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 № 6

по дисциплине «Технологии программирования»

Тема: «Абстрактные классы и интерфейсы»

Выполнил: Стычинский С.В., МВИ-121

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

Москва 2023

Цель работы

Самостоятельное определение абстрактного класса и интерфейса. Изучение особенностей наследования от абстрактного класса или интерфейса

Задание

1. Размер фигуры определяется размером прямоугольника, в который она помещается, назовем его базовым. Расположение фигуры в этом прямоугольнике, определяется вариантом задания. При выводе фигуры на экран базовый прямоугольник не рисуется.

2. Создать форму Dialog для ввода размеров базового прямоугольника. a. В класс формы Dialog добавить два открытых поля Х, У типа int, для хранения вводимой информации. b. На форму Dialog поместить два текстовых поля и одну кнопку. При нажатии на кнопку информация, записанная в поля ввода должна преобразовываться в тип int и записываться в поля Х и У.

3. Описать иерархию классов, определяемую вариантом задания. Каждый класс иерархии определить в отдельном файле.

4. В класс Form1 добавить поле Figure, которое является ссылкой на абстрактный класс для нечетных вариантов и ссылкой на интерфейс для четных вариантов.

5. На форму Form1 поместить три кнопки и два поля ввода. Первые две из кнопок должны иметь названия соответствующие именам производных классов, третья – надпись «Рисование».

6. При нажатии на одну из двух первых кнопок должны выполняться следующие действия: a. Создаваться объект класса, имя которого соответствует надписи на кнопке. Адрес созданного объекта должен записываться в ссылку типа Figure . b. Выводиться на экран форма Dialog. После ее закрытия, информация из ее полей Х и У должна быть переписана в поля созданного объекта.

7. При нажатии на кнопку «Рисование» должна выполняться проверка того, что ссылка Figure не является нулевой, и если - нет, то на экран должна выводиться фигура. Место положения фигуры должно определяться информацией, находящейся в полях ввода формы Form1.

Задание для 19 варианта: Первый вариант иерархии классов, 6, 7 варианты фигур.

Иерархия включает абстрактный базовый класс под названием «Фигура»,

и два его подкласса.Абстрактный класс содержит абстрактный метод

рисования фигуры и два поля или два свойства, определяющие размер

фигуры.В подклассах переопределяется абстрактный метод.

6. Равнобедренный треугольник, основание треугольника совпадает с

нижней базового прямоугольника, а противоположная основанию

вершина лежит на середине верхней стороны базового прямоугольника.

7. Прямоугольный треугольник, катеты которого совпадают со сторонами

базового прямоугольника

Теоретический материал

Абстрактные классы

Класс называется абстрактным, если он предназначен только для использования в качестве базового при наследовании и нельзя создавать его объекты. На языке С# для описания базового класса используется служебное слово abstract . Абстрактные классы могут содержать точно такие же члены, как и обычные классы. Кроме того, они могут содержать и абстрактные методы – методы, не имеющие реализации. Абстрактные методы описываются специальным образом. Во первых при их описании используется служебное слово abstract, а во вторых вместо описания тела метода ставится точка с запятой.

Пример.

abstract class M{

public int x;

public int g(){return x;}

abstract public int fun();

}

Абстрактный класс М, содержит поле х, не абстрактный метод g() и абстрактный метод fun(). Абстрактные методы не могут содержаться в неабстрактных классах. Все абстрактные методы должны быть определены в производных классах с использованием служебного слова override.

class N:M{

override public int fun(){return 3;}

}

Интерфейсы

Интерфейсы нужны для стандартизации интерфейсов объектов. Поскольку очень важно, чтобы методы, выполняющие сходные действия в разных классах, назывались одинаково и имели одинаковые формальные параметры. Интерфейс – это именованный набор абстрактных членов. Интерфейсы не могут содержать поля, но могут содержать абстрактные методы, свойства и события. В языке С# интерфейс – это специальный тип (interface), содержащий толь абстрактные члены. Поскольку других членов в интерфейсе быть не 44 может, то при их описании не используется слово abstract , поскольку интерфейс определяет только общедоступные члены, то при их описании не указывается уровень доступа. Пример. interface I { int Sq();} Если класс наследует интерфейс, то в нем обязательно должны быть определены все члены, входящие в интерфейс.

Class A:I {

public int Sq(){return 6;}

}

Замечание. Вместо фразы «класс А наследует интерфейс I» говорят «класс А реализует интерфейс I» или «класс А поддерживает интерфейс I». Класс может поддерживать несколько интерфейсов. Ссылки на интерфейс Ссылки на интерфейс - это ссылки на объекты классов, поддерживающих данный интерфейс. Если объявлена ссылка на интерфейс I, (I b;), а А – класс поддерживающий интерфейс I, то можно написать b=new A(); и работать с вновь созданным объектом класса А, через ссылку b. Если с объектом класса мы работаем через ссылку на интерфейс, то в этом случае будут доступны только члены, описанные в интерфейсе. Ссылки на интерфейс могут быть формальными параметрами методов и могут возвращаться методами. Свойства Поскольку в ООП поля классов рекомендуется создавать закрытыми, а работа с закрытыми членами класса осуществляется через открытые методы, то данная работа выполняется очень часто. Поэтому для ее облегчения создана специальная синтаксическая конструкция, которая называется свойство. 45 Свойство – это два метода, предназначенные для работы с закрытым полем, и оформленные специальным образом. В некоторых языках, таких как Объектный Паскаль, в определение свойства включают и само закрытое поле.

Пример описания свойства:

private int svv;

public int sv {

get{return svv;}

set{svv=value;}

}

Если в определении свойства отсутствует раздел get, то свойство предназначено только для записи, а если раздел set – то свойство предназначено только для чтения. Свойство имеет еще одно преимущество перед обычными методами, обеспечивающими доступ к закрытому члену. Они вызываются специальным образом, имитирующим обращение к полю. Например f.sv=5; int i=f.sv;

Распечатка программы

Класс Figure

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6

{

abstract class Figure //фигура

{

private int width;

private int height;

// СВОЙСТВО

public int Width

{

get { return width; }

set { width = value; }

}

public int Height

{

get { return height; }

set { height = value; }

}

//рисование

public abstract void Draw(Graphics g, int x, int y, int xf, int yf);

}

}

Класс RavnobedTriangle

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6

{

class RavnobedTriangle : Figure // равнобедренный

{

public override void Draw(Graphics g, int x, int y, int xf, int yf)

{

Pen pen = new Pen(Color.Black, 3);

Point[] points = { new Point(x, y), new Point(x + xf, y), new Point(x + xf / 2, y - yf) };

g.DrawPolygon(pen, points);

}

}

}

Класс RightTriangle

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6

{

class RightTriangle : Figure // прямоугольный треугольник

{

public override void Draw(Graphics g, int x, int y, int xf, int yf)

{

Pen pen = new Pen(Color.Black, 3);

Point[] points = { new Point(x, y), new Point(x + xf, y), new Point(x, y - yf) };

g.DrawPolygon(pen, points);

}

//Требование: добавить исключени и обработку исключений

// в этой лабораторной нужно защитить ввод данных обработкой исключительных ситуаций

}

}

Класс Form1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab6

{

public partial class Form1 : Form

{

Dialog d = new Dialog();

Figure ff; // ССЫЛКА

public Form1()

{

InitializeComponent();

d.FormClosed += new FormClosedEventHandler(d\_FormClosed); // отслеживание закрытия формы dialog

}

private void button1\_Click(object sender, EventArgs e) //равнобедренный треугольник - RavnobedTriangle

{

RavnobedTriangle F1 = new RavnobedTriangle();

ff = F1;

d.ShowDialog();

//отслеживание закрытия формы

}

private void button2\_Click(object sender, EventArgs e) //прямоугольный треугольник - RightTriangle

{

RightTriangle F2 = new RightTriangle();

ff = F2;

d.ShowDialog();

//отслеживание закрытия формы

}

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

{

Pryamougolnik F3 = new Pryamougolnik();

ff = F3;

d.ShowDialog();

}

private void d\_FormClosed(object sender, FormClosedEventArgs e)

{

ff.Height = d.X;

ff.Width = d.Y;

}

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

{

int x;

int y;

if (ff != null)

{

bool result1 = int.TryParse(textBox1.Text, out var \_);

bool result2 = int.TryParse(textBox2.Text, out var \_);

//обработка результата преобразований

if (!result1 || !result2)

{

// если хотя бы одно из чисел не int - выводим ошибку и очищаем неверное поле/неверные поля

MessageBox.Show("Введите целые числа!");

if (!result1) { textBox1.Text = ""; }

if (!result2) { textBox2.Text = ""; }

}

else

{

x = int.Parse(textBox1.Text);

y = int.Parse(textBox2.Text);

Graphics g = CreateGraphics();

ff.Draw(g, x, y, ff.Width, ff.Height);

}

}

}

}

}

Класс Dialog

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab6

{

public partial class Dialog : Form

{

public int X;

public int Y;

public Dialog()

{

InitializeComponent();

}

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

{

//попытки преобразования в int

bool result1 = int.TryParse(textBox1.Text, out var \_);

bool result2 = int.TryParse(textBox2.Text, out var \_);

//обработка результата преобразований

if (!result1 || !result2)

{

// если хотя бы одно из чисел не int - выводим ошибку и очищаем неверное поле/неверные поля

MessageBox.Show("Введите целые числа!");

if (!result1) { textBox1.Text = ""; }

if (!result2) { textBox2.Text = ""; }

}

else

{

X = int.Parse(textBox1.Text);

Y = int.Parse(textBox2.Text);

//MessageBox.Show($"X={X};\nY={Y}");

this.Close(); //закрытие формы

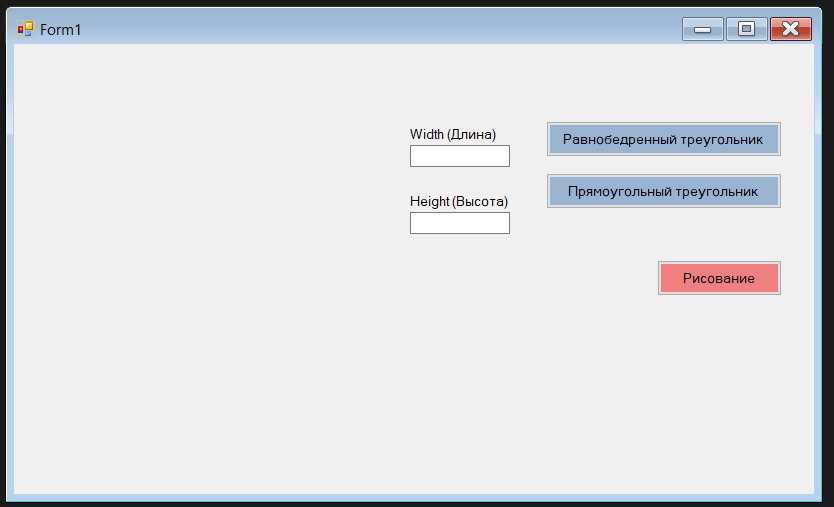
}

}

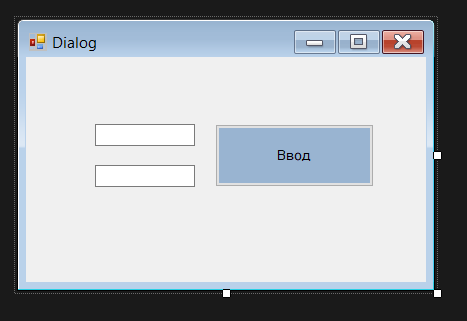
}

}

Конструктор формы Form1



Конструктор формы Dialog



Результат работы программы

