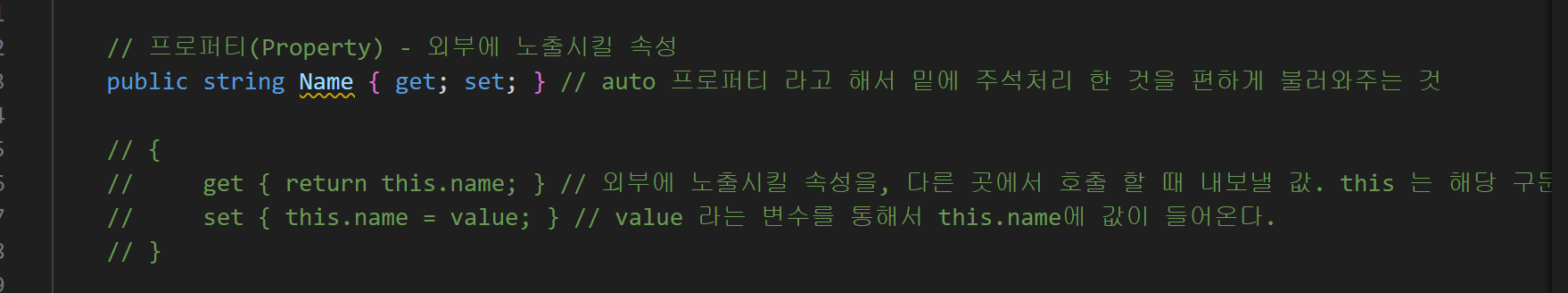
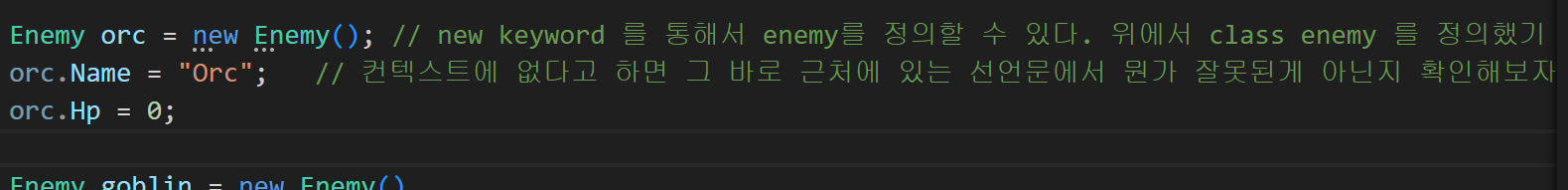
Unity 2일차 수업 내용

오토 프로퍼티

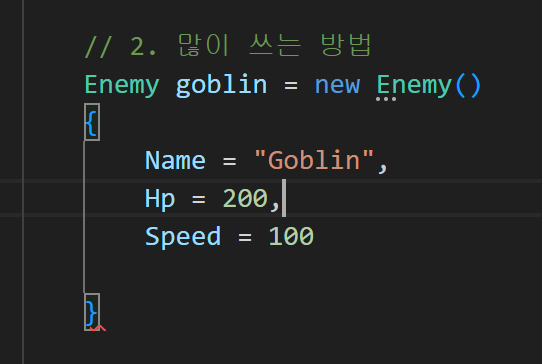


새 클래스를 생성하고 값을 추가하는 방법

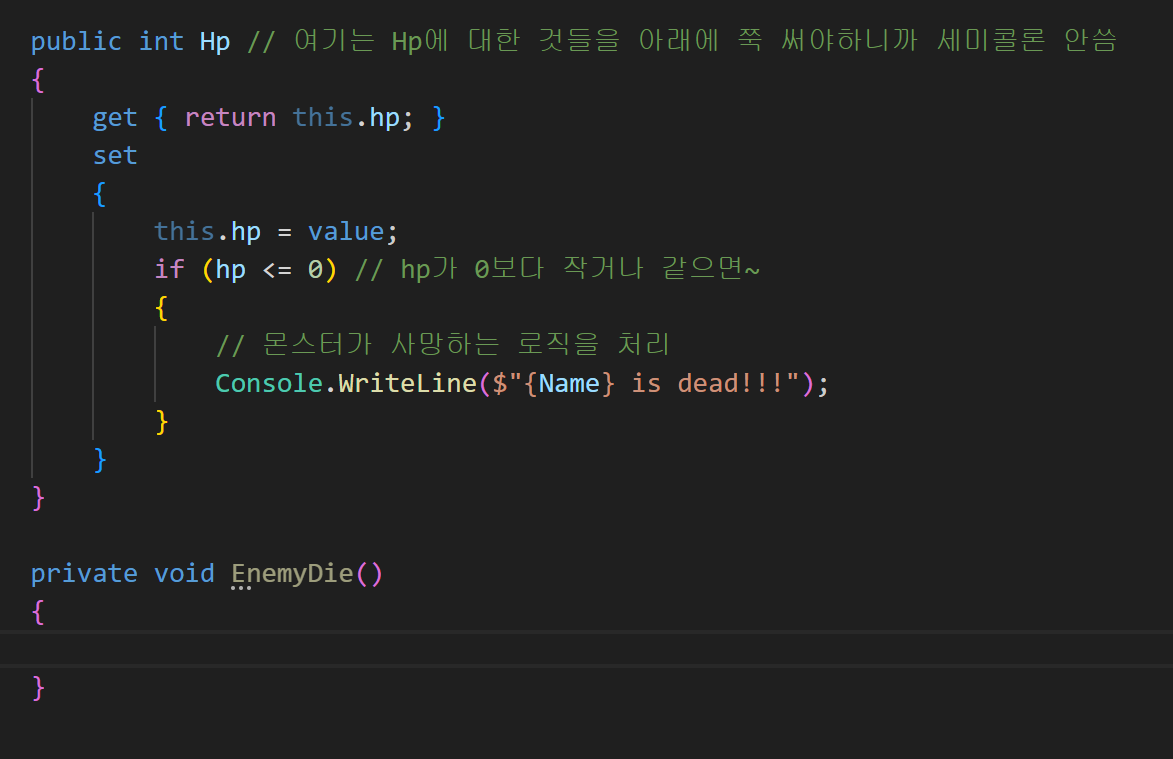
1. 일반적인 방법

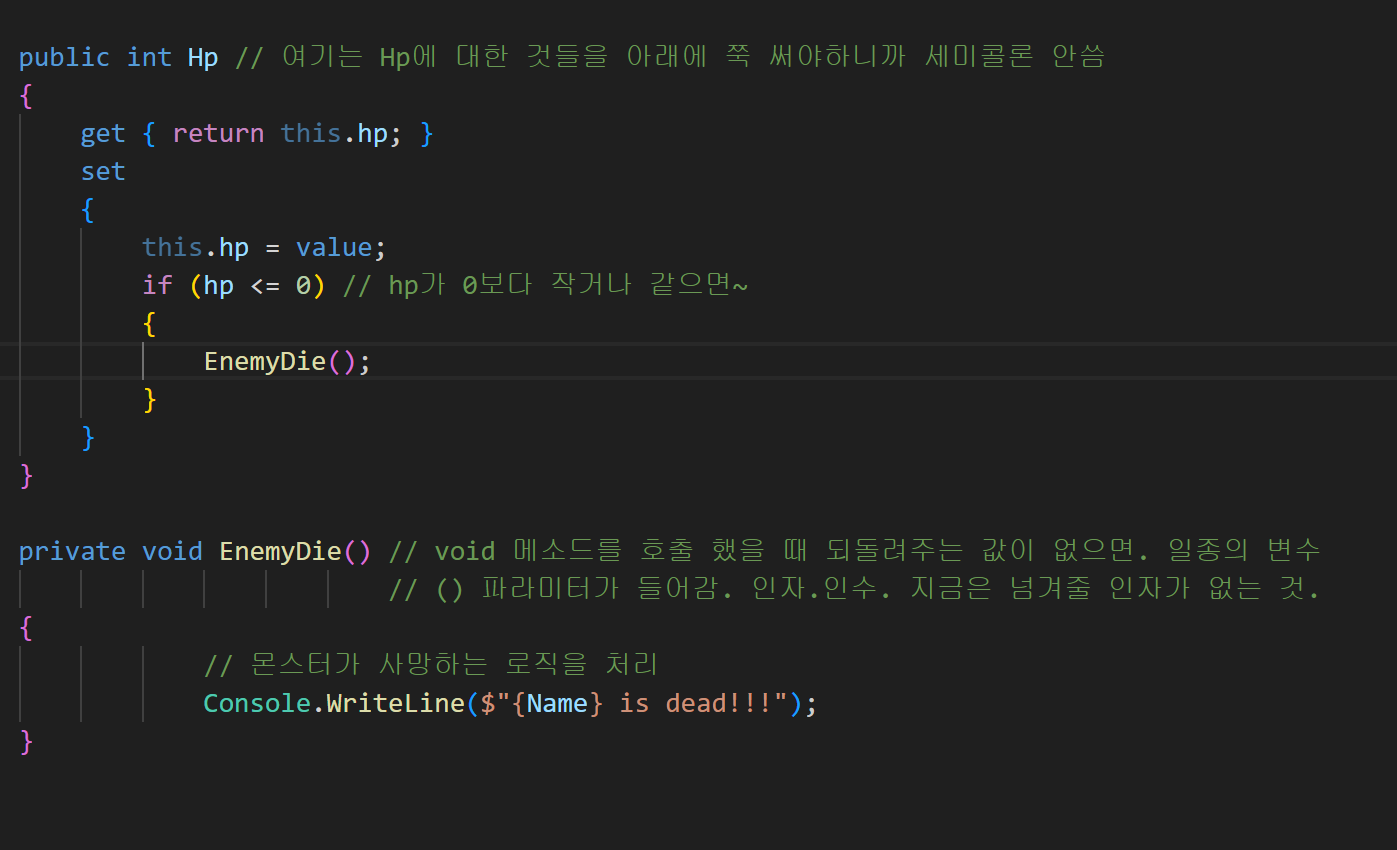


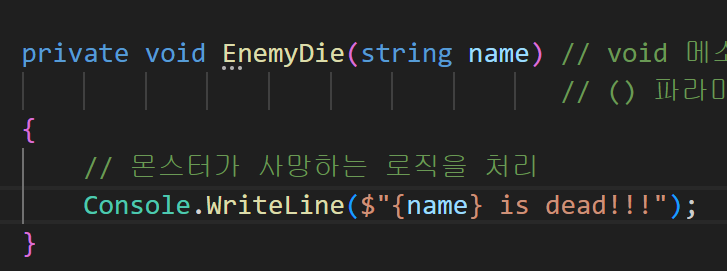
1. 편한 방법



몬스터 사망로직







파라미터에서 name 을 받기 위해서 대문자였던 N을 n으로 바꿔줌

    private void EnemyDie(string param) // void 메소드를 호출 했을 때 되돌려주는 값이 없으면. 일종의 변수

                                       // () 파라미터가 들어감. 인자.인수. 지금은 넘겨줄 인자가 없는 것.

    {

        // 몬스터가 사망하는 로직을 처리

        Console.WriteLine($"{param} is dead!!!");

    }

연습 때는 헷갈리니까 param으로 바꿔쓰기

상속 – 객체지향 oop 에서 중요한 개념

부모 -> 자식 -> 자식 …

부모로부터 유전적인 특징을 이어받는데, 자식은 또 다른 특징이 더해짐

즉 공통적인 특징은 부모에 두고 자식들은 각각 특징을 더할 때 쓰이면 편함

// Enemy 클래스를 상속한 오크

public class Orc : Enemy // Orc 라는 새로운 클래스를 생성하는데, 부모가 Enemy다.

{

}

// 생성자 Constructer 밑에서 new Enemy 를 만나면 생성하게 되는 것

    public Orc()

    {

        this.Name = "오크";

        this.Hp = 200;

        this.Speed = 50;

    }

파라미터가 있는 생성자

public class Goblin : Enemy

{

    // 파라메터가 있는 생성자

    public Goblin(string \_name, int \_hp, int \_speed) //\_를 넣는 이유는 기존에 있던 변수명과 다르게 하기 위해

    {

        this.Name = \_name;

        this.Hp = \_hp;

        this.Speed = \_speed;

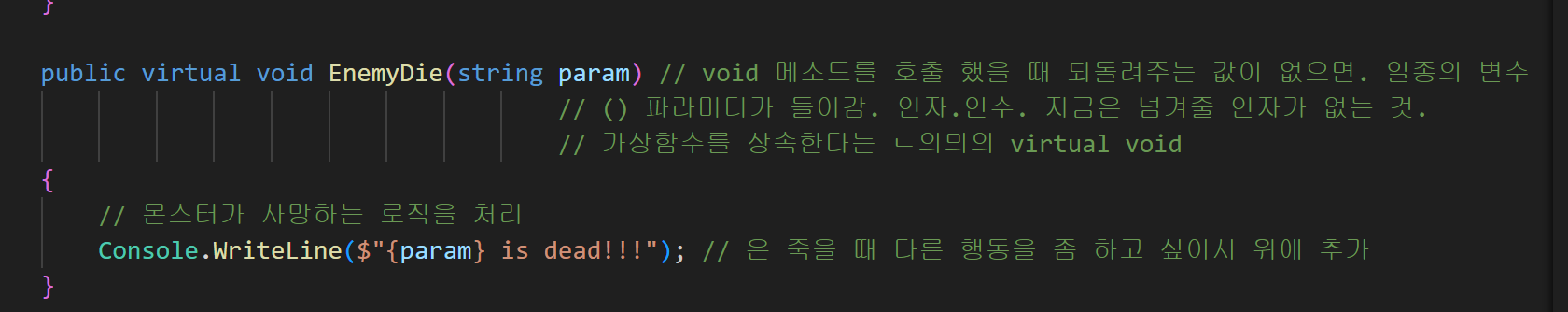
    }

}

이후에 밑으로 내려와서 상속받은 클래스 만들기

Orc 죽을 때 다른 행동 \*\*\*\*

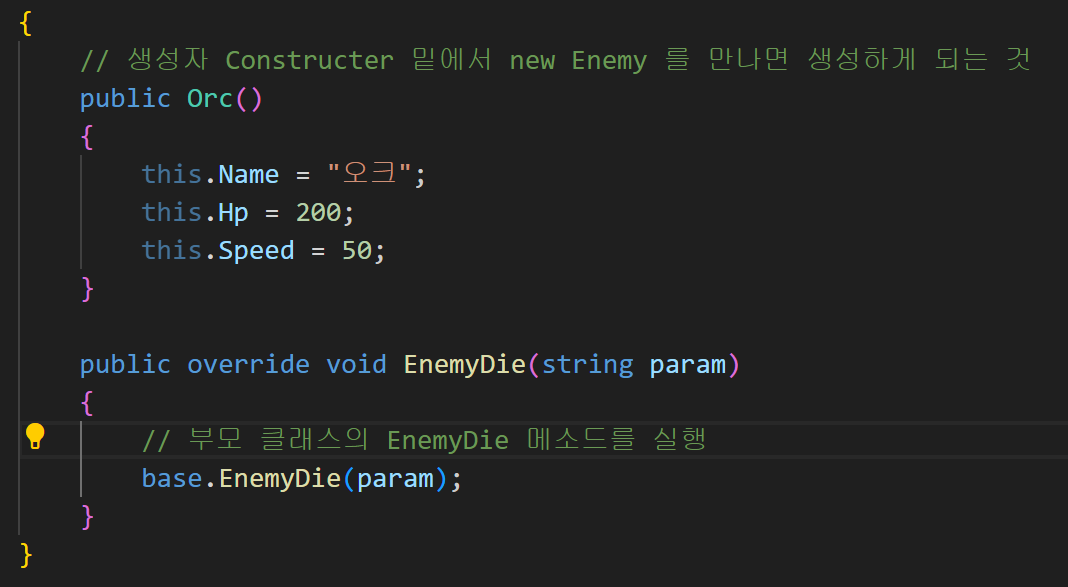
이거였다가~~



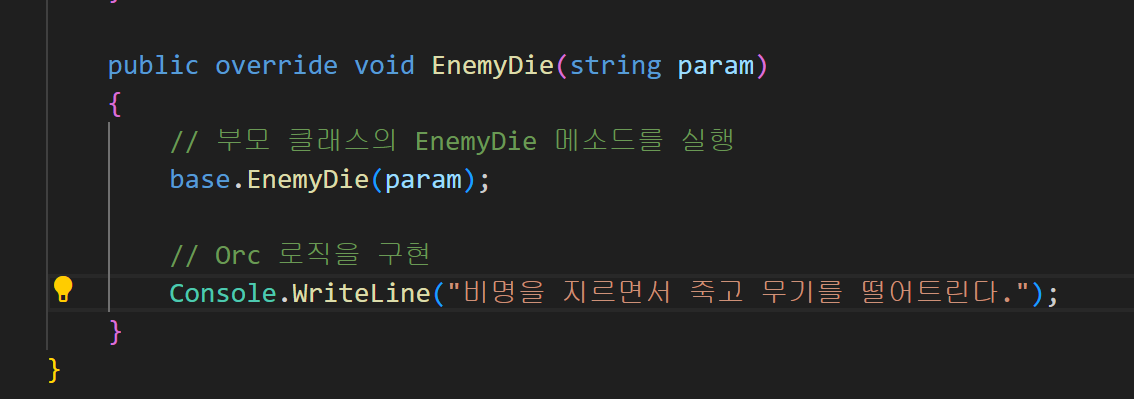
이렇게 바꿔썼음

Public virtual

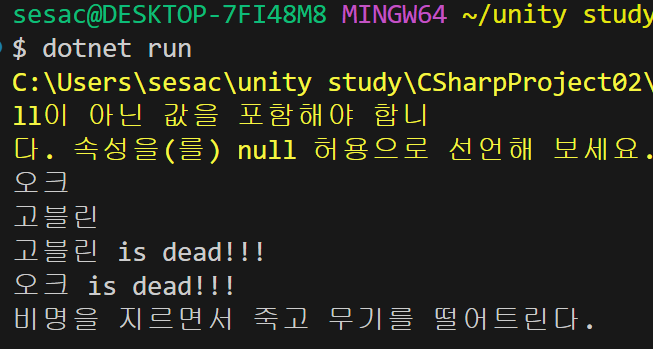
그 밑에 오크 상속받을 것에다가



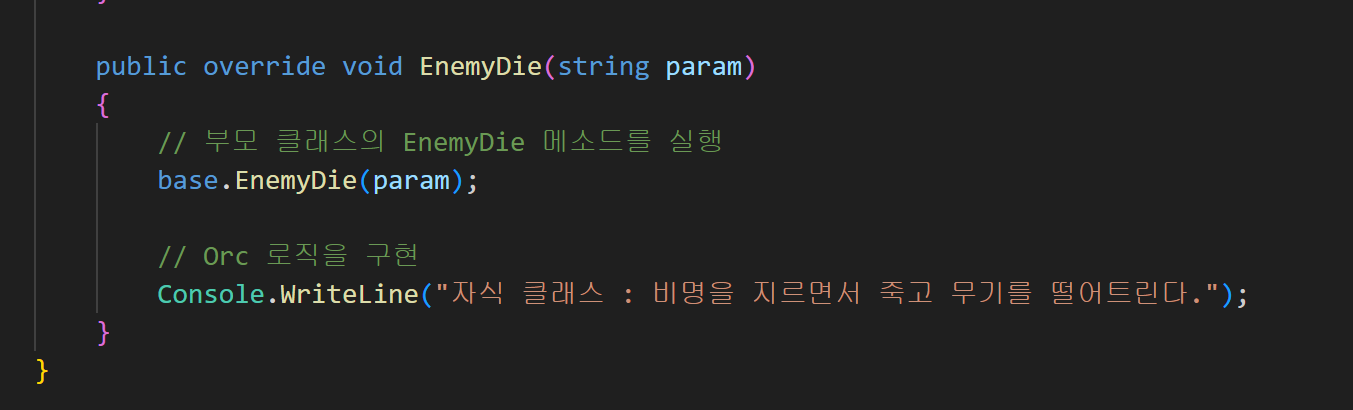
Override 추가

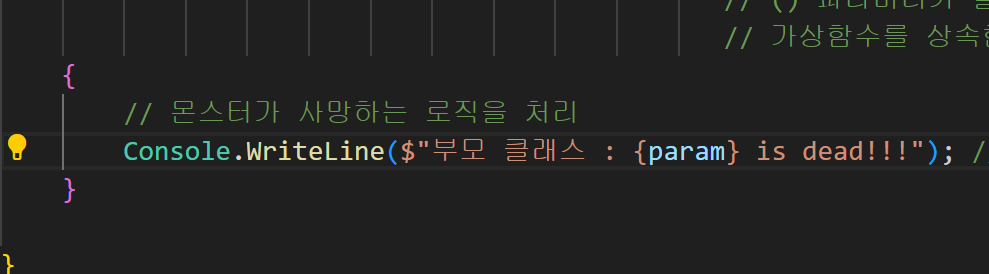


Orc 만의 죽을 때 로직 추가.

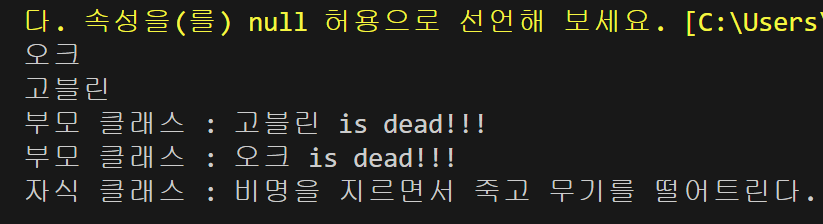
결과.

부모와 자식을 명확하게 하기 위해서 추가

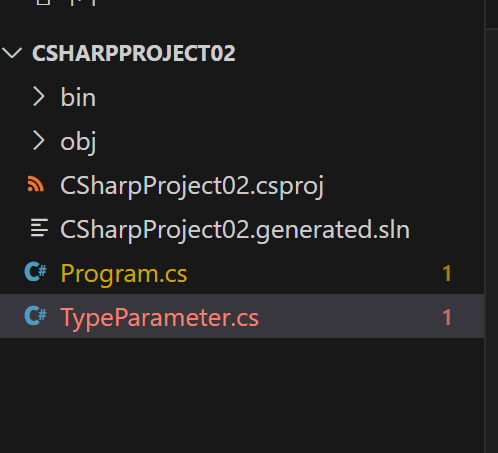


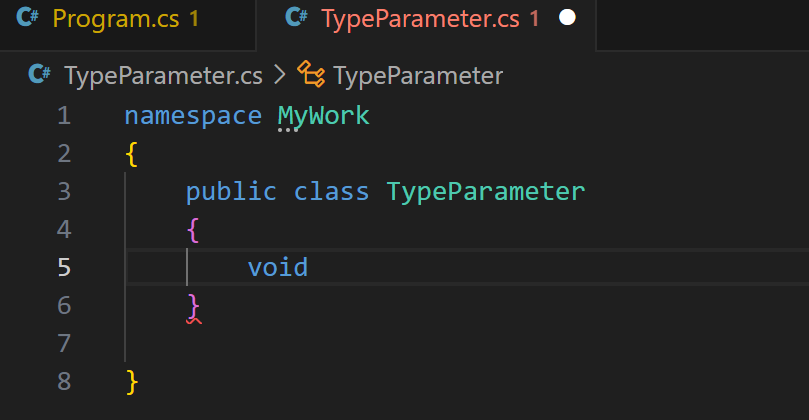


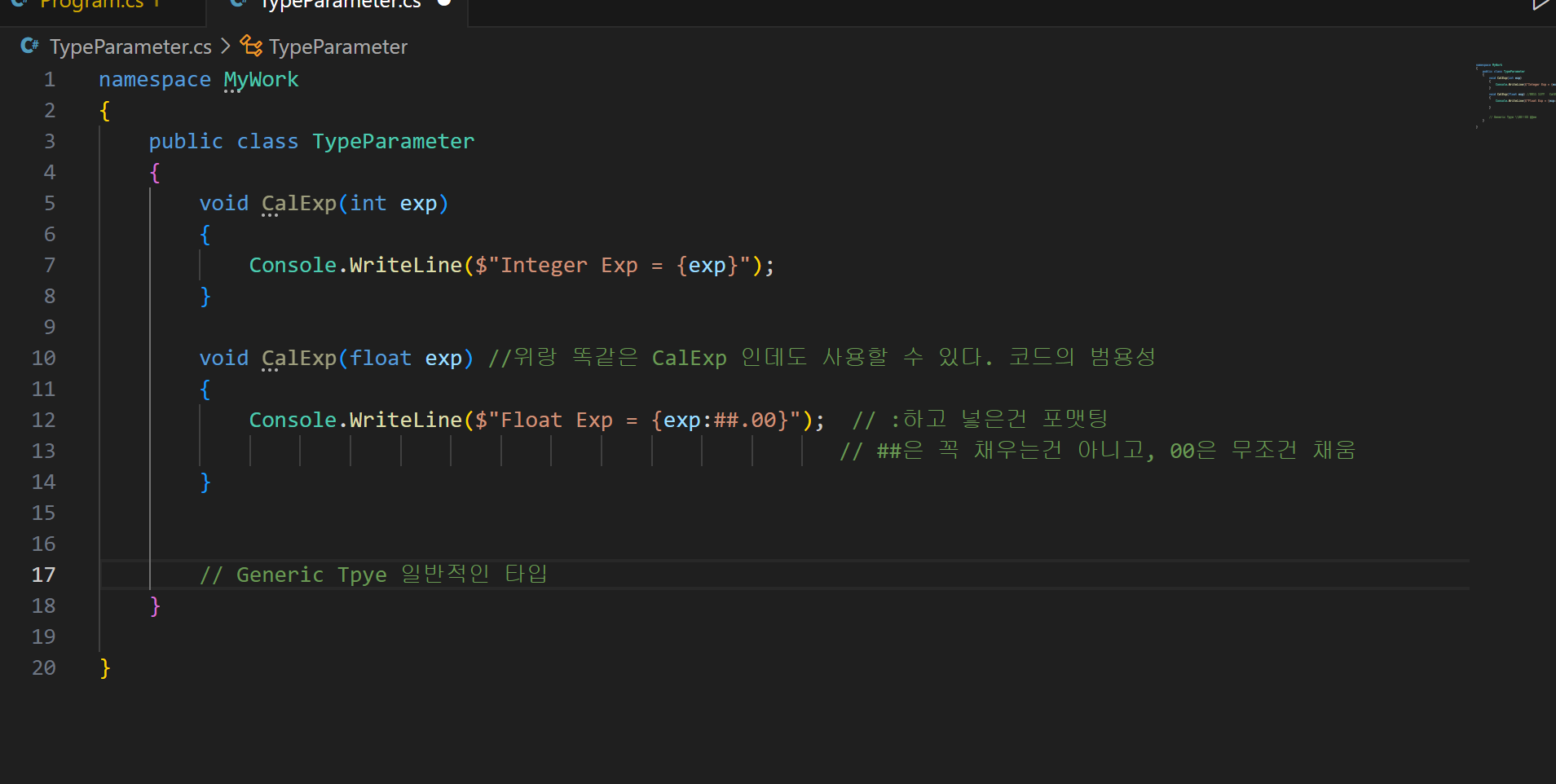
결과

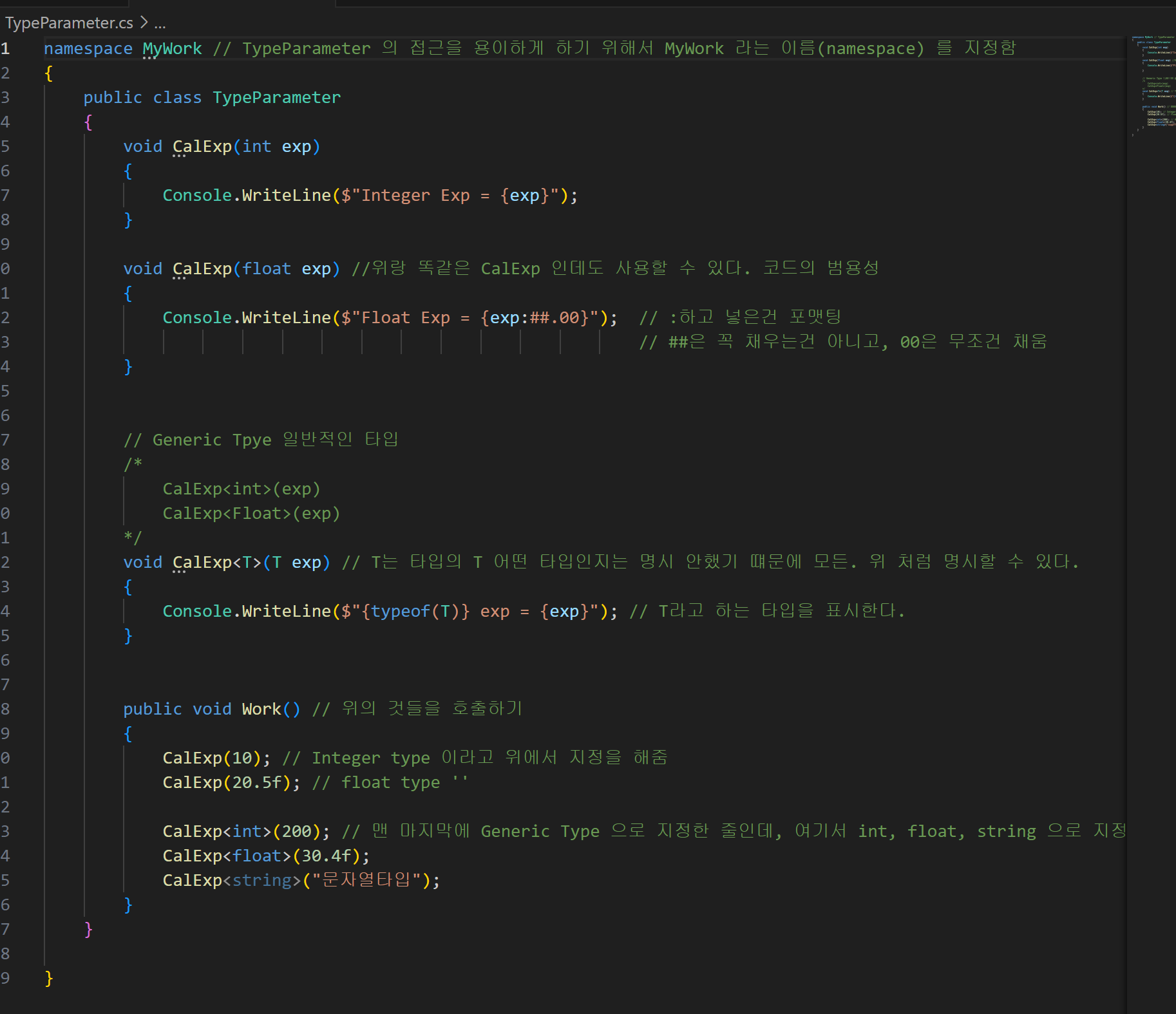


이후에 새 파일

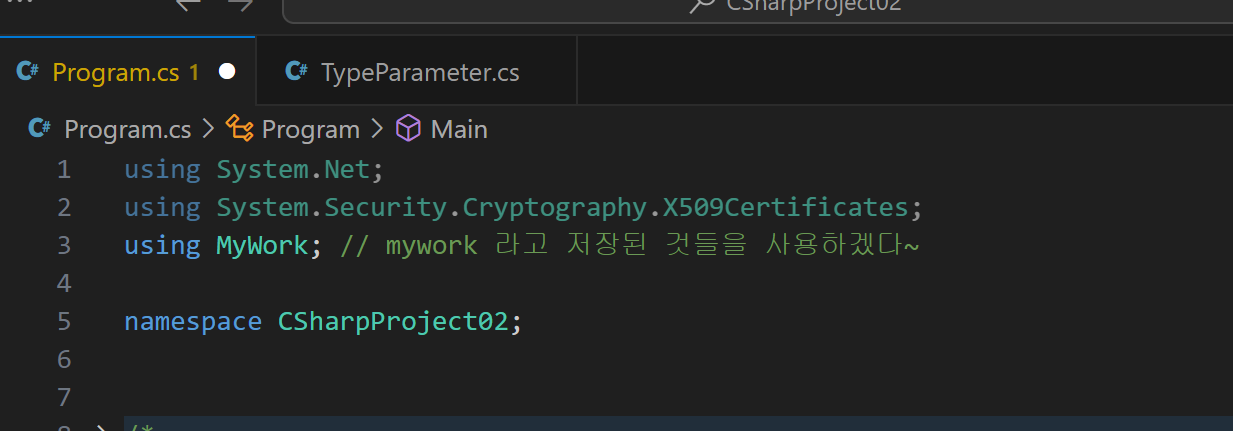




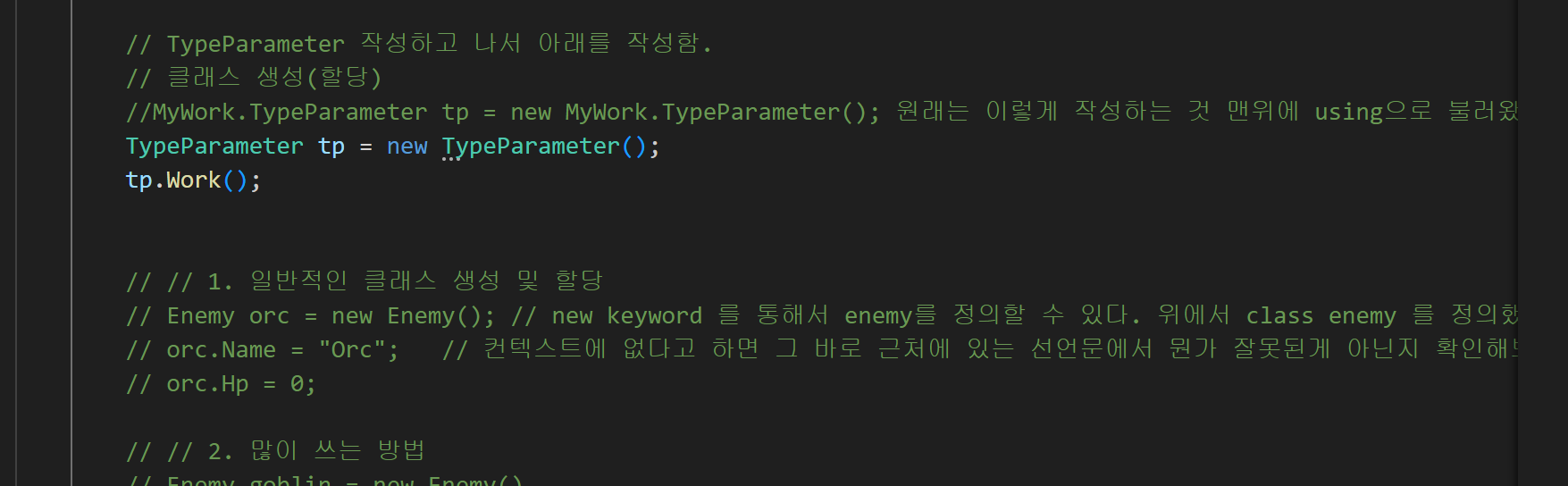




TypeParameter 작성 후에 namespace로 저장해서 다른 program.cs 에서 사용하기에 용이하게 만듦.



맨 위에 using 으로 MyWork 를 불러옴.



이렇게 작성해서 TypeParameter 를 불러와서 작성할 수 있다.

-------------------------여기까지가 VS CODE-----------------------------------------