实验3 C#面向对象程序设计

实验类型：验证性实验 要求：必做 学时：4

**一、实验目的**

1. 掌握C#中类的声明和对象定义方法。

2. 掌握C#中类的封装、继承和多态机制。

**二、实验内容**

1、定义一个矩形类Rectangle，实现对矩形类的封装。。

2、编写程序，使用继承和派生。

3、虚方法的定义和重写。

4、运算符的重载。

**三、实验步骤**

1、定义一个矩形类Rectangle，实现对矩形类的封装。新建控制台应用程序项目：

(1) Rectangle类中包括有字段长度length、宽度width，以公开方式访问字段的属性Length和Width;

(2)为Rectangle类添加一个构造函数（带两个参数，分别对应length和width）；

(3)成员方法Area()用来计算矩形面积。

(4)在Main()方法中创建Rectangle类对象，通过属性给长度和宽度赋值，要求输出矩形的长、宽及面积值。

关键代码如下：

namespace Project1

{

internal class sy3\_1

{

static void Main(String[] args)

{

Rectangle r = new Rectangle(10, 20);

Console.WriteLine("矩形的长为：{0:f2}", r.Length);

Console.WriteLine("矩形的宽为：{0:f2}",r.Width);

Console.WriteLine("矩形的面积：{0:f2}", r.area());

}

}

class Rectangle

{

private double length { get; set; }

private double width { get; set; }

public double Length

{

get { return length; }

set { length = value; }

}

public double Width

{

get { return length; }

set { length = value; }

}

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

public double area()

{

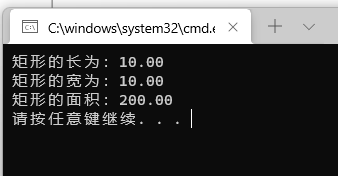
return this.length \* this.width;

}

}

}

运行结果截图如下：



2、编写程序，使用继承和派生。

新建控制台应用程序项目：

(1) 在项目中添加一个Person类，该类有name和age两个公有字段；

(2) 为Person类添加一个构造函数（带两个参数，分别对应name和age）；

(3) 为Person类编写方法ShowInfo，显示Person类的信息；

(4) 增加一个学生Student类，该类从Person类的派生，Student类有一个新的私有的学号字段number；

(5) 为Student类添加一个三个参数的构造函数，并调用父类的构造函数辅助实现对name和age字段的初始化。

(6) 在Student类中为学号number字段添加一个属性；

(7) 在Student类中增加ShowNumber方法用于显示学号信息；

(8) 在Student类中隐藏基类中的ShowInfo方法，方法的功能是输出学生的所有信息，其中name和age的信息要求调用基类的ShowInfo方法输出，学号信息调用ShowNumber方法输出。

(9) 在Main()方法中创建Student的一个实例student，并调用ShowInfo方法输出学生的所有信息。

关键代码如下：

namespace Project1

{

public class Person

{

public string name { get; set; }

public int age { get; set; }

public Person(string name, int age)

{

this.name = name;

this.age = age;

}

public void ShowInfo()

{

Console.Write("姓名：{0}\n 年龄：{1}",this.name,this.age);

}

}

public class Student:Person

{

private string number;

public Student(string name, int age,String number) : base(name, age)

{

this.number = number;

}

public void ShowNumber()

{

Console.WriteLine("\n学号:{0}",this.number);

}

public void ShowInfo()

{

base.ShowInfo();

ShowNumber();

}

}

public class sy3\_2

{

static void Main(string[] args)

{

Student student = new Student("李四",3,"18766");

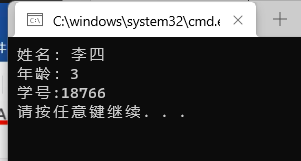
student.ShowInfo();

}

}

}

运行结果截图如下：



3、运算符的重载。

新建项目，在项目中增加类Time，该类有小时、分、秒三个属性；编写构造函数（带三个参数）用于对三个属性进行初始化；在Time类中对运算符“+”进行重载，实现两个时间的相加；在Main()方法中创建一从个Time时间实例，调用重载后的运算符“+”将两个时间相加，输出相加后的结果。

关键代码如下：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project1

{

internal class sy3\_3

{

static void Main(string[] args)

{

Time time1 = new Time(1,42,16);

Time time2 = new Time(2,26,52);

Time result = time1 + time2;

Console.Write("{0}时:{1}分:{2}秒+{3}时:{4}分:{5}秒=", time1.Hours, time1.Minutes, time1.Seconds, time2.Hours, time2.Minutes, time2.Seconds);

Console.WriteLine("{0}时:{1}分:{2}秒", result.Hours, result.Minutes, result.Seconds);

}

}

class Time

{

private int hours=0,minutes=0, seconds=0;

public Time(int hours, int minutes, int seconds)

{

this.hours = hours;

this.minutes = minutes;

this.seconds = seconds;

}

public Time()

{

}

public int Hours

{

get => hours;

set => hours = value;

}

public int Minutes

{

get => minutes;

set => minutes = value;

}

public int Seconds

{

get => seconds;

set => seconds = value;

}

public static Time operator +(Time time1, Time time2)

{

Time result = new Time();

result.seconds = time1.seconds + time2.seconds;

result.minutes = time1.minutes + time2.minutes;

result.hours = time1.hours + time2.hours + result.minutes/60;

result.seconds = result.seconds % 60;

result.minutes=result.minutes% 60;

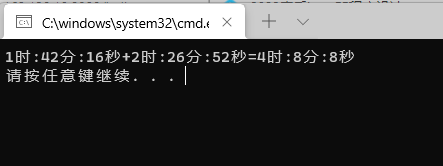
return result;

}

}

}

运行结果截图如下：



4、虚方法的定义和重写。

新建一个控制台应用程序项目

(1) 添加一个Shape类；在Shape类中添加一个公共的双精度常量PI，两个保护的双精度字段x，y；添加一个无参的构造函数，一个有两个参数的构造函数（用于初始化x与y两个字段）；添加一个虚拟的Area方法，用于求各种形状的面积。

(2) 添加一个Rectangle矩形类，继承基类Shape；在Rectangle类中添加一个无参构造函数与一个带两个参数的构造函数，并使用基类的构造函数；重写Shape类的Area方法，计算矩形的面积；

(3) 添加一个Ellipse椭圆类，继承基类Shape；在Ellipse类中添加一个带两个参数的构造函数，并使用基类的构造函数；重写Shape类的Area方法，计算椭圆的面积；

(4) 在Main()方法中，实例化一个Rectangle与一个Ellipse对象，并分别输入矩形的长和宽及椭圆的长半轴和短半轴，再调用各自计算面积的Area方法，计算出面积，并分别输出矩形及椭圆的面积。

关键代码如下：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Project1

{

internal class sy3\_4

{

static void Main(string[] args)

{

Console.WriteLine("请分别输入矩形的长和宽:");

double a = Convert.ToDouble(Console.ReadLine());

double b = Convert.ToDouble(Console.ReadLine());

Rectangle1 r1 = new Rectangle1(a,b);

Console.WriteLine("矩形面积为:"+r1.Area());

Console.WriteLine();

Console.WriteLine("请分别输入椭圆的长半轴和短半轴:");

double c = Convert.ToDouble(Console.ReadLine());

double d = Convert.ToDouble(Console.ReadLine());

Ellipse e1 = new Ellipse(c,d);

Console.WriteLine("椭圆的面积为:"+e1.Area());

}

}

class Shape

{

public const double PI = 3.14159;

protected double x,y;

public Shape()

{

}

public Shape(double x, double y)

{

this.x = x;

this.y = y;

}

public virtual double Area()

{

return x \* y;

}

}

class Rectangle1:Shape

{

public Rectangle1()

{

}

public Rectangle1(double x, double y) : base(x, y)

{

}

public override double Area()

{

// return base.Area();

return x \* y;

}

}

class Ellipse:Shape

{

public Ellipse(double x, double y) : base(x, y)

{

}

public override double Area()

{

// return base.Area();

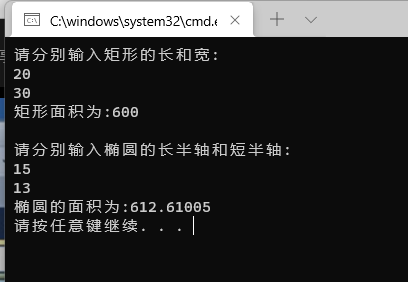
return PI \* x \*y;

}

}

}

运行结果截图如下：



**四、实验小结**

通过本次实验学会了面向对象的封装,继承,多态的机制.学习到属性封装以及类的继承,方法的重载,运算符的重载,虚方法与方法重写,抽象类和抽象方法,C#系统的类型转换.

**五、实验提交**

把本文档以学号姓名sy3为名(形如019301784160张三sy3)保存后提交到指定的ftp处。