МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 5

по дисциплине «Технологии программирования»

Тема: «Наследование классов»

Выполнил: Стычинский С.В., МВИ-121

Проверили: Кузьмина Т. М. и Адаев Р. Б.

Москва 2023

Цель работы

Изучение основных принципов наследования, создание собственной иерархии классов.

Задание

19. Базовый класс «Точка». Классы «Окружность», «Сектор», «Прямоугольник» являются производными от класса «Точка». Во всех классах определить по два метода, вычисляющих площадь фигуры (площадь точки равна нулю, для объемных тел вычисляется площадь полной поверхности). Один метод виртуальный, другой обычный. В классе «Прямоугольник» еще определить метод, вычисляющий диагональ. В классе «Окружность» еще определить метод, вычисляющий диаметр. В классе «Сектор» определить метод, вычисляющий длину дуги.

Теоретический материал

Наследование

Одним из наиболее мощных свойств класса С++, является возможность их расширения с помощью наследования. Наследование – это создание нового класса путём добавления новых членов к уже существующему классу. Класс, к которому добавляются новые члены называется базовым или родительским классом, а вновь созданный класс производным или дочерним классом. Двоеточие после имени класса указывает компилятору на то, что класс является производным от другого класса. За двоеточием следует имя базового класса. Напоминание: методы производного класса могут работать только с открытыми (public ) и защищёнными ( protected и protected internal) членами базового класса, и если классы находятся в одной сборке, то членами, имеющими уровень доступа internal. Но не могут работать с собственными ( private ) членами базового класса.

Пример:

сlass cir: vec {

public int R;

public void fun(void) { return x\*y+R;}

}

Класс cir является наследником класса vec, метод fun использует наравне с полем R , поля базового класса x,y, что возможно в том случае, когда уровень доступности указанных полей либо public, protected. Конструкторы при наследовании Конструкторы не наследуются, но используются при создании объектов производных классов. При создании объекта производного класса, сначала вызывается конструктор базового класса, а затем производного класса. 29 По умолчанию, при создании объектов производного класса вызывается конструктор базового класса без параметров. Что делать, если необходимо использование для базового класса конструктора с параметрами? В этом случае нужно обязательно написать конструктор производного класса и в описании этого конструктора включить дополнительную запись. После имени конструктора в его описании поставить двоеточие, затем, написать ключевое слово base и в скобках значения параметров.

class B{

public int x;

public B(int xx) { x=xx; }

} class C:B{ public C(int xx):base(xx) {}}

В С# нет множественного наследования классов, т.е. для у любого производного класса базовый класс всегда один. Сам базовый класс может быть производным от некоторого другого класса. В качестве базовых типов могут выступать интерфейсы, в этом случае класс может быть наследником многих интерфейсов. Замечание: Некоторые авторы определяют наследование, как создание нового класса на основе существующих. В этом случае, агрегация также подпадает под определение наследования, и ее называют наследованием по модели включение-делегирование. Запрет наследования: классы объявленные как sealed В некоторых случаях требуется запретить создание производных классов. Разработчики хотят, чтобы класс использовался только в том виде, в каком он создан, в этом случае ключевое слово sealed. Класс String пространства имен System является закрытым для наследования. 30 Замечание: слово sealed не запрещает агрегацию. Ссылки базового класса Ссылки базового класса могут указывать на объекты производных классов. Примеры.

B b=new C();// классы В и С определены выше B d=new D(); При работе с объектами производных классов, через ссылки базовых классов, доступны, за одним исключением, только члены, определенные в базовом классе (классе ссылки). Исключение – это виртуальные методы. Если нужно перейти от ссылки базового класса к ссылке производного класса, то надо выполнить преобразование типов. C c; c=(C)d; b=c; Поскольку по ссылке базового класса не всегда ясно на какой объект она указывает, то в С# введены специальные операторы, которые выполняют проверку возможно ли сделать соответствующее преобразование типов. Операторы is и as. Оператор is никогда не генерирует исключение, он работает со всеми типами. Он имеет следующий вид: Ссылка is Тип Оператор is возвращает значение булево типа, true - если объект на который указывает ссылка можно привести к типу, стоящему после is и false – если ссылка нулевая или объект нельзя привести к указанному типу.

Например.

int i=8;

Object o=new Object();

Boolean b=(i is Object); //b будет равно true

Boolean b1=(o is int); //b1 будет равно false Оператор as работает со ссылочными типами, он производит нужное преобразование типов, если оно возможно и возвращает null, если оно невозможно. Команда A a=o as A; будет выполнена успешно, но в ссылку а будет записан null, и любое обращение к а как к объекту, например, a.ToString(); вызовет генерацию исключения. Проверку того, не получила ли ссылка нулевого значения, должен выполнить программист. В нашем примере она будет выглядеть так if(a!=null)a.ToString(); Переопределение методов базового класса. Производный класс наследует все члены базового класса, в том числе все его методы. Во многих случаях требуется изменить работу какого-либо метода.

Изменение метода происходит путем его переопределение в производном классе. Например:

public class A {

public int f() {return 0;}

}

public class B:A {

public int f() {return 1;}

}

Метод f описан и в базовом и в производном классе. В следующем фрагменте программы вызывается метод из класса В:

B b=new B(); int i=b.f();

Но для объекта b можно вызвать метод f базового класса, для этого надо выполнить преобразование типов: ((A)b).f(); или A a=b; a.f(); В обоих случаях будет вызвана метод f базового класса. Для обычных (не виртуальных) методов справедливо следующее правило: 32 При вызове невиртуального метода, метод определяется типом ссылки. Но во многих случаях для работы с объектами производных типов необходимы методы определенные в этих же типах, даже если работа выполняется через ссылки базовых классов. Для решения данной проблемы в объектно-ориентированных языках созданы виртуальные методы. При вызове виртуального метода, метод определяется типом объекта. Определение виртуального метода происходит следующим образом: В базовом классе при определении метода используется ключевое слово virtual, а во всех производных классах, в которых этот метод переопределяется, слово override.

Например:

public class B {

public virtual int f() {return 1;}

}

public class C:B {

public override int f() {return 2;}

}

public class D:C {

public override int f() {return 3;}

}

В следующем фрагменте программы во всех случаях будет вызван метод f из класса D: B b=new D(); int i=b.f(); i= ((C)b).f();

Этапы выполнения лабораторной работы

1. Создать иерархию классов, определенную вариантом задания. Базовый «Точка» должен содержать два метода, вычисляющие площадь, один метод обычный, другой виртуальный, оба эти метода должны возвращать значение 0. Базовый класс может не содержать полей, все производные должны содержать поля, которые позволяют решать поставленные в индивидуальном задании задачи. Все производные классы должны содержать конструкторы, задающие не нулевые значения полям класса.

2. В классе формы(Form1) определить поля, которые являются объектами созданных Вами классов, по одному на каждый класс и еще одно поле t, являющееся ссылкой базового класса «Точка». Ссылка t всегда будет указывать на активный объект, по умолчанию, t указывает на объект базового класса.

3. На форму поместить две панели (компоненты Panel). На них будут размещены кнопки и метки (компоненты Label).

4. Кнопки, расположенные на первой панели (кнопки первой группы), предназначены для выбора активного объекта. Число этих кнопок равно числу полей класса Form1, являющихся объектами Ваших классов. Каждому объекту ставиться в соответствии кнопка. Если кнопке поставлен в соответствие объект некоторого класса, то надпись на этой кнопке должна отражать имя этого класса. Например, если кнопка связана с объектом класса Circle, то название этой кнопки должно быть «Окружность»

5. При нажатии на кнопку из первой группы ссылке t присваивается адрес объекта, с которым эта кнопка связана, т.е. данный объект считается активным. Кроме того, через одну из меток выводится на форму имя класса активного объекта или название нажатой кнопки.

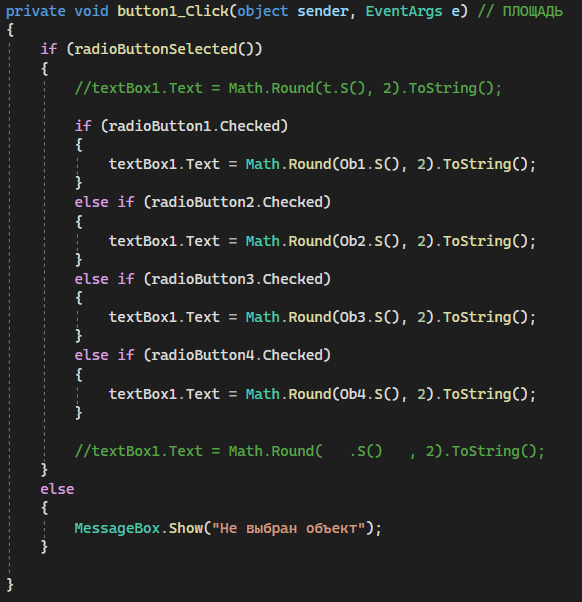
6. На вторую панель помещаются кнопки, которым в соответствие ставятся методы, определенные в классах Вашей иерархии. Для одноименных методов, описанных в разных классах ставиться в соответствие одна 34 кнопка. Поскольку в каждом классе определены по два метода (они должны иметь разные имена), вычисляющие площадь (одна – обычная, другая – виртуальная), то для вычисления площади помещаются две кнопки. Например, кнопки с названиями «Площадь\_О» и «Площадь\_В». Названия кнопок второй группы должны соответствовать именам методов поставленных в соответствие кнопкам.

7. При нажатии на любую кнопку из второй группы, кроме кнопки с именем «Площадь\_В», производится проверка того может ли активный объект приведен к типу, в котором определен метод, соответствующий нажатой кнопки, если да, то выводится значение этого метода, а если нет то сообщение о невозможности произвести вычисления. Например: if(t is Circle) label2.Text=((Circle)t).f().ToString(); else label2.Text=”Метод f() для активного объекта не определен”; Вызываемый метод, должн работать корректно, т.е. при вычислении площади окружности должна вычисляться площадь окружности, а не треугольника.

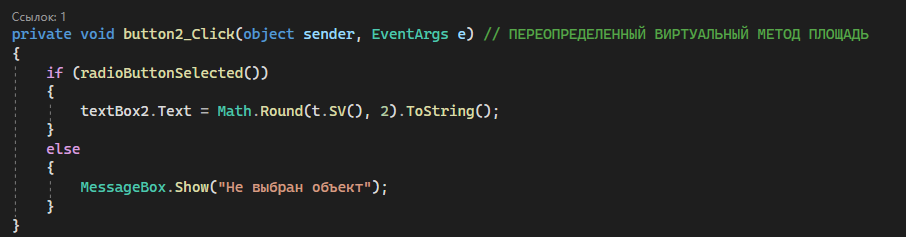
8. При нажатии на кнопку «Площадь\_В», вызывается виртуальный метод вычисления площади для активного объекта.

Анализ фрагментов программы

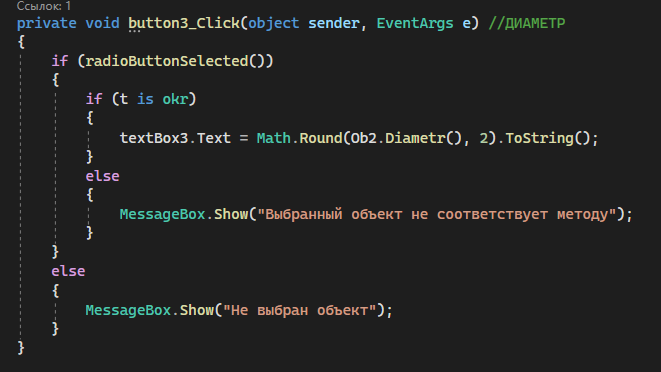
Для подсчитывания площади следует не брать ссылку на объект, а обращаться конкретно по нужному объекту и одноименному методу:



Касательно переопределенного виртуального метода можно использовать просто ссылку на объекты:



В программе для подсчета нужно учитывать принадлежность к определенному классу, например для кнопки подсчета диаметра проверяется принадлежность к классу окружности:



Распечатка программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace lab5

{

public partial class Form1 : Form

{

tochka Ob1 = new tochka();

okr Ob2 = new okr(10);

sektor Ob3 = new sektor(10, 100);

rectangle Ob4 = new rectangle(5, 5);

tochka t;

public Form1()

{

InitializeComponent();

t = Ob2;

}

// очистка полей после выбора объекта

private void Ochistka()

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

Ochistka();

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

Ochistka();

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

Ochistka();

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

Ochistka();

}

// -- конец очистки

private Boolean radioButtonSelected() // проверка выбранной кнопки

{

if (radioButton1.Checked)

{

t = Ob1;

}

else if (radioButton2.Checked)

{

t = Ob2;

}

else if (radioButton3.Checked)

{

t = Ob3;

}

else if (radioButton4.Checked)

{

t = Ob4;

}

else

{

return false;

}

return true;

}

private void button1\_Click(object sender, EventArgs e) // ПЛОЩАДЬ

{

if (radioButtonSelected())

{

//textBox1.Text = Math.Round(t.S(), 2).ToString();

if (radioButton1.Checked)

{

textBox1.Text = Math.Round(Ob1.S(), 2).ToString();

}

else if (radioButton2.Checked)

{

textBox1.Text = Math.Round(Ob2.S(), 2).ToString();

}

else if (radioButton3.Checked)

{

textBox1.Text = Math.Round(Ob3.S(), 2).ToString();

}

else if (radioButton4.Checked)

{

textBox1.Text = Math.Round(Ob4.S(), 2).ToString();

}

//textBox1.Text = Math.Round( .S() , 2).ToString();

}

else

{

MessageBox.Show("Не выбран объект");

}

}

private void button2\_Click(object sender, EventArgs e) // ПЕРЕОПРЕДЕЛЕННЫЙ ВИРТУАЛЬНЫЙ МЕТОД ПЛОЩАДЬ

{

if (radioButtonSelected())

{

textBox2.Text = Math.Round(t.SV(), 2).ToString();

}

else

{

MessageBox.Show("Не выбран объект");

}

}

private void button3\_Click(object sender, EventArgs e) //ДИАМЕТР

{

if (radioButtonSelected())

{

if (t is okr)

{

textBox3.Text = Math.Round(Ob2.Diametr(), 2).ToString();

}

else

{

MessageBox.Show("Выбранный объект не соответствует методу");

}

}

else

{

MessageBox.Show("Не выбран объект");

}

}

private void button4\_Click(object sender, EventArgs e) // ДЛИНА ДУГИ

{

if (radioButtonSelected())

{

if (t is sektor)

{

textBox4.Text = Math.Round(Ob3.Ldugi(), 2).ToString();

}

else if (t is rectangle) {

textBox4.Text = Math.Round(Ob4.Perimetr(), 2).ToString();

}

else

{

MessageBox.Show("Выбранный объект не соответствует методу");

}

}

else

{

MessageBox.Show("Не выбран объект");

}

}

private void button5\_Click(object sender, EventArgs e) // ДИАГОНАЛЬ

{

if (radioButtonSelected())

{

if (t is rectangle)

{

textBox5.Text = Math.Round(Ob4.Diag(), 2).ToString();

}

else

{

MessageBox.Show("Выбранный объект не соответствует методу");

}

}

else

{

MessageBox.Show("Не выбран объект");

}

}

}

class tochka // КЛАСС ТОЧКА

{

public double S()

{

return 0;

}

public virtual double SV()

{

return 0;

}

}

class okr : tochka

{

private double R;

public okr(double r)

{

R = r;

}

public new double S()

{

return Math.PI \* R \* R;

}

public override double SV()

{

return Math.PI \* R \* R;

}

public double Diametr()

{

return R \* 2;

}

}

class sektor : tochka // СЕКТОР

{

private double r;

private double ugol;

public sektor(double r, double ugol)

{

this.r = r;

this.ugol = ugol;

}

public double S()

{

return Math.PI \* r \* r \* (ugol / 360);

}

public override double SV()

{

return Math.PI \* r \* r \* (ugol / 360);

}

public double Ldugi() //длина дуги

{

return Math.PI \* r \* (ugol / 360);

}

}

class rectangle : tochka // прямоугольник

{

// класс прямоугольник дополнить методом вычисляющим периметр, обеспечить корректный вызов этого метода с помощью длина дуги

private double a;

private double b;

public rectangle(double a, double b)

{

this.a=a;

this.b=b;

}

public double S()

{

return a \* b;

}

public override double SV()

{

return a \* b;

}

public double Diag()

{

return Math.Sqrt(a \* a + b \* b);

}

public double Perimetr()

{

return (a + b) \* 2;

}

}

}

Результат работы программы

