클래스-2

이것이 C#이다

Contents

- 오버라이딩과 다형성
- 메소드 숨기기
- 오버라이딩 봉인하기
- 중첩 클래스
- 분할 클래스
- 확장 메소드
- 구조체
- 튜플

7.10 오버라이딩과 다형성

❖ 객체가 여러 형태를 가질 수 있음을 의미

- **하위 형식 다형성**(Subtype Polymorphism)

```
class IronMan : ArmorSuite
{
    public override void Initialize()
    {
        base.Initialize();
        Console.WriteLine("Repulsor Rays Armed");
    }
}

class WarMachine : ArmorSuite
{
    public override void Initialize()
    {
        base.Initialize();
        Console.WriteLine("Double-Barrel Cannons Armed");
        Console.WriteLine("Micro-Rocket Launcher Armed");
    }
}
```

❖ 데모 예제 – Overriding

7.11 메소드 숨기기

- ❖ 기반 클래스의 메소드를 감추고 파생 클래스 구현만 표시
- ❖ 파생 클래스 버전의 메소드를 new 한정자로 수식

```
class Base
{
    public void MyMethod()
    {
        Console.WriteLine("Base.MyMethod()");
    }
}

class Derived : Base
{
    public new void MyMethod()
    {
        Console.WriteLine("Derived.MyMethod()");
    }
}
```

Base.MyMethod()를 감추고 Derived 클래스에서 구현된 MyMethod()만 노출합니다.

- ❖ 오버라이드와 다름 → 완전한 다형성 표현의 한계
- ❖ 데모 예제 - MethodHiding

7.12 오버라이딩 봉인하기

❖ 메소드의 오버라이딩 봉인

- 대상 - virtual 가상 메소드를 오버라이딩한 메소드
- 오작동 위험이 있거나 잘못 오버라이딩함으로써 문제가 예상되는 경우

❖ 사용 예

```
class Base
{
    public virtual void SealMe()
    {
        // …
    }
}

class Derived : Base
{
    public sealed void SealMe()
    {
        // …
    }
}
```

```
'WantToOverride.SealMe()': cannot override inherited member 'Derived.SealMe()'
because it is sealed
```

❖ 데모 예제 – SealedMethod

7.13 중첩 클래스

❖ 클래스 안에 선언되어 있는 클래스

- 소속되어 있는 클래스의 멤버에 자유롭게 접근 (private 멤버 포함)

❖ 선언 형식

```
class OuterClass
{
    class NestedClass
    {

    }
}
```

```
class OuterClass
{
    private int OuterMember;

    class NestedClass
    {
        public void DoSomething()
        {
            OuterClass outer = new OuterClass();
            outer.OuterMember = 10;
        }
    }
}
```

OuterClass의 private 멤버에 접근하여 값을 할당하거나 읽을 수 있습니다.

❖ 사용 이유

- 클래스 외부에 공개하고 싶지 않은 형식을 만들고자 할 때
- 현재 클래스의 일부처럼 표현 가능한 클래스를 만들고자 할 때

❖ 데모 예제 - NestedClass

7.14 분할 클래스

❖ 여러 번에 나눠서 구현하는 클래스

- 클래스의 구현이 길어질 경우 여러 파일에 나눠서 구현
- →소스 코드 관리의 편의를 제공
- partial 키워드 사용

❖ 사용 형식

```
partial class MyClass
{
    public void Method1( ) { }
    public void Method2( ) { }
}

partial class MyClass
{
    public void Method3( ) { }
    public void Method4( ) { }
}
```

클래스 이름은 동일해야 합니다.

❖ 데모 예제 - PartialClass

7.15 확장 메소드

- 기존 클래스의 기능을 확장하는 기법
- 선언 방법
- 선언 및 사용 예

```
namespace MyExtension
{
    ...
    }
}
```

```
using MyExtension;

// ...
int a = 2;
Console.WriteLine( a.Power( 3 ) );
Console.WriteLine( 10.Power(4) );
```

확장 메소드를 담는 클래스의 네임스페이스를 사용합니다.

마치 Power()가 원래부터 int 형식의 메소드였던 것처럼 사용할 수 있습니다.

- 데모 예제 - ExtensionMethod

7.16 구조체

❖ 클래스 vs. 구조체

특징	클래스	구조체
키워드	class	struct
형식	참조 형식	값 형식
복사	얕은 복사(Shallow Copy)	깊은 복사(Deep Copy)
인스턴스 생성	new 연산자와 생성자 필요	선언만으로도 생성
생성자	매개 변수 없는 생성자 선언 가능	매개 변수 없는 생성자 선언 불가능
상속	가능	모든 구조체는 System.Object 형식을 상속하는 System.ValueType으로부터 직접 상속받음

❖ 선언 형식 및 사용 예

```
struct 구조체이름
{
    // 필드 …
    // 메소드 …
}
```

```
struct MyStruct
{
    public int MyField1
    public int MyFiled2

    public void MyMethod()
    {
        // …
    }
```

❖ 데모 예제 - Structure

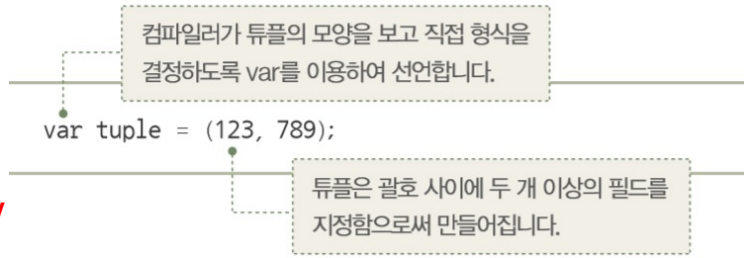
7.17 튜플(1)

❖ 여러 필드를 담을 수 있는 구조체

- 형식의 이름을 갖지 않음
- 임시적으로 사용할 복합 데이터 형식 선언에 적합

❖ 튜플 선언

- 명명되지 않은 선언
  - Item1, Item2....Item*N*




```
var tuple = (123, 789);
Console.WriteLine($"{tuple.Item1}, {tuple.Item2}"); // 출력 결과 : 123, 789
```

- 명명된 선언
  - 필드 명: 값

```
var tuple = (Name: "박상현", Age: 17);
Console.WriteLine($"{tuple.Name}, {tuple.Age}"); // 출력 결과 : 박상현, 17
```

7.17 튜플(2)

❖ 튜플 분해

```
var tuple = (Name: "박상현", Age: 17);
var (name, _) = tuple; // Age 필드는 무시
Console.WriteLine($"{name}"); // 출력 결과 : 박상현
```

❖ 명명되지 않은 튜플과 명명된 튜플 사이의 할당

```
var unnamed = ("슈퍼맨", 9999); // (string, int)
var named = (Name: "박상현", Age: 17); // (string, int)

named = unnamed;
Console.WriteLine($"{named.Name}, {named.Age}"); // 출력 결과 : 슈퍼맨, 9999

named = ("원더우먼", 10000)

unnamed = named;
Console.WriteLine($"{unnamed.Name}, {unnamed.Age}"); // 출력 결과 : 원더우먼, 10000
```

❖ System.ValueType 추가 패키지 필수

❖ 데모 예제 - Tuple

Thank You !

이것이 C#이다