Abstract Classes & Methods

Wednesday, February 10, 2021 10:39 PM

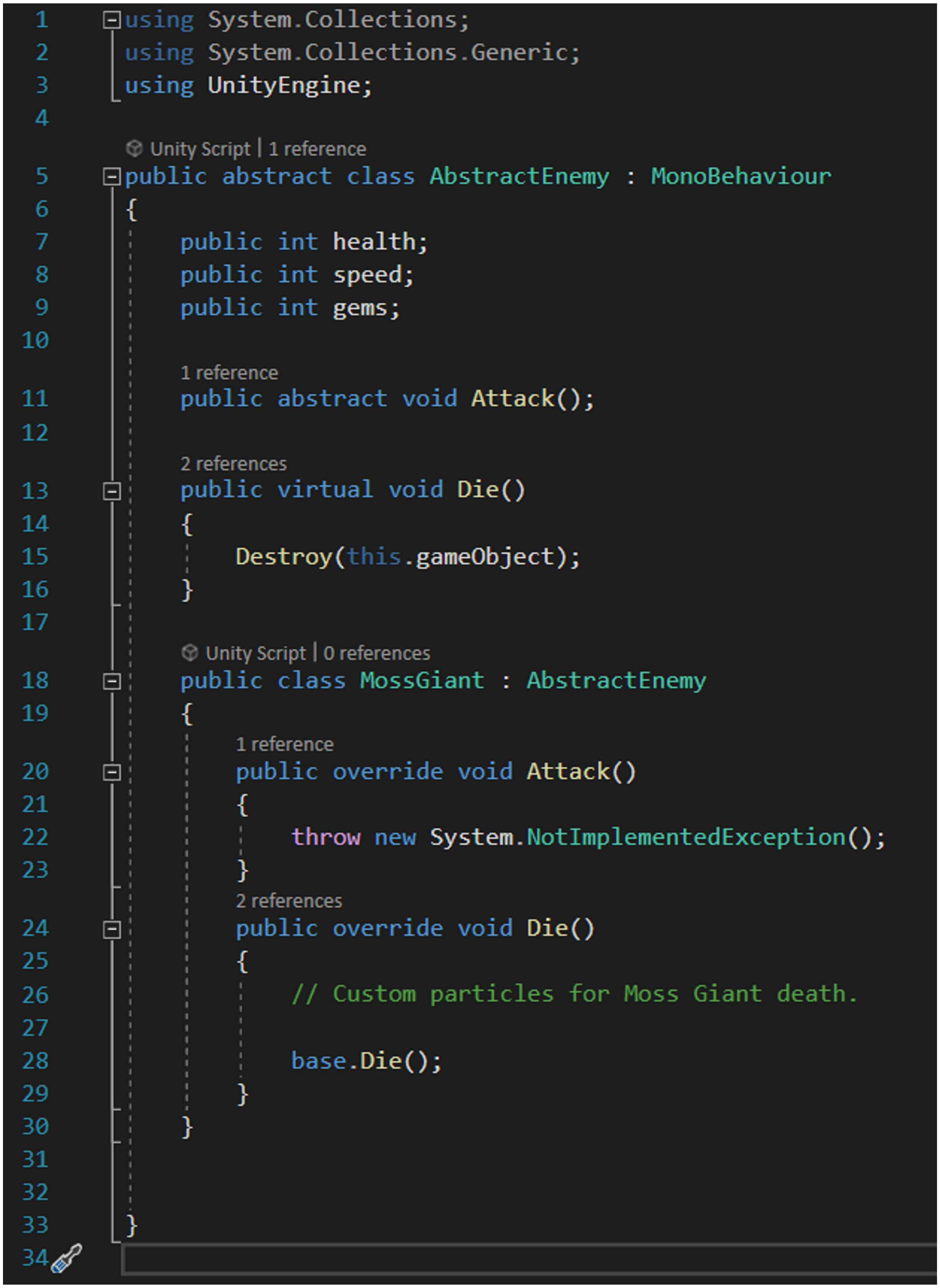
Abstract Classes force inheritance allowing the creation of easily manageable and maintainable code.
Abstract Methods force implementation.

Create a partial template for a class where the inherited implementation finishes the class.
Cannot be added to a **gameObject.**
Cannot be instantiated.
Think of abstract classes as a partial template.

(**Abstract)**
Enemy

- ☐ **int Speed**
- ☐ **int Health**
- ☐ **int Gems**
- ☐ **Attack()**

Moss Giant	**Skeleton**	**Spider**
~~int Speed~~	~~int Speed~~	~~int Speed~~
~~int Health~~	~~int Health~~	~~int Health~~
~~int Gems~~	~~int Gems~~	~~int Gems~~
~~Attack()~~	~~Attack()~~	~~Attack~~



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 1 reference
public abstract class AbstractEnemy : MonoBehaviour
{
    public int health;
    public int speed;
    public int gems;

    1 reference
    public abstract void Attack();

    2 references
    public virtual void Die()
    {
        Destroy(this.gameObject);
    }

    Unity Script | 0 references
    public class MossGiant : AbstractEnemy
    {
        1 reference
        public override void Attack()
        {
            throw new System.NotImplementedException();
        }
        2 references
        public override void Die()
        {
            // Custom particles for Moss Giant death.

            base.Die();
        }
    }


}
```