【Unity基础教程】重点知识汇总

（二）

C#三大核心特性封装、继承与多态

视频链接:https://www.bilibili.com/video/BV1GZDtYvEj1?vd_source=90c7d8485752ee5a35ccafa7923a69bf

封装（概念）

概念（通俗解释）：想象一下，你有一个装满宝物的箱子。这个箱子就是一个**类**（Class，一种**用户自定义的数据类型**）。里面的宝物就是**数据**，而你**打开箱子**、**关上箱子**以及**从箱子中拿取宝物**的这个动作就是**操作数据的方法**。封装就是把这个箱子**关好**，只留几个特定的口子（**公开的方法**），让你可以**安全的**拿取宝物，而别人是不能随便把手伸进箱子里面乱翻的。这样做的好处是你的宝物更加安全了，不会被别人弄坏或偷走。而且你可以随时改变箱子里宝物的摆放方式（**内部的实现细节**），只要不影响那几个口子的使用方法即可。这样既降低了别人使用你箱子的**难度**，也方便你自己**修改和维护**箱子里面的东西。

封装（实际应用）

```csharp
/// <summary>
/// 定义一个玩家类 父类 装满宝物的箱子
/// </summary>
public class Player
{
    // 声明一个私有的字段 用于存储玩家的生命值 箱子里的宝物
    private int health;
    // 公共属性 用于访问和修改玩家的生命值 特定的口子
    public int Health
    {
        // 获取玩家当前的生命值 从箱子中拿取宝物
        get { return health; }
        // 设置玩家的生命值 同时确保这个生命值在0-100之间 往箱子中放入宝物
        set { health = Mathf.Clamp(value, 0, 100); }
    }
    // 构造函数 用来在初始化玩家对象时设置默认的生命值 100
    public Player()
    {
        health = 100;
    }
    // 公共方法 用来减少玩家生命值并输出当前的生命值
    public void TakeDamage(int damage)
    {
        // 减少生命值
        Health -= damage;
        // 输出
        Debug.Log("玩家生命值: " + Health);
    }
}
```

```csharp
public class GameTest : MonoBehaviour
{
    void Start()
    {
        // 创建玩家对象并测试受伤
        Player player = new Player();
        player.TakeDamage(20);
    }
}
```

说明：player不能直接访问health，只能通过Health方法获取和设置health，并且必须按照规定进行设置（0-100）。

继承（概念）

概念（通俗解释）：把继承看作是一个**家族继承**。比如爷爷有一些财产（**属性**）和技能（**方法**），爸爸作为爷爷的儿子，可以继承爷爷的财产和技能，同时爸爸还可以自己再去挣更多的财产，学习更多新的技能。然后儿子又可以继承爸爸的东西，并且也能发展自己独特的东西。这样一代代传承下来，就**避免了每一个人都要从头开始积累财产和学习技能**。大大提高了效率，而且通过这种方式，**家族成员之间的关系也很清晰。大家都知道自己从哪里继承了什么东西，也方便管理和扩展家族的财富和能力。**

继承（实际应用）

```csharp
/// <summary>
/// 定义一个敌人类 继承自玩家类
/// </summary>
public class Enemy : Player
{
    // 自己特有的属性 表示敌人的类型
    public string EnemyType { get; set; }
    // 构造函数 初始化敌人的类型
    public Enemy(string enemyType)
    {
        EnemyType = enemyType;
    }
    // 自己的攻击方法
    public void Attack()
    {
        Debug.Log(EnemyType + "被攻击啦！");
    }
}
```

```csharp
public class GameTest : MonoBehaviour
{
    void Start()
    {
        // 创建敌人对象并测试攻击和受伤
        Enemy enemy = new Enemy("Goblin");
        enemy.Attack();
        enemy.TakeDamage(30);
    }
}
```

说明：Enemy（爸爸）继承了Player（爷爷）的TakeDamage方法并且自己创新出了一个Attack方法（学习新的技能）。

多态（概念）

概念（通俗解释）：就像是一个**万能的遥控器**，可以控制不同的家用电器。虽然这些家用电器的功能和操作方式不完全一样，但是通过万能遥控器上的几个通用的按钮（**同一个操作**）可以让不同的电器做出**不同的反应**，比如按一下“开”按钮，电视机就打开了，空调就开始制冷，音响就开始播放音乐。这样就很方便，**不需要为每一个电器都准备一个专门的遥控器**。在编程中也一样，通过多态可以用同一个方法调用不同对象的不同实现，使得代码**更加简洁**和**灵活**，也更易于**扩展**和**维护**。

多态（实际应用）

```csharp
/// <summary>
/// 定义一个Boss敌人类 继承自敌人类
/// </summary>
public class BossEnemy : Enemy
{
    // 构造函数 初始化Boss的类型
    public BossEnemy() : base("Boss")
    {

    }
    // 覆盖父类 Enemy类中的Attack方法 用于隐藏这个父类的方法而不是重写
    public new void Attack()
    {
        Debug.Log("Boss向玩家发出猛烈攻击！！");
    }
}
```

```csharp
public class GameTest : MonoBehaviour
{
    void Start()
    {
        // 创建Boss敌人对象并测试攻击和受伤
        BossEnemy bossEnemy = new BossEnemy();
        bossEnemy.Attack();
        bossEnemy.TakeDamage(50);
    }
}
```

说明：BossEnemy对Enemy里面的Attack方法进行了覆盖，同一个方法输出了不同的内容。（如果不覆盖，那么输出的是Boss被攻击啦！）

【Unity基础教程】重点知识汇总

（二）

C#三大核心特性封装、继承与多态

视频链接:https://www.bilibili.com/video/BV1GZDtYvEj1?vd_source=90c7d8485752ee5a35ccafa7923a69bf