Chapter 08
클래스 심화

Section 01 제네릭

- 클래스 내부에서 자료형에 별칭Alias을 지정하는 기능
- <>기호 내부에 식별자를 지정
- 제네릭 활용 방법
  - 제네릭 하나 지정할 때는 보통 식별자로 T를 사용(개발자들 사이의 약속)

```
class Wanted<T>
{

}
```

➔ Wanted<int>, Wanted<float>

코드 8-2 두 개 이상의 제네릭을 사용하는 클래스 /8장/Generics

```
01 class Test<T, U>
02 {
03
04 }
```

Section 01 제네릭

- **기본예제 8-1** 정수와 연산자(교재 396p) /8장/GenericBasic

```
class Wanted<T>
    {
        public T Value;
        public Wanted(T value)
        {
            this.Value = value;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Wanted<string> wantedString = new Wanted<string>("String");
            Wanted<int> wantedInt = new Wanted<int>(52273);
            Wanted<double> wantedDouble = new Wanted<double>(52.273);

            Console.WriteLine(wantedString.Value);
            Console.WriteLine(wantedInt.Value);
            Console.WriteLine(wantedDouble.Value);
        }
    }
```

실행 결과

```
String
52273
52.273
```

■ where 키워드

- 제네릭의 제한

코드 8-3 where 키워드 /8장/Generics

```
class Test<T, U>
    where T : class
    where U : struct
{

}
```

T는 클래스여야 합니다.

U는 구조체여야 합니다.

```
class Test<T, U>
    where T : IComparable
    where U : IComparable, IDisposable
{

}
```

T는 IComparable 또는 IComparable의 상속을 받은 것이어야 합니다.

U는 IComparable과 IDisposable 또는 이러한 것들의 상속을 받은 것이어야 합니다.

Section 02 인덱서

- 사용자 정의 클래스에서 [ ] 괄호를 사용하려면? → 인덱서(Indexer)
- 인덱서 선언 예

코드 8-4 인덱서 선언 /8장/Indexers

```
01 class Products
02 {
03     public int this[int i]  ——— 인덱서를 선언합니다.
04     {
05         // Products products = new Product();
06         // products[i] 할 때에 호출
07         get { return i; }
08         // products[i] = 10 할 때에 호출
09         set { Console.WriteLine(i + "번째 상품 설정"); }
10     }
11 }
```

Section 02 인덱서

- **기본예제 8-2** 인덱서로 배열처럼 사용하는 제곱 클래스 (교재 399p) /8장/IndexerBasic

```
class SquareCalculator
    {
        public int this[int i]
        {
            get
            {
                return i * i;
            }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            SquareCalculator square = new SquareCalculator();
            Console.WriteLine(square[10]);
        }
    }
```

실행 결과

```
100
```

Section 03 out 키워드(1)

- 값을 여러 개 반환하고자 할 때 사용, 대표적인 메서드는 TryParse( ) 메서드
- int.TryParse( ) 메서드의 기본 형태

C# 복사

```
public static bool TryParse (string s, out int result);
```

그림 8-2 int.TryParse() 메서드 선언

- out 키워드가 붙은 매개변수 입력할 때

```
int.TryParse("52273", out output);
```

- TryParse( ) 메서드는 숫자로 바꿀 수 있는 문자열을 매개변수로 넣으면 true 반환
- 바꿀 수 없는 문자열을 매개변수로 넣으면 false 반환
- 문자열을 숫자로 변환한 결과는 반환 않고 매개변수 out int result 에 넣은 변수로 반환

Section 03 out 키워드(2)

- **기본예제 8-3** int.TryParse( ) 메서드 (교재 401p) /8장/TryMethod

```
class Program
    {
        static void Main(string[] args)
        {
            Console.Write("숫자 입력: ");
            int output;
            bool result = int.TryParse(Console.ReadLine(), out output);

            if (result)
            {
                Console.WriteLine("입력한 숫자: " + output);
            }
            else
            {
                Console.WriteLine("숫자를 입력해주세요!");
            }
        }
    }
```

out 키워드를 붙여서 매개변수에 넣어야 함

실행 결과

```
숫자 입력: 52273
입력한 숫자: 52273
```

실행 결과

```
숫자 입력: ㅇㅂㅇ
숫자를 입력해주세요!
```

Section 03 out 키워드(2)

- out 키워드는 매개변수로 넣은 변수로 값 넣어줌.
- 변수가 아닌 일반 자료를 넣으면 오류 발생

그림 8-3 out 키워드를 사용하는 메서드

Section 03 out 키워드(3)

- **기본예제 8-4** out 키워드를 사용하는 메서드 생성 (교재 403p) /8장/MethodWithOut

```
class Program
    {
        static void NextPosition(int x, int y, int vx, int vy, out int rx, out int ry)
        {
            // 다음 위치 = 현재 위치 + 현재 속도
            rx = x + vx;
            ry = y + vy;
        }

        static void Main(string[] args)
        {
            int x = 0;
            int y = 0;
            int vx = 1;
            int vy = 1;

            Console.WriteLine("현재 좌표: (" + x + "," + y + ")");
            NextPosition(x, y, vx, vy, out x, out y);
            Console.WriteLine("다음 좌표: (" + x + "," + y + ")");
        }
    }
```

실행 결과

```
현재 좌표: (0,0)
다음 좌표: (1,1)
```

Section 04 구조체(1)

- 간단한 객체 만들 때 사용하는 형식
- 클래스와 거의 동일한 구문 사용, 복사 형식이 다르고 클래스보다 제한 많음
- 구조체는 상속, 인터페이스 구현 불가능, 클래스보다 안정성 높음
- C#의 기본 자료형은 모두 구조체로 정의

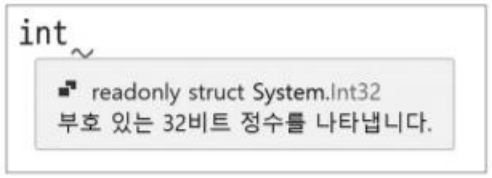

그림 8-4 int 자료형은 구조체

- 구조체를 만드는 기본 방법은 클래스를 만드는 방법과 같음

Section 04 구조체(2)

■ 구조체 선언

- 멤버 변수로 x와 y를 갖는 구조체 만들기

```
struct Point
{
    public int x;
    public int y;
}
```

- 이렇게 생성한 구조체는 new 키워드를 사용하지 않아도 인스턴스 생성할 수 있음
- 단, 멤버 변수를 꼭 초기화 해야함 아니면, 해당 멤버 변수 사용시 오류 발생!

Section 04 구조체(2)

- **기본예제 8-5** new 키워드를 사용하지 않고 구조체 인스턴스 생성 (교재 405p)

/8장/StructBasic

```
class Program
    {
        struct Point
        {
            public int x;
            public int y;
        }

        static void Main(string[] args)
        {
            Point point;
            point.x = 10;
            point.y = 10;

            Console.WriteLine(point.x);
            Console.WriteLine(point.y);
        }
    }
```

→ **x, y 초기화**

실행 결과

```
10
10
```

■ 구조체의 생성자

- 매개변수 없는 생성자 선언 불가 → 매개변수 없는 생성자 자동 정의됨!
- 예(오류 발생)

코드 8-9 구조체에 매개변수 없는 생성자는 선언 불가능 /8장/Structures

```
01 struct Point
02 {
03     public int x;
04     public int y;
05
06     public Point()
07     {
08
09     }
10 }
```

Section 04 구조체(4)

- 반드시 매개변수를 넣어 생성자를 만들어야 함

코드 8-10 구조체의 생성자 /8장/Structures

```
01 struct Point
02 {
03     public int x;
04     public int y;
05
06     public Point(int x, int y)
07     {
08         this.x = x;
09         this.y = y;
10     }
11 }
```

- 매개변수 있는 생성자를 만들어도 매개변수 없는 생성자 사용 가능
- 내부 변수는 자동적으로 해당 자료형의 기본 값으로 초기화됨

  → 숫자 관련 자료형이라면 0, 문자열 또는 객체라면 null 로 초기화됨

Section 04 구조체(5)

- **기본예제 8-6** 구조체의 생성자 (교재 407p) /8장/ConstructorOfStruct

```
class Program
    {
        struct Point
        {
            public int x;
            public int y;

            public Point(int x, int y)
            {
                this.x = x;
                this.y = y;
            }
        }

        static void Main(string[] args)
        {
            Point point = new Point();

            Console.WriteLine(point.x);
            Console.WriteLine(point.y);
        }
    }
```

구조체의 인스턴스를 생성하고,
멤버 변수 출력

실행 결과

```
0
0
```

Section 04 구조체(6)

- 생성자는 반드시 모든 멤버 변수 초기화 상태로 만들어줘야 함
- 선언과 동시에 멤버 변수 초기화 할 수 없음

코드 8-12 구조체는 생성자에서 모든 변수를 초기화해야 함 /8장/Structures

```
01 struct Point
02 {
03     public int x;
04     public int y;
05     public string testA;
06     public string testB = "init";
07
08     public Point(int x, int y)
09     {
10         this.x = x;
11         this.y = y;
12     }
13 }
```

초기화되지 않은 멤버 변수이므로
생성자를 만든다면, 생성자에서 반드시 초기화해줘야 합니다.

선언과 동시에 초기화할 수 없습니다. 오류가 발생합니다.

멤버 변수 testA를 초기화하지 않았으므로, 오류가 발생합니다.

Section 04 구조체(7)

```
struct Point
{
    public int x;
    public int y;
    public string testA;
    public string testB = "init";

    public Point(int
    {
        this.x = x;
        this.y = y;
    }
}
```

(필드) string Point.testB

CS0573: 'Point': 구조체에는 인스턴스 속성 또는 이니셜라이저를 사용할 수 없습니다.

그림 8-7 구조체 관련 오류(1)

```
struct Point
{
    public int x;
    public int y;
    public string testA;
    public string testB = "init";

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

Point.Point(int x, int y)

CS0171: 제어를 호출자에게 반환하려면 'Point.testA' 필드가 완전히 할당되어야 합니다.

그림 8-8 구조체 관련 오류(2)

Section 04 구조체(8)

- 구조체의 가장 중요한 의미는 "안전성"
- 모든 매개변수를 초기화해서 사용하게 만드는 것

코드 8-13 구조체의 초기화 형태 /8장/Strucures

```
01 struct Point
02 {
03     public int x;
04     public int y;
05     public string testA;
06     public string testB;
07
08     public Point(int x, int y)──────────────── 생성자 오버로딩했습니다.
09     {
10         this.x = x;
11         this.y = y;
12         this.testA = "초기화";
13         this.testB = "초기화";
14     }
15
16     public Point(int x, int y, string test)──── 생성자 오버로딩했습니다.
17     {
18         this.x = x;
19         this.y = y;
20         this.testA = test;
21         this.testB = test;
22     }
23 }
```

■ 구조체에서 클래스 인스턴스를 멤버 변수로 선언할 때

- **반드시 초기화**

코드 8-14 구조체에서 클래스 인스턴스를 멤버 변수로 선언 /8장/Strucures

```
struct Point
{
    public int x;
    public int y;
    public Program program;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
        this.program = null;
    }
}
```

null로 초기화해주었습니다.

Section 04 구조체(9)

- **기본예제 8-7 구조체의 값 복사** (교재 411p) /8장/StructCopy

```
class Program
    {
        class PointClass
        {
            public int x;
            public int y;

            public PointClass(int x, int y)
            {
                this.x = x;
                this.y = y;
            }
        }

        struct PointStruct
        {
            public int x;
            public int y;

            public PointStruct(int x, int y)
            {
                this.x = x;
                this.y = y;
            }
        }
```

실행 결과

```
pointClassA: 100,200
pointClassB: 100,200

pointStructA: 10,20
pointStructB: 100,200
```

Section 04 구조체(9)

- **기본예제 8-7 구조체의 값 복사** (교재 411p) /8장/StructCopy

```
    static void Main(string[] args)
    {
        // 클래스
        PointClass pointClassA = new PointClass(10, 20);
        PointClass pointClassB = pointClassA;

        pointClassB.x = 100;
        pointClassB.y = 200;

        Console.WriteLine("pointClassA: " + pointClassA.x + "," + pointClassA.y);
        Console.WriteLine("pointClassB: " + pointClassB.x + "," + pointClassB.y);
        Console.WriteLine();

        // 구조체
        PointStruct pointStructA = new PointStruct(10, 20);
        PointStruct pointStructB = pointStructA;

        pointStructB.x = 100;
        pointStructB.y = 200;

        Console.WriteLine("pointStructA: " + pointStructA.x + "," + pointStructA.y);
        Console.WriteLine("pointStructB: " + pointStructB.x + "," + pointStructB.y);
    }
}
```

Section 10 윈도 폼: 윈도 폼에 메뉴와 상태 표시줄 만들기(1)

■ 메뉴

- 도구 상자에서 MenuStrip 드래그, 폼 위에 놓으면 메뉴 만들어짐

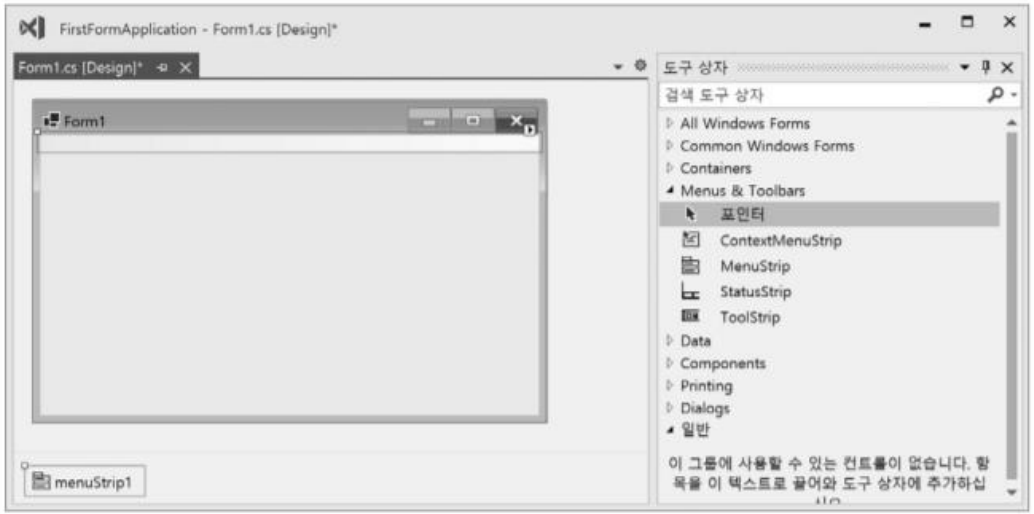

그림 8-9 MenuStrip 배치

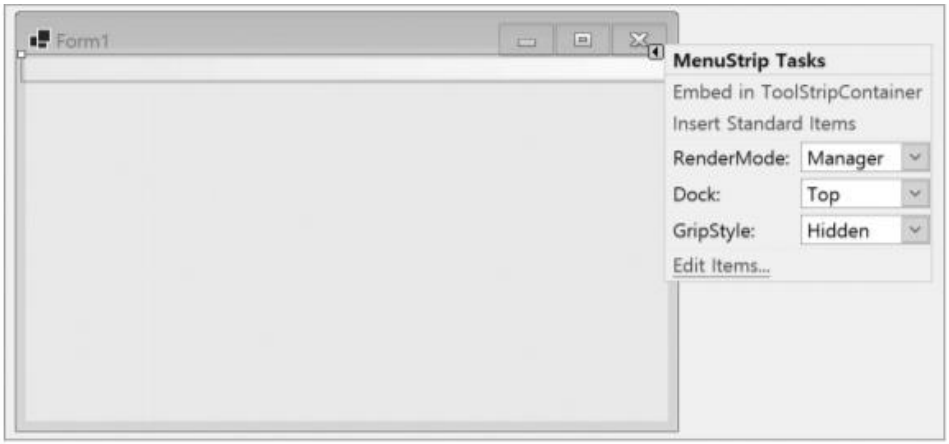

그림 8-10 메뉴 오른쪽 위의 화살표 아이콘을 클릭한 상태

Section 10 윈도 폼: 윈도 폼에 메뉴와 상태 표시줄 만들기(2)

- 메뉴를 추가하려면, 메뉴 오른쪽 위에 있는 삼각형 모양의 아이콘을 클릭해서 [MenuStrip Tasks]를 선택

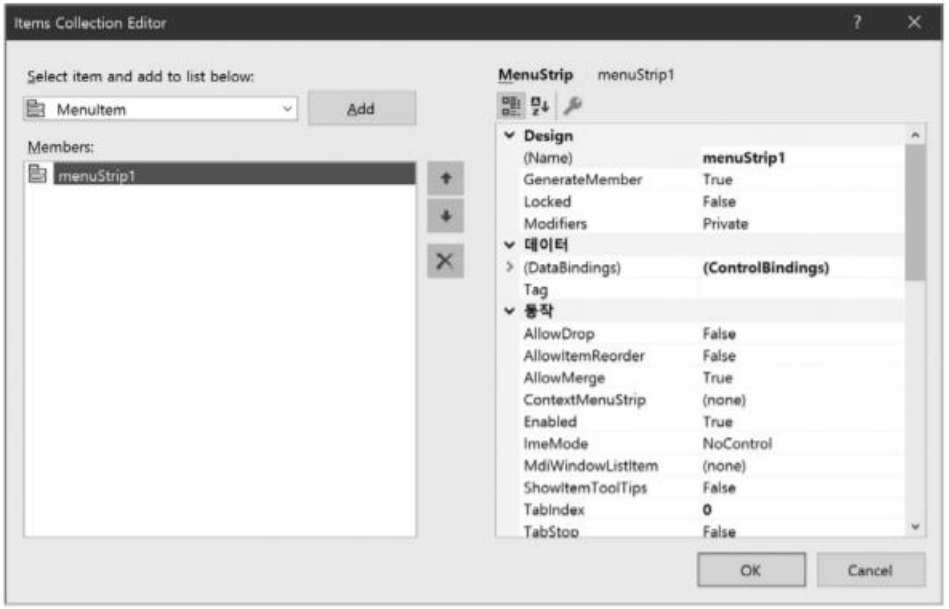

그림 8-11 Items Collection Editor

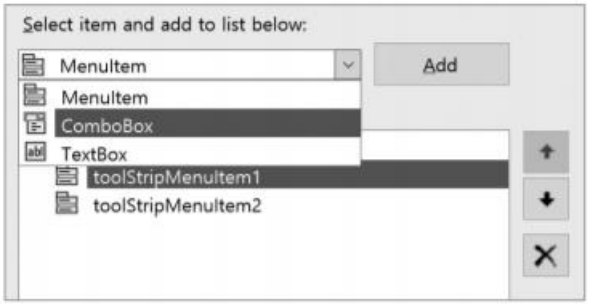

그림 8-12 추가할 수 있는 항목

Section 10 윈도 폼: 윈도 폼에 메뉴와 상태 표시줄 만들기(3)

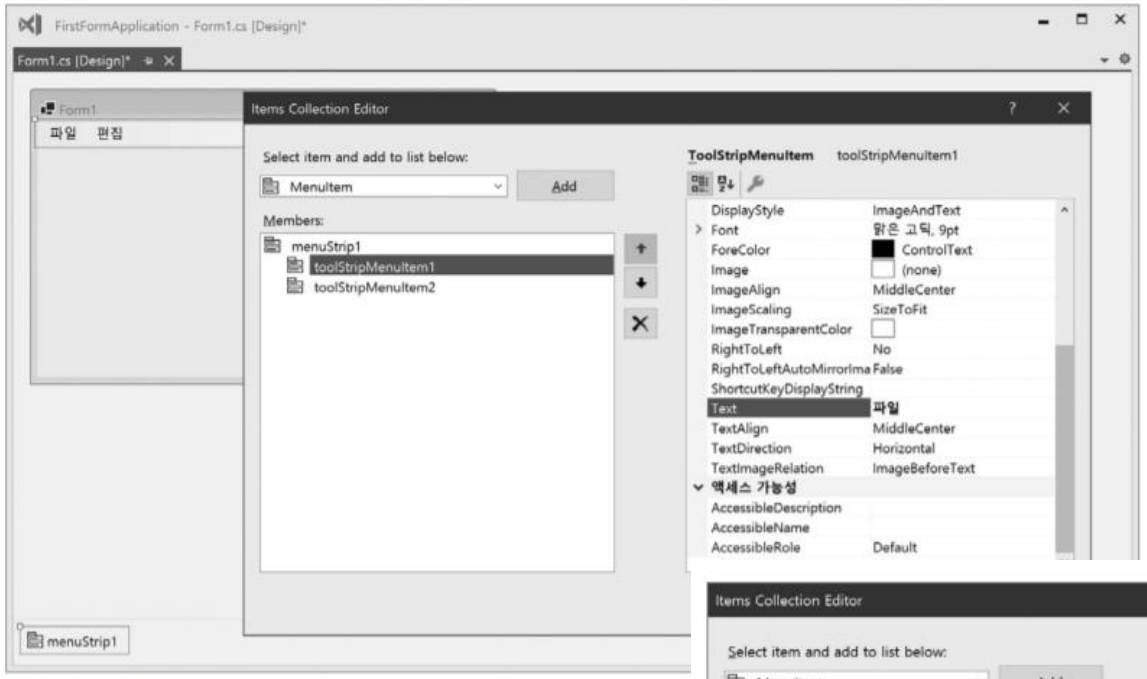

그림 8-13 Text 속성 변경으로 메뉴 글자 변경하기

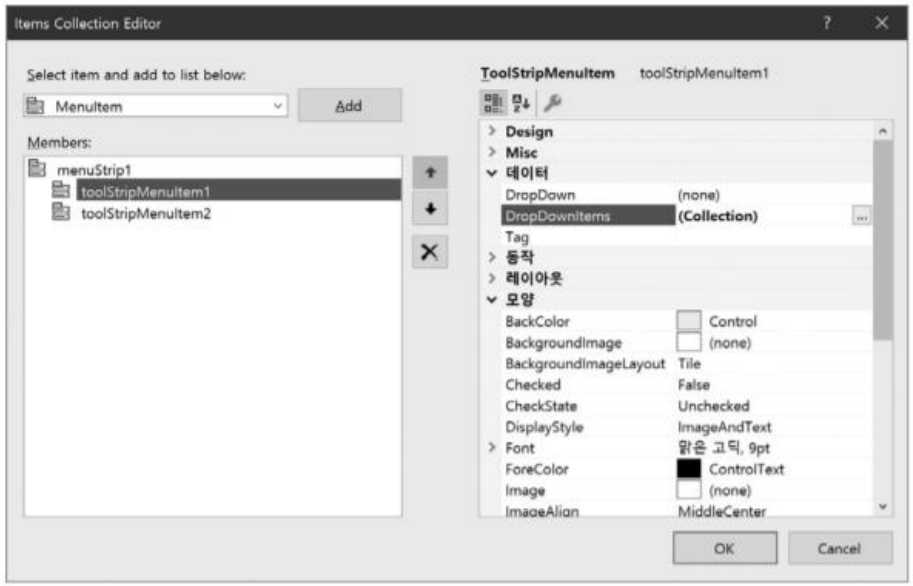

그림 8-14 DropDownItems 속성

Section 10 윈도 폼: 윈도 폼에 메뉴와 상태 표시줄 만들기(4)

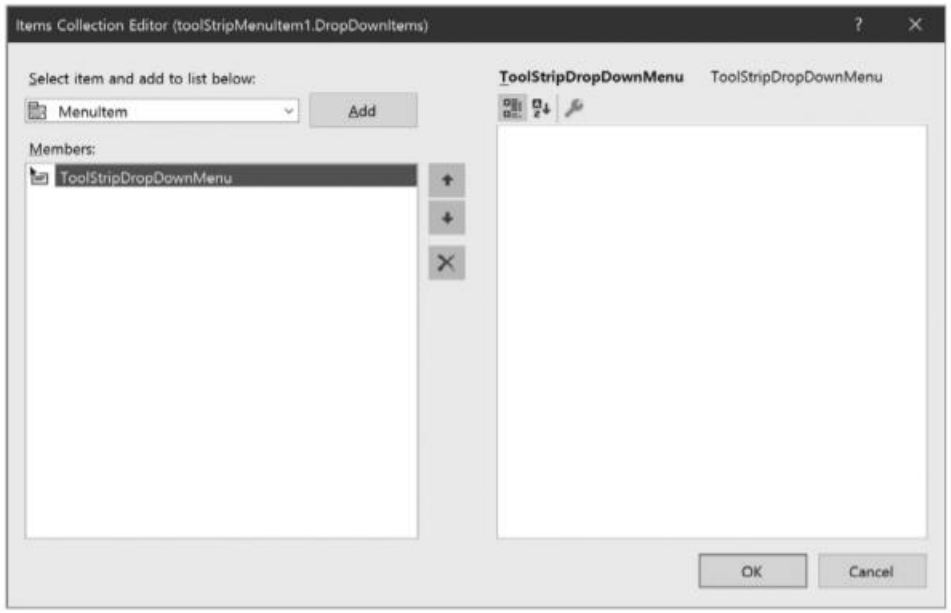

그림 8-15 추가적인 Items Collection Editor

그림 8-16 단계를 추가한 메뉴

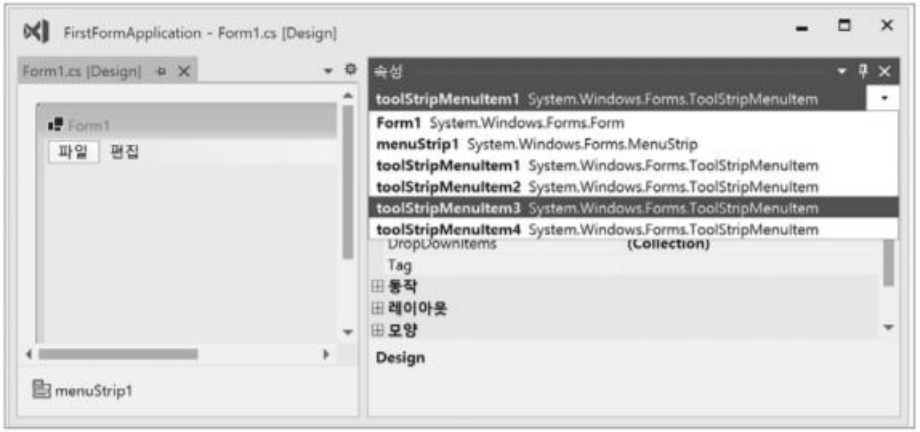

그림 8-17 속성 패널에서 컴포넌트 선택하기

Section 10 윈도 폼: 윈도 폼에 메뉴와 상태 표시줄 만들기(5)

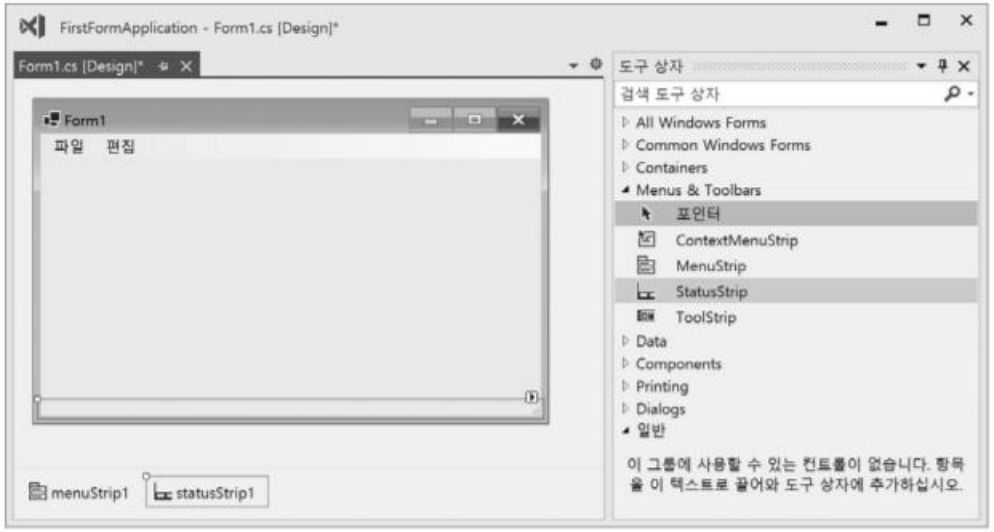

그림 8-18 StatusStrip을 사용한 메뉴

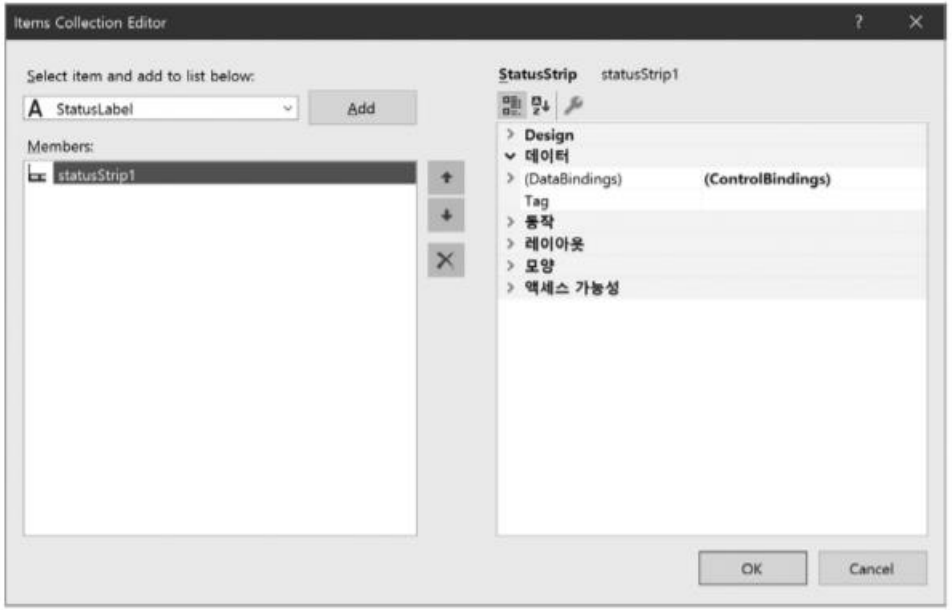

그림 8-19 StatusStrip의 Items Collection Editor0