Lab5 面向对象与 C#基础

学习目标：

1.理解继承

2.掌握数据类型强制转换的方法

3.理解访问修饰符

一、面向对象三大特征

面向对象的三个基本特征是封装、继承和多态。

1、封装是指将客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。简单的说，一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体。

2、继承是指可以让某个类型的对象获得另一个类型对象的属性和方法。通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程，就是从一般到特殊的过程。

3、多态是指一个类实例的相同方法在不同情形有不同的表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，这些操作可以通过相同的方式予以调用。

二、继承

C#中提供了类的继承机制，但只支持单继承，而不支持多重继承，即在 C#中一次只允许继承一个类，不能同时继承多个类。利用类的继承机制，可以通过增加、修改或替换类中的方法对这个类进行扩充，以适应不同的应用需求。继承使子类可以从父类自动地获得父类所具备的特性，故可以大大节省代码，提高代码的可重用性。

实现继承的语法格式如下：

[访问修饰符]class 类名:基类名

{

类成员;

}

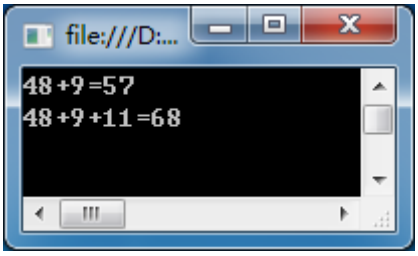
【实例】继承。

程序代码如下：

```
class MyClass1
    {
        public int X { get; set; }
        public int Y { get; set; }
        public int Add()
        {
            return X+Y;
        }
        public virtual int Subtract()
        {
            return X-Y;
        }
    }
class MyClass2 : MyClass1
```

```
        {
            public int Z { get; set; }
            public int Add2()
            {
                return X + Y + Z;
            }
        }
class Program
    {
        static void Main(string[] args)
        {
            MyClass2 myclass2 = new MyClass2();
            myclass2.X = 48;
            myclass2.Y = 9;
            myclass2.Z = 11;
            Console.WriteLine("48+9=" + myclass2.Add());
            Console.WriteLine("48+9+11=" + myclass2.Add2());
            Console.ReadLine();
        }

    }
```



【分析】

本实例中定义了 MyClass2 类，该类继承于 MyClass1 类并扩展其成员方法 Add2()求三个整数的和。在测试类 Inheritance 中，通过 MyClass2 类的对象调用 MyClass1 类中的 Add()和 Add2()方法，实现了整数求和。

【注意】

在 C#中，类的继承遵循以下原则：

（1）派生类只能从一个类中继承，即单继承；

（2）派生类自然继承基类的成员，但不能继承基类的构造方法；

（3）类的继承可以传递，例如：假设类 C 继承于类 B，类 B 又继承类 A，那么 C 类即具有类 B 和类 A 的成员，可以认为类 A 是类 C 的祖先类。

三、数据类型转换

在程序中，当把一种数据类型的值赋给另一种数据类型的变量时，需要进行数据类型转换。根据转换方式的不同，数据类型转换可以分为自动转换和强制转换，同时，某些值类型之间也可以采用 Convert 类提供的静态方法进行转换。

1.自动转换

自动转换又称为隐式转换，指的是两种数据类型在转换的过程中不需要显式地进行声明。自动转换一般在不同类型的数据进行混合运算时发生，当编译器能判断出转换的类型，而且

转换不会带来精度的损失时，C#语言编译器会自动进行转换。自动转换一般是安全的，不会造成数据溢出或丢失等问题。

要实现自动类型转换，必须同时满足两个条件：

（1）两种数据类型彼此兼容；

（2）目标类型的取值范围大于源类型的取值范围。

进行自动类型转换时，遵循以下规则：

（1）如果参与运算的数据类型不相同，则先转换成同一类型，然后进行运算；

（2）转换时按数据长度增加的方向进行，以保证精度不降低，例如 int 型和 long 型运算时，先把 int 数据转换成 long 型后再进行运算；

（3）所有的浮点运算都是以双精度进行的，即使仅有 float 单精度运算的表达式，也要先转换成 double 型，再做运算；

（4）byte 型和 short 型参与运算时，必须先转换成 int 型；

（5）char 可以隐式转换成 ushort、int、uint、long、ulong、float、double 或 decimal，但不存在从其他类型到 char 类型的隐式转换。

例如：

byte x = 9;

int y = x;

在上面程序中，不需要特殊声明，byte 类型的变量 x 就直接转换成 int 类型，因为 x 是一个 byte 字节型，占 8 位，y 是一个 int 整型，占 64 位，编译器自动转换后，不会损失精度。

而以下代码在编译时将出现错误。

int x = 9;

unit y = x;

虽然 int 和 unit 都占 32 位，但 uint 不能存储负数，因此不能进行自动数据类型转换。

2.强制转换

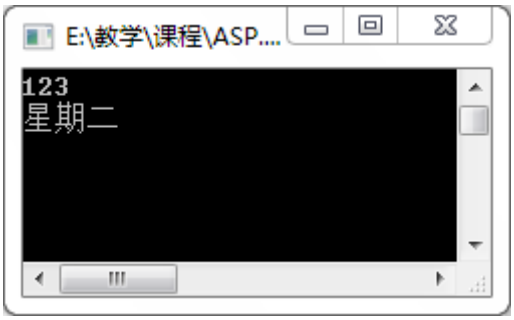
强制转换又称为显式转换，指的是两种数据类型之间的转换需要显式地进行声明。当两种类型彼此不兼容，或者目标类型取值范围小于源类型时，就需要进行强制转换。强制转换的语法格式如下：

源类型　变量 =(目标类型) 值

【实例】使用强制转换方法，分别将 double 类型强制转换成 int 类型，int 类型强制转换为 Weekdays 类型。

程序代码如下：

```
class Convert1
{
    static void Main(string[] args)
    {
        double x = 123.45;
        int y = (int)x;
        Weekdays w = (Weekdays)2;
        Console.WriteLine(y);
        Console.WriteLine(w);
        Console.Read();
    }
}
enum Weekdays { 星期日, 星期一, 星期二, 星期三, 星期四, 星期五, 星期六 };
```



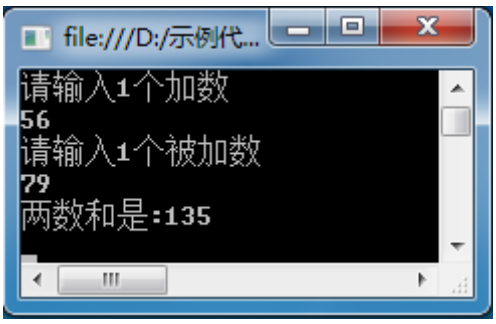
3.Convert 类的方法

Convert 类可以将一个基本数据类型转换成另一个基本数据类型。

【实例】使用 Convert 类实现将 string 类型转换为 int 类型。

程序代码如下：

```
class Convert2
{
    static void Main(string[] args)
    {
        Console.WriteLine("请输入1个加数");
        string a = Console.ReadLine();
        Console.WriteLine("请输入1个被加数");
        string b = Console.ReadLine();
        int x = Convert.ToInt32(a);
        int y = Convert.ToInt32(b);
        int re = x + y;
        Console.WriteLine("两数和是:" + re);
        Console.ReadLine();
    }
}
```



【注意】

（1）待转换数据不是单个变量时，类型和待转换数据都必须加圆括号，例如，(int)(x+y)，表示把 x+y 的结果转换为 int 型；

（2）无论是强制转换或是自动转换，都只是临时性转换，不会改变变量声明时对该变量定义的类型；

（3）当被转换的目标为字符串时，C#内置的简单类型均自带 Parse 方法，调用该方法可自动解析字符串，并转换为指定的数据类型。例如，int x = int.Parse(“2016.9”);

（4）将变量转换为字符串时，C#数据类型均带有 ToString 方法，调用该方法可将数据类型转换为对应的字符串。例如，int x = 2016;string y = a.ToString();

四、访问修饰符

访问修饰符用于限定外界对类和方法的访问权限。在 C#中，访问修饰符共有 4 种，分别是 public、protected、internal 和 private，使用这 4 种访问修饰符可以组合成 5 个可访问级别，访问级别从高到低描述如下。

（1）public：公有访问，最高访问级别，访问不受任何限制。

（2）protected：保护访问，只限于本类和子类访问，实例不能访问。

（3）internal：内部访问，只限于本项目内访问，其他不能访问。

（4）protected internal：内部保护访问，只限于本项目中的类或子类访问，其他不能访问。

（5）private：私有访问，最低访问级别，只限于在声明它们的类和结构中才可以访问，子类，实例都不能访问。