第9章

常用类

思政目标：

- 理解课程中对常用类的讲解，树立理论联系实际的意识
- 通过面向对象程序设计方法的学习，提高学生的抽象思维能力

技能要求：

- 掌握面向对象程序设计的基本概念
- 了解常用类概念及用法
- 理解并掌握抽象类的用法
- 掌握接口与抽象类的关系
- 了解匿名类

实践目标：

- 根据示例，对所学的概念进行验证
- 熟练应用面向对象方法实现编程设计

- 抽象类
- 接口
- 匿名类
- 结构
- Object类

01
PART

抽象类

抽象类

抽象类的概念

抽象类是只表达一个抽象的概念，是作为其派生类的一个基类的类。抽象类使用abstract修饰，抽象类不能被实例化，即抽象类没有实例存在。

抽象方法

抽象方法是只有方法声明但没有方法实现的一个空方法，使用abstract修饰。抽象的方法隐式为虚方法，必须被覆盖，即若某个类继承了一个抽象类，则该类一定要实现基类中的所有抽象方法。另外，抽象方法必须声明在抽象类中。

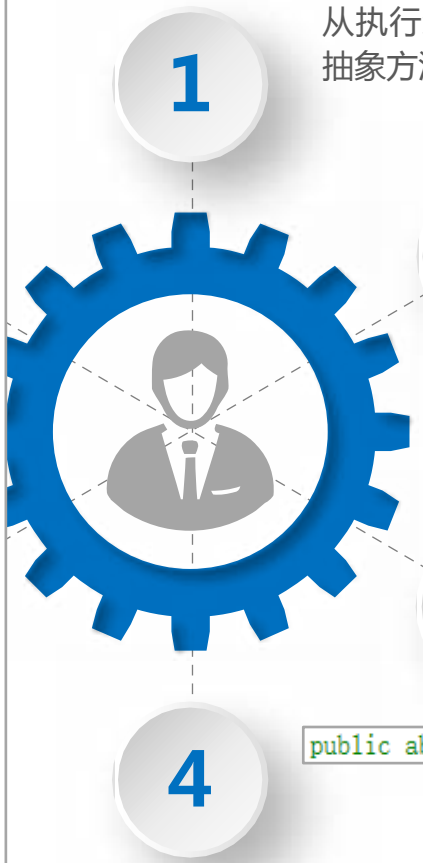
实例

设计3个类：star（明星）类、singer（歌星）类、filmStar（影星）类。其中，star类可以设计为singer类和filmStar类的基类。给这些类都设计一个用于介绍自己的方法Introduce()，由于star类的Introduce()方法在singer类和filmStar类并不是很适用，此时，可以在基类star中将该方法声明为抽象方法，star类自然也要声明为抽象类，然后在singer类和filmStar类中对Introduce()方法进行完善（重写）。

抽象类

```
//基类
2 个引用
public abstract class star
{
    4 个引用
    public abstract void Introduce();       //抽象方法只有声明没有方法体
}
1 个引用
class singer: star
{
    //使用override重写基类同名方法
    4 个引用
    public override void Introduce()
    {
        Console.WriteLine("我是一个歌星！");
    }
}
1 个引用
class filmStar: star
{
    4 个引用
    public override void Introduce()
    {
        Console.WriteLine("我是一个影星！");
    }

    0 个引用
    static void Main(string[] args)
    {
        //star s = new star();                  //错误,抽象类不能实例化
        singer sr = new singer();
        sr.Introduce();
        filmStar fs = new filmStar();
        fs.Introduce();
        Console.ReadLine();
    }
}
```



结果

从执行结果可以知道，子类实现了基类中的抽象方法，能够顺利输出

注意事项

（1）不确定某个类的方法，但知道其实现的细节是由子类决定，那么就可以把它定义为abstract。

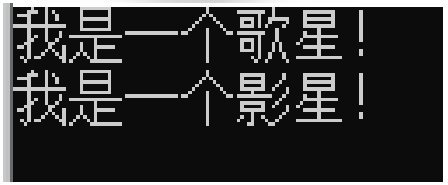
（2）抽象方法必须声明在抽象类中，但抽象类不一定包含抽象方法。

（3）声明抽象方法时，不能使用virtual、static、private修饰符

（4）抽象类不仅可以包含抽象方法，还可以包含虚方法。

（5）抽象类中还可以有抽象属性，实例如下

```
public abstract string Name { get; set; }       //声明一个可读可写的属性Name
```



02
PART

接口

接口

接口的概念

接口，可理解为抽象类的一种特例。在接口中，所有的方法都必须是抽象的，而且接口没有实现代码，只有声明语句。

接口的作用

通过接口，可以定义类的原形，不用定义类的实现。抽象类中要求其子类必须实现其抽象方法，接口则要求实现该接口的类，必须实现接口内声明的所有成员（也可以理解为实现该接口的类都必须具有相同的形式），因此，接口被看作是类和类之间的协议。在C#中，虽然一个类只能继承一个父类，但可以实现多个接口，借助这个特性，可以间接实现多重继承。

接口

声明接口

声明接口成员时，接口成员不允许有任何修饰符，如public、private等。
接口的声明语法

```
[修饰符] interface 接口名 [ ：父接口列表]
    {
     //接口体;
    }
```

重载

声明一个接口，接口包含一个属性和一个方法，其中方法有两个重载版本

```
interface star
{
    //private string Name { get; set;}       //错误,不能使用private修饰符
    0 个引用
    string Name { get; set; }
    //public void Introduce( );              //错误,不能使用public修饰符
    0 个引用
    void Introduce();
    0 个引用
    void Introduce(string sName);
}
```

一个类实现接口的形式

```
[修饰符] class类名 ： 接口1, 接口2, …
    {
     //类体;
    }
```

定义一个singer类实现该接口

```
class singer:star
{
    2 个引用
    public string Name
    {
        get;
        set;
    }
    2 个引用
    public void Introduce()
    {
        Console.WriteLine("我的名字叫：{ 0}", Name);
    }
//此处变量名不一定sName,可以自行修改,但参数类型不得修改
    1 个引用
    public void Introduce(string sName)
    {
        Console.WriteLine("我的名字叫：{ 0}", sName);
    }
}
```

接口

定义一个singer类实现该接口

```
class singer:star
{
    2 个引用
    public string Name
    {
        get;
        set;
    }
    2 个引用
    public void Introduce()
    {
        Console.WriteLine("我的名字叫：{ 0}", Name);
    }
//此处变量名不一定sName,可以自行修改,但参数类型不得修改
    1 个引用
    public void Introduce(string sName)
    {
        Console.WriteLine("我的名字叫：{ 0}", sName);
    }
}
```

解析

由于已经在接口中定义了属性和方法，故在此一定要实现，当然也可继续在该类中添加需要的成员。

```
static void Main(string[] args)
{
    singer sr = new singer();
    sr.Name = "梅艳芳";
    sr.Introduce();
    sr = new singer();
    sr.Introduce("张国荣");
    Console.ReadLine();
}
```

```
我的名字叫：梅艳芳
我的名字叫：张国荣
```

03
PART

匿名类

匿名类

匿名类的概念

没有名字的类就是匿名类型

创建匿名类

```
new {name1 = value1, name2 = value2, …}
```

解读

其中，name1、name2最终成为该匿名类型的属性，value1、value2成为相应的属性值。表面上，匿名类型没有名称，实则编译器会在内部给它分配一个名称，故匿名类型仍遵从C#的强类型检查规则。

引用

由于匿名类型没有名字，所以若想引用它，就需要借助隐式类型变量

```
new {Max = 100, Min = 0};
```

此处所得到得匿名类型有两个属性，分别为Max和Min，值分别为100和0。可以使用myOb.Max，myOb.Min来访问这两个属性

引用如下

```
var myOb = new {Max = 100, Min = 0};
```

在使用匿名类型的过程中，需要注意以下六个问题：

（1）程序编写人员不能决定匿名类型的名称。

（2）匿名类型是隐式封闭的。

（3）匿名类型的字段和属性总是只读的。

（4）匿名类型继承自System.Object。

（5）匿名类型的实体创建只能使用默认的构造函数，不能自定义。

（6）匿名类型中不支持事件、自定义方法、自定义操作符以及自定义重写。

04
PART

结构

结构

01

结构概念

结构类型是一种可封装数据和相关功能的值类型 ，它与类具有很多相同或者类似特征。

02

结构与类的不同之处

结构属于值类型，声明它使用的是关键字struct；而类属于引用类型，声明它使用的是关键字class

03

声明结构

```
struct 结构名 ： 接口
    {
        成员声明;
    }
```

注意问题

（1）结构不能声明无参数构造函数，只能自行定义有参构造函数。

（2）与类一样，结构的成员有字段、属性、方法、构造函数、事件等，但结构不能定义析构函数。

（3）结构不能继承其他类或者结构，即结构不能作为类的的基础类型，但是结构可以实现接口。所有结构的直接父类就是System.ValueType。

（4）结构创建对象可以像类实例化一样使用new，当然也可以不使用new。使用new时，会调用指定的构造函数完成一些初始化工作；不使用new时，不会调用构造函数，此时，需要自己手工完成初始化工作。