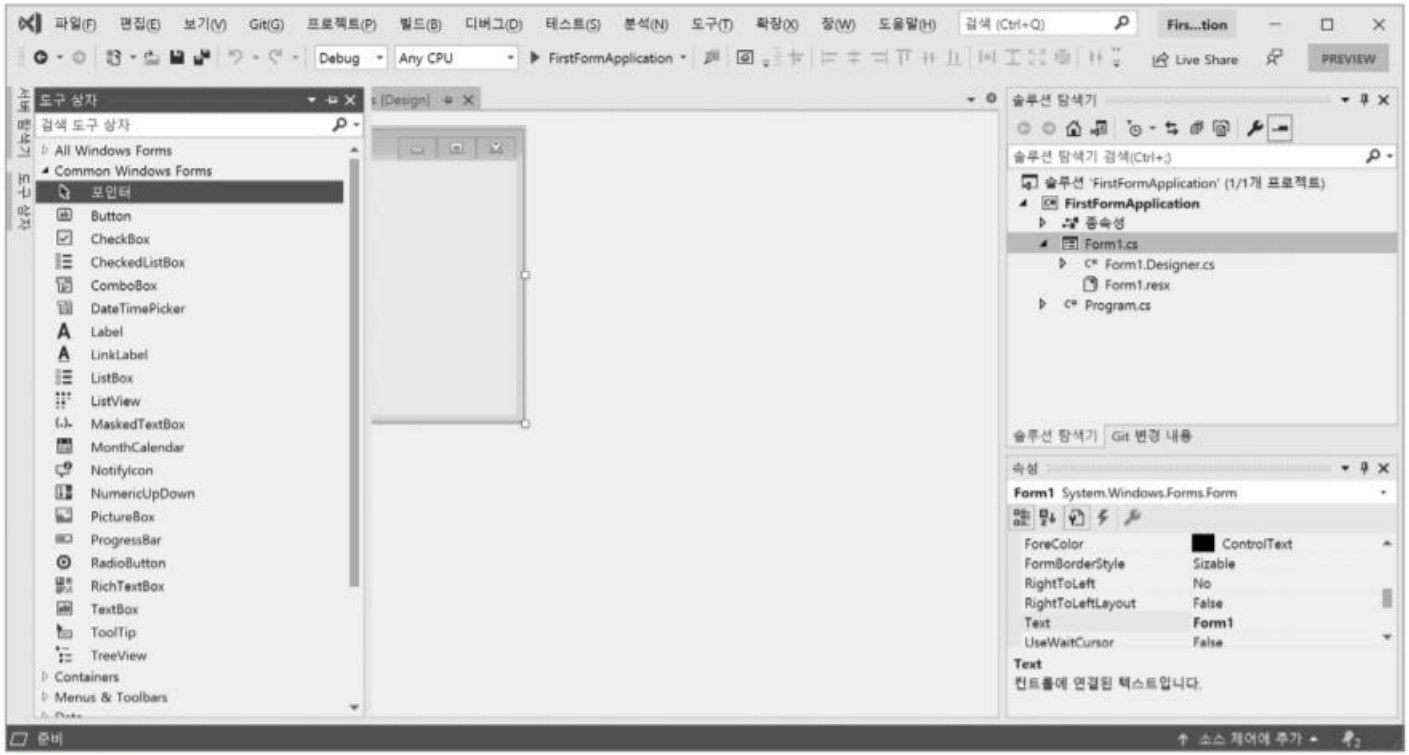


그림 5-30 도구 상자 열기

Section 07 윈도 폼: 윈도 폼 기본 익히기(8)

- 도구 상자에서 원하는 요소를 폼 위로 드래그

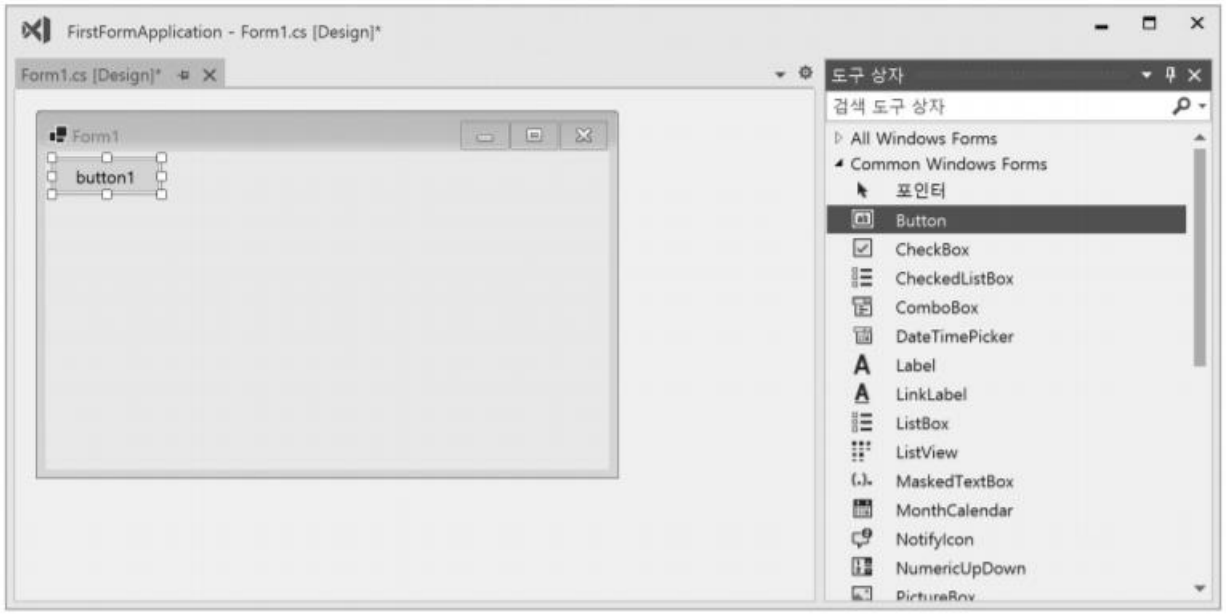


그림 5-31 도구 상자의 사용

Section 07 윈도 폼: 윈도 폼 기본 익히기(9)

- 속성 보기(메뉴 [보기] - [속성]), 속성 지정

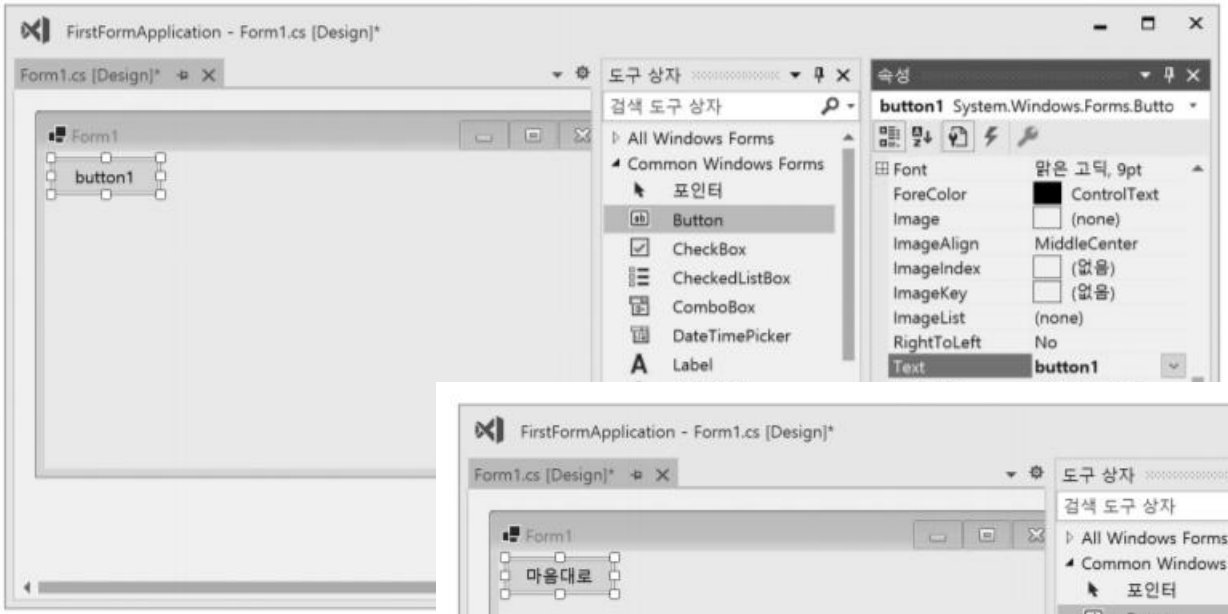


그림 5-32 속성 메뉴 열기

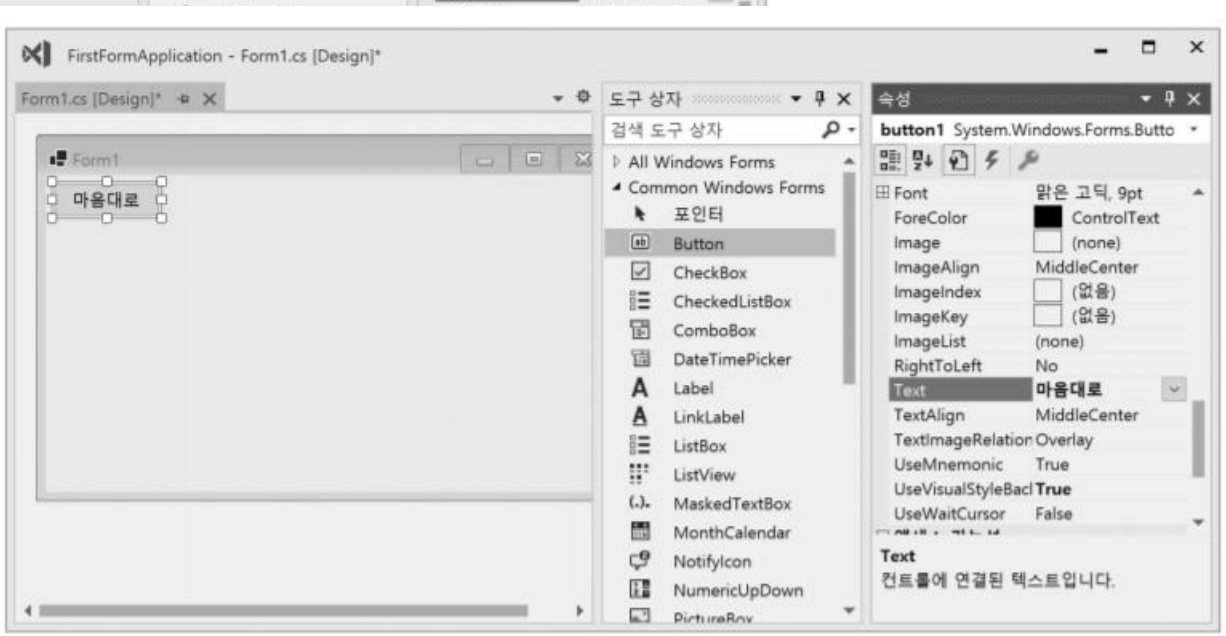


그림 5-33 속성의 지정

Section 07 윈도 폼: 윈도 폼 기본 익히기(10)

■ 디자인 코드

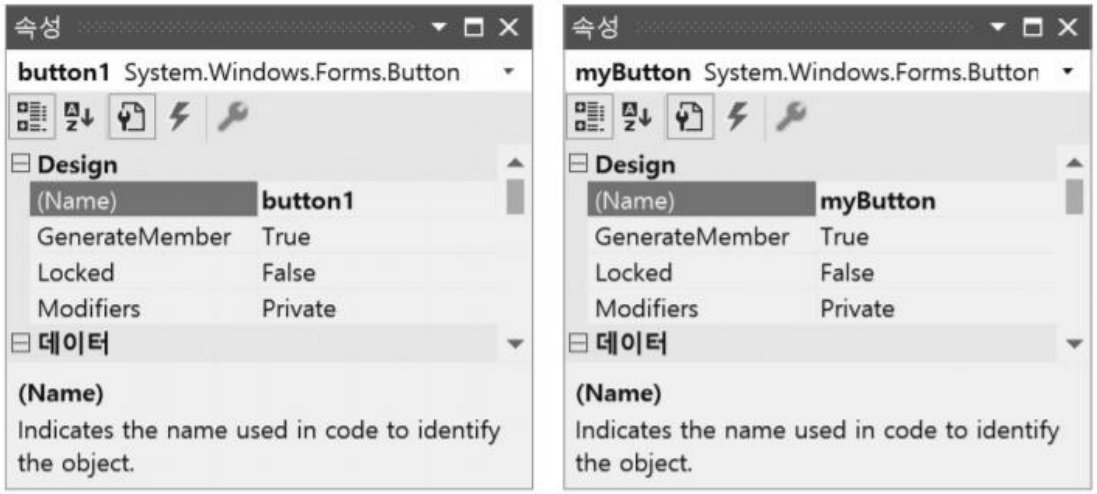


그림 5-34 Name 속성

- Form1.Designer.cs 파일 확인 : 버튼과 관련된 코드들 추가된 것 확인
- Name 속성에서 지정한 myButton 글자로 변수 만들어짐
- 코드로 인스턴스 생성, 속성 지정을 살펴볼 수 있음

■ 코드에서 요소의 속성 지정

```
public partial class Form1 : Form
{
    참조 1개
    public Form1()
    {
        InitializeComponent();

        myB
    }
}
```

myButton (필드) Button Form1.myButton

그림 5-35 myButton 객체

```
myButton.
```

★ Enabled
★ Anchor
★ Location
★ Text
★ DialogResult
AccessibilityObject
AccessibleDefaultActionDescription
AccessibleDescription

bool Control.Enabled { get; set; }
컨트롤이 사용자 상호 작용에 응답할 수 있는지를 나타내는 값을 가져오거나 설정합니다.
★ 이 컨텍스트를 기반으로 한 IntelliCode 제안

그림 5-36 myButton 객체의 속성

```
myButton.te
```

★ Text
Text
TextAlign
TextChanged
TextImageRelation
ResetText
UseCompatibleTextRendering

string ButtonBase.Text { get; set; }
이 컨트롤과 관련된 텍스트를 가져오거나 설정합니다.
★ 이 컨텍스트를 기반으로 한 IntelliCode 제안

그림 5-37 myButton 객체의 Text 속성

■ 코드에서 요소의 속성 지정

코드 5-30 Text 속성을 코드에서 변경

```
01 public partial class Form1 : Form
02 {
03     public Form1()
04     {
05         InitializeComponent();
06
07         myButton.Text = "코드에서 변경!";
08     }
09 }
```

그림 5-38 코드에서 변경한 글자

■ 코드에서 요소 생성

코드 5-32 Button 클래스의 인스턴스 생성

```
01 public partial class Form1 : Form
02 {
03     public Form1()
04     {
05         InitializeComponent();
06
07         myButton.Text = "코드에서 변경!";
08         myButton.Width = 100;
09         myButton.Height = 23;
10         Button button = new Button();
11     }
12 }
```

Section 07 윈도 폼: 윈도 폼 기본 익히기(14)

- 생성한 버튼 화면에 붙이기 : Form1 클래스가 가지고 있는 Controls 속성 사용
- Controls 속성은 부모로부터 상속
- Controls.Add( ) 메서드를 사용

코드 5-33 생성한 버튼 추가

```
01 public partial class Form1 : Form
02 {
03     public Form1()
04     {
05         InitializeComponent();
06
07         myButton.Text = "코드에서 변경!";
08         myButton.Width = 100;
09         myButton.Height = 23;
10         Button button = new Button();
11         Controls.Add(button);
12     }
13 }
```

상속입니다. 이와 관련된 내용은 7장에서 다룹니다. 일단 이때 상속을 사용했었다고 대충 기억해두면 나중에 공부할 때 도움이 될 것입니다.

Section 07 윈도 폼: 윈도 폼 기본 익히기(15)

- 새로 만든 button 객체에 속성을 지정

그림 5-41 Location 속성

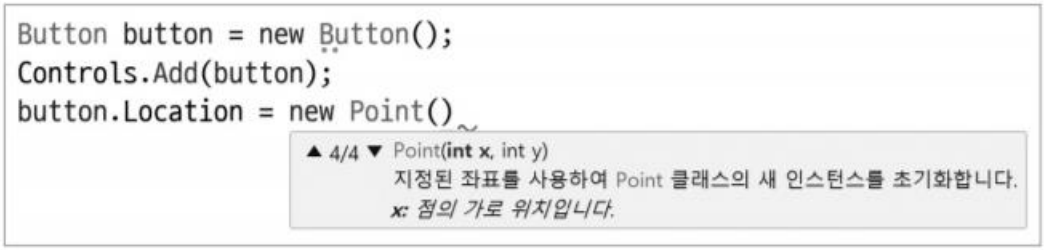


그림 5-42 Point 클래스

Section 07 윈도 폼: 윈도 폼 기본 익히기(16)

코드 5-34 생성한 버튼의 속성 지정

```
01 public partial class Form1 : Form
02 {
03     public Form1()
04     {
05         InitializeComponent();
06
07         myButton.Text = "코드에서 변경!";
08         myButton.Width = 100;
09         myButton.Height = 23;
10         Button button = new Button();
11         Controls.Add(button);
12     }
13 }
```

Section 07 윈도 폼: 윈도 폼 기본 익히기(17)

코드 5-35 반복문으로 여러 개의 버튼 생성

```
01 public partial class Form1 : Form
02 {
03     public Form1()
04     {
05         InitializeComponent();
06         // 숫자를 적절하게 조절해서 사용해주세요
07         int width = 100;
08         int height = 23;
09         int margin = 6;
10         myButton.Text = "코드에서 변경!";
11         myButton.Width = 100;
12
13         for (int i = 0; i < 5; i++)
14         {
15             Button button = new Button();
16             Controls.Add(button);
17             button.Location = new Point(margin, (height + margin) * (i + 1) + margin);
18             button.Text = "동적 생성 " + i + "번째";
19             button.Width = width;
20             button.Height = height;
21         }
22     }
23 }
```

그림 5-43 동적으로 생성된 버튼