实验三 继承与多态编程练习

【实验目的】

1. 理解继承的含义，掌握派生类的定义方法和实现；
2. 理解虚函数在类的继承层次中的作用，虚函数的引入对程序运行时的影响，能够对使用虚函数的简单程序写出程序结果。
3. 编写体现类的继承性（成员变量，成员方法，成员变量隐藏）的程序；
4. 编写体现类多态性（成员方法重载，构造方法重载）的程序。

【实验要求】

1. 写出程序，并调试程序，要给出测试数据和实验结果。
2. 整理上机步骤，总结经验和体会。
3. 完成实验日志和上交程序。

【实验内容】

1.进一步理解继承的含义

新类可从现有的类中产生，并保留现有类的成员变量和方法并可根据需要对它们加以修改。新类还可添加新的变量和方法。这种现象就称为类的继承。当建立一个新类时，不必写出全部成员变量和成员方法。只要简单地声明这个类是从一个已定义的类继承下来的，就可以引用被继承类的全部成员。被继承的类称为父类或超类（superclass），这个新类称为子类。

2. 进一步理解继承的意义

C# 提供了一个庞大的类库让开发人员继承和使用。设计这些类是出于公用的目的，因此，很少有某个类恰恰满足你的需要。你必须设计自己的能处理实际问题的类，如果你设计的这个类仅仅实现了继承，则和父类毫无两样。所以，通常要对子类进行扩展，即添加新的属性和方法。这使得子类要比父类大，但更具特殊性，代表着一组更具体的对象。继承的意义就在于此。

一、类的继承和基类构造方法的应用

程序功能要求如下：

编写一个学生和教师数据输入和显示程序，学生数据有编号、姓名、班级和成绩，教师数据有编号、姓名、职称和部门。要求将编号、姓名输入和显示设计成一个类 person,并作为学生数据操作类 student 和教师类数据操作类 teacher 的基类。

参考代码如下：

```
public class person
{
    string ID, name;
    public void personin(string id,string name)
    {
        this.ID = id;
        this.name = name;
    }
    public void displayin()
    {
        Console.WriteLine("编号：{0}",ID);
        Console.WriteLine("名字：{0}", name);
    }
}
public class student : person
{
    string classname;
    int grads;
    public student(string classname, int grads)
    {
        this.classname = classname;
        this.grads = grads;
    }
    public void displays()
    {
        Console.WriteLine("班级：{0},成绩：{1}",classname,grads);
    }

}
public class teacher : person
{

    string title,department;
    public teacher(string title, string department)
    {
        this.title = title;
        this.department = department;
    }
    public void displayt()
    {
        Console.WriteLine("职称：{0},部门：:{1}", title, department);
    }
```

```
    }
    public class Tester
    {
        static void Main()
        {
            student su = new student("0601",69);
            su.personin("s00001","Tom");
            su.displayin();
            su.displays();
            teacher tc = new teacher("lecture", "IM");
            tc.personin("t0001","LiLi");
            tc.displayin();
            tc.displayt();
            Console.ReadLine();
        }
    }
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
编号: s00001
名字: Tom
班级: 0601,成绩: 69
编号: t0001
名字: LiLi
职称: lecture,部门: :IM
```

将以上程序尝试改成通过调用基类构造方法的方式来初始化编号和姓名,并总结调用基类构造方法的应用要点。参考代码如下：

```
public class person
    {
        string ID, name;
        public person(string id,string name)
        {
            this.ID = id;
            this.name = name;
        }
        public void displayin()
        {
            Console.WriteLine("编号：{0}",ID);
            Console.WriteLine("名字：{0}", name);
        }
    }
    public class student : person
    {
        string classname;
        int grads;
        public student(string sid,string sname, string classname, int
```

```
grads):base(sid,sname)
        {
            this.classname = classname;
            this.grads = grads;
        }
        public void displays()
        {
            Console.WriteLine("班级：{0},成绩：{1}",classname,grads);
        }

    }
    public class teacher : person
    {

        string title, department;
        public teacher(string tid, string tname,string title, string
department):base(tid,tname)
        {
            this.title = title;
            this.department = department;
        }
        public void displayt()
        {
            Console.WriteLine("职称：{0},部门：:{1}", title, department);
        }

    }
    public class Tester
    {
        static void Main()
        {
            student su = new student("s00001","Tom","0601",69);
            su.displayin();
            su.displays();
            teacher tc = new teacher("t0001","LiLi","lecture", "IM");
            tc.displayin();
            tc.displayt();
            Console.ReadLine();
        }
 }
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
编号：s00001
名字：Tom
班级：0601,成绩：69
编号：t0001
名字：LiLi
职称：lecture,部门：:IM
```

再尝试将 displayin 方法归并到基类构造方法中去，在主函数中注释掉对本方法的调用，可以实现相同的效果么？

可以实现相同的效果

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
编号：s00001
名字：Tom
班级：0601,成绩：69
编号：t0001
名字：LiLi
职称：lecture,部门：:IM
```

二、类的继承和构造函数的灵活应用

1. 编写一个程序计算出球、圆柱和圆锥的表面积和体积。要求：

定义一个基类圆，至少含有一个数据成员半径；

定义基类的派生类球、圆柱、圆锥，都含有求体积函数，可以都在构造函数中实现，也可以将求体积和输出写在一个函数中，或者写在两个函数中，请比较使用。

定义主函数，求球、圆柱、圆锥的和体积。

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
请输入球半径：5
球的体积是：392.69908169872417
请输入圆柱体高度：5
圆柱体的体积是：392.69908169872417
请输入圆锥体高度：5
h2=5,r=5
圆锥体的体积是：130.89969389957471
```

```
class MyCircle
  {
      double r;
      public MyCircle(double r)
      {
          this.r = r;
      }
  }
  class MyBall : MyCircle
  {
      double volumn;
```

```
    public MyBall(double r)
        : base(r)
    {
        volumn = Math.PI * r * r * r;
        Console.WriteLine("球的体积是：{0}",volumn);
    }

}
class MyCylinder:MyCircle
{
    double  volumn;
    public MyCylinder(double r,double h)
        : base(r)
    {

    }
    public void calvolomn(double r,double h)
    {
        volumn = Math.PI * r * r * h;
    }
    public void print()
    {
        Console.WriteLine("圆柱体的体积是：{0}", volumn);
    }
}
class MyCone:MyCircle
{
   double h,volumn;
    public MyCone(double r, double h)
        : base(r)
    {
        this.h = h;
    }
      public void calvolumn(double r,double h)
      {
          volumn = Math.PI * r * r * h / 3;
          Console.WriteLine("圆锥体的体积是：{0}", volumn);
      }

}
public class Tester
{
    public static void Main()
    {
```

```
            Console.Write("请输入球半径：");
            double r=Convert.ToDouble(Console.ReadLine());
            MyBall mb = new MyBall(r);
            Console.Write("请输入圆柱体高度：");
            double h = Convert.ToDouble(Console.ReadLine());
            MyCylinder mc = new MyCylinder(r,h);
            mc.calvolomn(r,h);
            mc.print();
            Console.Write("请输入圆锥体高度：");
            double h2 = Convert.ToDouble(Console.ReadLine());
            MyCone mo = new MyCone(r, h2);
            Console.WriteLine("h2={0},r={1}",h2,r);
            mo.calvolumn(r,h2);
            Console.ReadLine();
        }
    }
```

三、 类的多态性练习 1

1. 理解类的多态性

类的继承发生在多个类之间，而类的多态只发生在同一个类上。在一个类中，可以定义多个同名的方法，只要确定它们的参数个数和类型不同。这种现象称为类的多态。

多态使程序简洁，为程序员带来很大便利。在OOP 中，当程序要实现多个相近的功能时，就给相应的方法起一个共同的名字，用不同的参数代表不同的功能。这样，在使用方法时不论传递什么参数，只要能被程序识别就可以得到确定的结果。

类的多态性体现在方法的重载（overload）上，包括成员方法和构造方法的重载。

2. 构造方法的重载

方法的重载是指对同名方法的不同使用方式。构造方法的名称和类同名，没有返回类型。尽管构造方法看起来和一般的成员方法没有差别，但它不是方法，也不是类的成员。因此，构造方法不能直接调用，只能由new 操作符调用。

构造方法对于类是十分重要的，对象的初始化任务要靠构造方法来完成。

重载构造方法的目的是提供多种初始化对象的能力，使程序员可以根据实际需要选用合适的构造方法来初始化对象。

3. 多态程序练习：功能要求如下

基类 shape 类是一个表示形状的抽象类，area( )为求图形面积的函数。请从 shape 类派生三角形类(triangle)、圆类（circles）、并给出具体的求面积函数，并在主函数中多态地实现调用。

```
using System;
namespace Variables
{
    //public abstract class shape
    //{
```

```
//    public abstract void MyArea();

//}
public class shape
{
    public virtual void MyArea()
{
    Console.WriteLine("no use");
}
}
public class circle : shape
{
    double r,carea;
    public circle(double r)
    {
        this.r = r;
    }
    public  override void MyArea()
    {
        carea = Math.PI*r * r;
        Console.WriteLine("该图形的面积为{0}",carea);
    }
}
public class triangle : shape
{
    double tarea,hemiline,h;
    public triangle(double hemiline,double h)
    {
        this.hemiline = hemiline;
        this.h = h;
    }
    public override void MyArea()
    {
        tarea = hemiline * h / 2;
        Console.WriteLine("hemiline={0},h={1}",hemiline,h);
        Console.WriteLine("该图形的面积为{0}", tarea);
    }
}
public class Tester
{
    public static void Main()
    {
        shape MyShape;
        double r = Convert.ToDouble(Console.ReadLine());
```

```
            MyShape = new circle(r);
            MyShape.MyArea();
            double h = Convert.ToDouble(Console.ReadLine());
            double hemiline = Convert.ToDouble(Console.ReadLine());
            MyShape = new triangle(hemiline, h);
            MyShape.MyArea();
            Console.ReadLine();
        }
    }
}
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
5
该图形的面积为78.53981633974483
4
5
hemiline=5,h=4
该图形的面积为10
```

思考：

(1) 将类 shape 定义为抽象类含有 MyArea 抽象方法再调试程序，查看效果，并回答抽象方法是否可以含{ }主体？；如果将基类中虚方法关键字 virtual 去掉程序会怎样？

```
3 references
9   public class shape
10  {
        2 references
11      public abstract void MyArea()
12      {
13          Console.WriteLine("no use");
14      }
15  }
    1 reference
16  public class circle : shape
17  {
        3 references | 2 references
18      double r,carea;
        1 reference
19      public circle(double r)
20      {
21          this.r = r;
22      }
```

问题 2　输出　调试控制台　终端　筛选器(例如 text、**/*.ts、

C# Program.cs 2

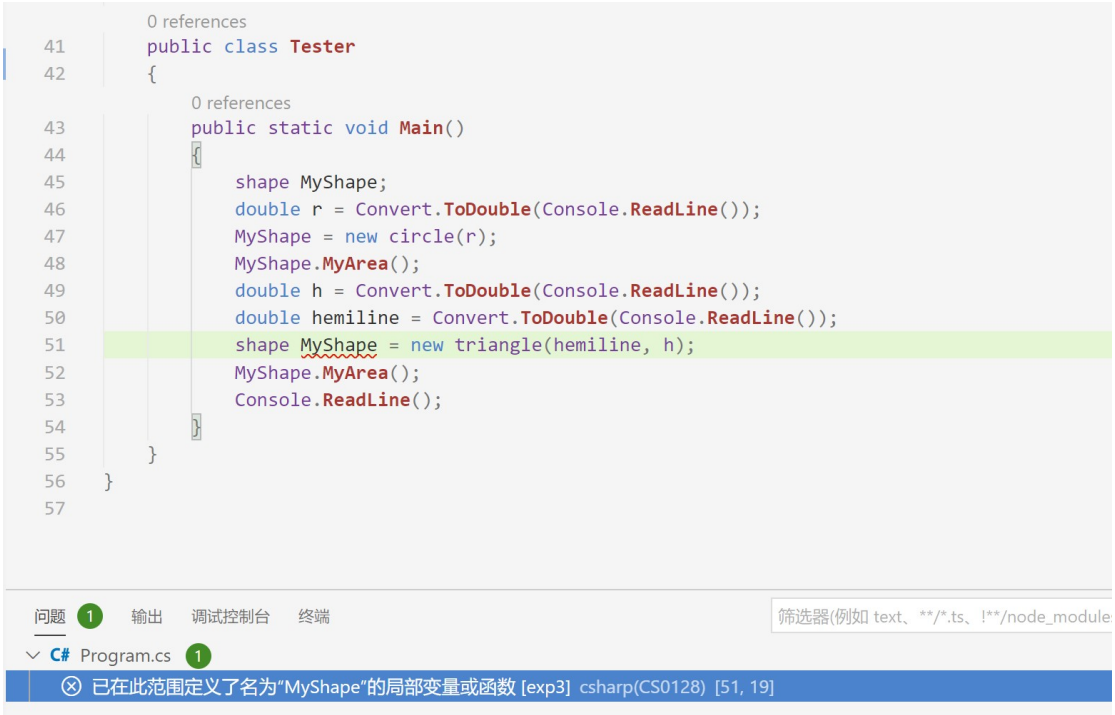
⊗ '"shape.MyArea()"无法声明主体，因为它标记为 abstract [exp3] csharp(CS0500) [11, 30]

⊗ '"shape.MyArea()"是抽象的，但它包含在非抽象类"shape"中 [exp3] csharp(CS0513) [11, 30]


```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
5
该图形的面积为78.53981633974483
4
5
hemiline=5,h=4
该图形的面积为10
```

抽象方法不能包含函数体，且只能在抽象类中定义。如果在基类中去掉 virtual 关键字，在派生类中使用 override。

(2) 将shape MyShape;和MyShape = new circle(r); MyShape = new triangle(hemiline, h);三行改成shape MyShape = new circle(r); shape MyShape = new triangle(hemiline, h);行不行，为什么？



不行，因为已经在当前域中已经定义了MyShape变量。

四、多态处理雇员工资算法

雇员系统，定义雇员基类，共同的属性，姓名、地址和出生日期，子类：程序员，秘书，高层管理，清洁工具有不同工资算法，以多态的方式处理程序。

```
using System;
namespace Variables
{
    public class employee
    {
        internal string name, address, birthday;
        public employee(string name,string address, string birthday)
        {
            this.name = name;
            this.address = address;
            this.birthday = birthday;
        }
        internal double salary;

        public virtual void Salary()
        {
            Console.WriteLine("for overriding");
        }

    }
```

```
public class senior:employee

{
    public senior(string name,string address, string birthday )
        : base(name, address, birthday)
    {
        Console.WriteLine("senior");
    }
    public override void Salary()
    {
        salary = 10000;
        Console.WriteLine("the salary for senoir {0} is{1}",name,salary);



    }
}
public class programmer:employee
{
    public programmer(string name, string address, string birthday)
        : base(name, address, birthday)
    {
        Console.WriteLine("programmer");
    }
    public override void Salary()
    {
        salary = 5000;
        Console.WriteLine("the salary for programmer is{0}", salary);
    }
}
public class secretary:employee
{
    public secretary(string name, string address, string birthday)
        : base(name, address, birthday)
    {
        Console.WriteLine("secretary");
    }
    public override void Salary()
    {
        salary = 2000;
        Console.WriteLine("the salary for secretary is{0}", salary);
    }
}
public class dustman : employee
{
```

```
        public dustman(string name, string address, string birthday)
            : base(name, address, birthday)
        {
            Console.WriteLine("dustman");
        }
        public override void Salary()
        {
            salary = 1000;
            Console.WriteLine("{0}'s birthday is {1}.As a dustman, the salary
is{2}", name,birthday,salary);
        }
    }
    public class Tester
    {
        public static void Main()
        {
            employee emp;
            emp = new senior("Jansy", "zibo", "1970-09-11");
            emp.Salary();
            emp = new dustman("Tom", "zhangdian", "1960-01-22");
            emp.Salary();
            Console.ReadLine();
           }
        }
}
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
senior
the salary for senoir Jansy is10000
dustman
Tom's birthday is 1960-01-22.As a dustman, the salary is1000
```

将主程序中多态变量部分用数组来实现，实现以上功能。

参考代码如下：将主程序中内容用以下内容替换，并运行程序。

```
           employee[] emp = new employee[4];
            emp[0] = new senior("na", "a1", "bd1");
            emp[1] = new programmer("nb", "a2", "bd2");
            emp[2] = new secretary("nc", "a3", "bd3");
            emp[3] = new dustman("nd", "a4", "bd4");
            for (int i = 0; i < 4; i++)
            {
                emp[i].Salary();
            }
            Console.ReadLine();
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
senior
programmer
secretary
dustman
the salary for senoir na is10000
the salary for programmer is5000
the salary for secretary is2000
nd's birthday is bd4.As a dustman, the salary is1000
```

分析本方法在和多态思想结合方面的优势。
多态的意义在于基类给出了一个接口，各个派生类对应不同的实现，有利于提高代码的可读性，在工厂模式中经常使用。

五、Object 类应用

1. object 类的 ToString、GetType、Equials、ReferenceEquials 的使用

程序功能要求：定义一个 Dog 类含有名字变量，重载的构造函数能够初始化有名字和没有名字的对象；属性 Name 实现读取；主方法中测试狗对象的 ToString 方法输出，GetType 方法输出结果的 ToString 方法输出，狗对象间的 Equals 方法，ReferenceEquals 的使用。
参考代码如下：

```
using System;
namespace Animals
{
    class Dog
    {
        private string name;
        public Dog()
        {
            name = "unknown";
        }

        public Dog(string initialName)
        {
            name = initialName;
        }

        public string Name
        {
            get
            {
                return name;
            }
        }
    }
}
```

```
class ObjectTester
{
    public static void Main()
    {
        Type dogType;
        Animals.Dog myDog = new Animals.Dog("Fido");
        Animals.Dog yourDog = new Animals.Dog("Pluto");
        Animals.Dog sameDog = myDog;
        Console.WriteLine("ToString(): " + myDog.ToString());
        dogType = myDog.GetType();
        Console.WriteLine("Type: " + dogType.ToString());

        if(myDog.Equals(yourDog))
            Console.WriteLine("myDog is referencing the same object as yourDog");
        else
            Console.WriteLine("myDog and yourDog are referencing different objects");

        if(myDog.Equals(sameDog))
            Console.WriteLine("myDog is referencing the same object as sameDog");
        else
            Console.WriteLine("myDog and sameDog are referencing different objects");

        if(object.ReferenceEquals(myDog, yourDog))
            Console.WriteLine("myDog and yourDog are referencing the same object");
        else
            Console.WriteLine("myDog and yourDog are referening different objects");
    }
}
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
ToString(): Animals.Dog
Type: Animals.Dog
myDog and yourDog are referencing different objects
myDog is referencing the same object as sameDog
myDog and yourDog are referening different objects
```

2. ToString 方法、Equials 方法重定义练习

程序功能要求：

重定义以上狗类对象的 ToString 方法，返回狗的 name；重定义 Equals 方法，用狗类对象进行比较，若名字相同，则认为狗相等，否则不等；在主方法中测试以上定义。

参考代码如下：

```
using System;
namespace Animals
{
    class Dog
```

```
    {
        private string name;

        public Dog()
        {
            name = "unknown";
        }

        public Dog(string initialName)
        {
            name = initialName;
        }

        public string Name
        {
            get
            {
                return name;
            }
        }

        public override string ToString()
        {
            return "Dog name: " + name;
        }

        public override bool Equals(object obj)
        {
            Dog tempDog = (Dog) obj;

            if(tempDog.Name == name)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}

class ObjectTester
{
```

```
    public static void Main()
    {
        Animals.Dog myDog = new Animals.Dog("Fido");
        Animals.Dog yourDog = new Animals.Dog("Fido");

        Console.WriteLine(myDog);

        if(myDog.Equals(yourDog))
            Console.WriteLine("myDog has the same name as yourDog");
        else
            Console.WriteLine("myDog and yourDog have different names");
    }
}
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
Program.cs(4,11): warning CS0659: '“Dog”重写 Object.Equals(object o) 但不重写 Object.GetHashCode() [C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3\exp3.csproj]
Dog name: Fido
myDog has the same name as yourDog
```

【思考题】

1. 写一个动物基类，具有动物的名称变量，叫虚方法，写出子类猫、狗、牛的叫方法，以多态的方式实现程序；

```
using System;
using System.Reflection;

namespace exp3 {
    public abstract class Animal {
        protected string Name {get;set;}
        public abstract void Bark();
    }

    public class Cat : Animal
    {
        public Cat(string name){
            this.Name = name;
        }
        public override void Bark()
        {
            System.Console.WriteLine($"cat {Name}: miao~~");
        }
    }

    public class Dog : Animal
    {
        public Dog(string name){
```

```
            this.Name = name;
        }
        public override void Bark()
        {
            System.Console.WriteLine($"dog {Name}: wang wang ~~");
        }
    }

    public class Cow : Animal
    {
        public Cow(string name){
            this.Name = name;
        }
        public override void Bark()
        {
            System.Console.WriteLine($"cow {Name}: mou mou~");
        }
    }

    public class AnimalFactory {
        public Animal Create<T>(string name) where T: Animal {
            Animal animal = (Animal)Activator.CreateInstance(typeof(T),
args: new object[]{name});
            return animal;
        }
    }

    class Program {


        public static void Main(){
            AnimalFactory animalFactory = new AnimalFactory();
            Animal animal;
            animal = animalFactory.Create<Cat>("a");
            animal.Bark();
            animal = animalFactory.Create<Dog>("b");
            animal.Bark();
            animal = animalFactory.Create<Cow>("c");
            animal.Bark();
        }
    }
}
```

```
PS C:\Users\cht\Program\homework.gallery\computer science\csharp\exp3> dotnet run
cat a: miao~~
dog b: wang wang ~~
cow c: mou mou~
```