3) 고급 문법 및 기능

매 강의 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

- C#의 제너릭과 out, ref 키워드의 개념을 이해하고 활용할 수 있습니다.

[목차]

모든 토글을 열고 닫는 단축키
Windows : `Ctrl` + `alt` + `t`
Mac : `⌘` + `⌥` + `t`

---

01. 제너릭

▼ 1) 제너릭 사용법

- 제너릭은 클래스나 메서드를 일반화시켜 다양한 자료형에 대응할 수 있는 기능입니다.
- 제너릭을 사용하면 코드의 재사용성을 높일 수 있습니다.
- C#에서는 `<T>` 형태의 키워드를 이용하여 제너릭을 선언합니다.
- 제너릭 클래스나 메서드에서 사용할 자료형은 선언 시점이 아닌 사용 시점에 결정됩니다.

- 제너릭 클래스나 메서드를 사용할 때는 `<T>` 대신 구체적인 자료형을 넣어줍니다.

▼ [코드스니펫] 제너릭 기초

```
class Stack<T>
{

}

Stack<int> intStack = new Stack<int>();
```

```
// 제너릭 클래스 선언 예시
class Stack<T>
{
    private T[] elements;
    private int top;

    public Stack()
    {
        elements = new T[100];
        top = 0;
    }

    public void Push(T item)
    {
        elements[top++] = item;
    }

    public T Pop()
    {
        return elements[--top];
    }
}

// 제너릭 클래스 사용 예시
Stack<int> intStack = new Stack<int>();
intStack.Push(1);
intStack.Push(2);
intStack.Push(3);
Console.WriteLine(intStack.Pop()); // 출력 결과: 3
```

- 제너릭을 두개 이상 사용하는 예

```
class Pair<T1, T2>
{
    public T1 First { get; set; }
    public T2 Second { get; set; }
```

```
    public Pair(T1 first, T2 second)
    {
        First = first;
        Second = second;
    }

    public void Display()
    {
        Console.WriteLine($"First: {First}, Second: {Second}");
    }
}
```

```
Pair<int, string> pair1 = new Pair<int, string>(1, "One");
pair1.Display();

Pair<double, bool> pair2 = new Pair<double, bool>(3.14, true);
pair2.Display();
```

출력

```
First: 1, Second: One
First: 3.14, Second: True
```

02. out, ref 키워드

▼ 1) 사용법

- out, ref 키워드는 메서드에서 매개변수를 전달할 때 사용합니다.
- out 키워드는 메서드에서 반환 값을 매개변수로 전달하는 경우에 사용합니다.
- ref 키워드는 메서드에서 매개변수를 수정하여 원래 값에 영향을 주는 경우에 사용합니다.
- out, ref 키워드를 사용하면 메서드에서 값을 반환하는 것이 아니라, 매개변수를 이용하여 값을 전달할 수 있습니다.

▼ [코드스니펫] out & ref 기초

```
// out 키워드 사용 예시
void Divide(int a, int b, out int quotient, out int remainder)
{

}
```

```
// ref 키워드 사용 예시
void Swap(ref int a, ref int b)
{

}
```

```
// out 키워드 사용 예시
void Divide(int a, int b, out int quotient, out int remainder)
{
    quotient = a / b;
    remainder = a % b;
}

int quotient, remainder;
Divide(7, 3, out quotient, out remainder);
Console.WriteLine($"{quotient}, {remainder}"); // 출력 결과: 2, 1

// ref 키워드 사용 예시
void Swap(ref int a, ref int b)
{
    int temp = a;
    a = b;
    b = temp;
}

int x = 1, y = 2;
Swap(ref x, ref y);
Console.WriteLine($"{x}, {y}"); // 출력 결과: 2, 1
```

▼ 2) 주의 사항

1. **값의 변경 가능성**:
   `ref` 매개변수를 사용하면 메서드 내에서 해당 변수의 값을 직접 변경할 수 있습니다. 이는 예기치 않은 동작을 초래할 수 있으므로 주의가 필요합니다.
2. **성능 이슈**:
   `ref` 매개변수는 값에 대한 복사 없이 메서드 내에서 직접 접근할 수 있기 때문에 성능상 이점이 있습니다. 그러나 너무 많은 매개변수를 `ref`로 전달하면 코드의 가독성이 떨어지고 유지보수가 어려워질 수 있습니다. 적절한 상황에서 `ref`를 사용하는 것이 좋습니다.
3. **변수 변경 여부 주의**:
   `out` 매개변수는 메서드 내에서 반드시 값을 할당해야 합니다. 따라서 `out` 매개변수를 전달할 때 해당 변수의 이전 값이 유지되지 않으므로 주의해야 합니다.

이전 강의

📕 2) 상속과 다형성

다음 강의

📕 1) 인터페이스와 열거형