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 № 3

по дисциплине «Технологии программирования»

Тема: «Работа с закрытыми членами классов»

Выполнил: Стычинский С.В. ,МВИ-121

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

Москва 2023

Цель работы

Научится создавать конструкторы классов и контролировать ввод данных в поля класса.

Задание

Ограничение на одно из полей следующее: его значение должно быть больше 4 и меньше 16. Форма должна содержать 5 полей, являющихся объектами Вашего класса. (4 задание)

Справочный материал

Конструктор класса – это метод класса, который предназначен для создания объектов класса и инициализации их. Конструктор обладает следующими свойствами:

1. Имя конструктора совпадает с именем класса.

2. Конструктор не имеет возвращаемого значения, даже типа void

3. Класс может содержать несколько конструкторов, отличающихся либо количеством, либо типами параметров (отличие в этом случае должно быть существенным), либо и тем и другим.

4. Конструктор без параметров называется конструктором по умолчанию.

5. Если ни один конструктор не был написан программистом, то конструктор по умолчанию генерируется компилятором.

6. Если в классе объявлен хотя бы один конструктор, то компилятор не создает своего.

7. В C# , в отличие от многих других языков, любой конструктор, если не указано иное, обнуляет значения полей.

8. Конструктор не может быть вызван явно, он вызывается автоматически при создании экземпляров класса.

Рассмотрим пример. Определим в классе А два конструктора.

class A {

public int x,y;

public int f() {

return x+y;

}

public A() {

x=1; y=2; }

public A (int xx, int yy) {

x=xx; y=yy;}

}

При выполнении команды A a=new A(); для создания объекта класса используется конструктор без параметров. Полям объекта а будут присвоены, соответственно, значения 1 и 2. Если при создании объекта требуется использовать конструктор с параметрами, то в операторе new после имени класса в круглых скобках указываются значения параметров, например, A b=new A(6,13); А команда A c=new A(3); вызовет ошибку, потому что класс А не содержит конструктора с одним параметром, которому можно было бы присвоить значение 3. В языке С# существует упрощенная форма написания конструктора: при описании полей класса можно написать инициализирующие выражения.

Пример.

class В {

public int i=10;

}

Все строки, содержащие инициализирующие выражения, включаются в начало всех конструкторов. В данном примере строка i=10; будет включена в конструктор, определяемый компилятором. Уровни доступа к членам класса В С# имеется пять спецификаторов доступа: public, private, protected, internal, protected internal . Значения спецификаторов доступа следующие: - public - член класса может быть использован любым методом; - private - член класса может быть использован только методом того же класса, - - член класса может быть использован только либо методом того же класса, либо методом производного класса; - internal- член класса может быть использован только методом из той же сборки; 18 - protected internal- член класса может быть использован только либо методом из той же сборки или методом производного класса из любой сборки. Работа с закрытыми полями класса При выполнении лабораторной работы №2 были созданы классы, которые нужны при работе с геометрическими фигурами. Размеры геометрических фигур должны задаваться неотрицательными числами. Это требование не учитывалось при выполнении лабораторной работы №2. В поля классов можно вводить любые, в том числе отрицательные значения. Этот недостаток можно исправить, если изменить уровень доступа к полям классов на private или protected, т.е. сделать их закрытыми от обращений извне. Но в этом случае нужно будет написать по два открытых метода, для каждого поля, для того, что бы вводить и выводить информацию.

Например,

class A{

private int x;

public int get\_X(){

return x;

}

public bool set\_X(int xx){

if(xx < 0)

return true;

x=xx;

return false;

}

}

В поле х класса А уже нельзя ввести отрицательного значения, метод set\_X контролирует ввод, причем он возвращает информацию о том была ли допущена ошибка. При отрицательных значениях параметра хх, правильнее было бы сгенерировать исключение, что бы обратить внимание на попытку недопустимого ввода, но генерировать и обрабатывать исключения мы научимся позднее. Рассмотрим еще один пример. Предположим, что создается класс, в котором, в одном из полей хранится информация о температуре тела человека. Обозначим это поле буквой t, ясно, что значение поля t должно быть больше 35 и меньше 42. Попытка ввести в это поле значение не из указанного 19 интервала, свидетельствует об ошибке. Что бы сохранить информацию о наличии ошибки создадим поле булевого типа err.

class В{

private float t;

private bool err =false;

public float get\_T(){

return t;

}

public void set\_T(float tt){

if(tt < 35 || tt>42 ) {

MessageBox.Show("Ошибка, температура тела человека не может быть равной "+tt.ToString() ,"Внимание!");

err= true;

}

else {

t=tt ;

err=false;

}

public bool get\_Err(){

return err;

}

}

В последнем примере, не только сразу сообщается об ошибке с помощью MessageBox, но и информация о ней сохраняется в поле err. Значение поля err нельзя изменить напрямую, оно меняется только при изменении поля t. Переключатели Компоненты CheckBox и RadioButton используются, для того чтобы пользователь мог включать и выключать какие-то режимы работы приложения или индикации состояния. CheckBox называют флажком. Это прямоугольник с надписью, содержание которой определяется свойством Text. В прямоугольнике может находиться галочка, в этом случае, говорят “флажок поднят”, при отсутствии галочки говорят «флажок снят». Щелчок мышки поднимает или снимает флажок. Поднятие и снятие флажка обуславливается свойством Checked. Это свойство имеет булев тип. Значение true – соответствует поднятому флажку, false – опущенному флажку. Компонент RadioButton в единственном числе очень похож на компонент CheckBox. Отличие заключается в том, что вместо прямоугольника на экран выводится кружочек, а вместо галочки – точка. Появление или исчезновение 20 точки тоже связно со свойством Checked. Но компоненты RadioButton могут объединяться в функциональные группы. В группу попадают все компоненты RadioButton, расположенные в одном контейнере (контейнером называется компонент, на котором могут располагаться другие компоненты, например, форма или панели). В группе может быть выбран только один компонент RadioButton. Компоненты CheckBox, расположенные в одном контейнере, в группу не объединяются, они работают независимо друг от друга.

Этапы выполнения лабораторной работы

1. Поля, определенного в лабораторной работе №2 класса, сделать закрытыми, т.е. задать им соответствующий уровень доступа.

2. Значения всех полей должны быть неотрицательными. Для одного поля определить границы, согласно варианту индивидуального задания. Для каждого поля написать по два открытых метода, обеспечивающих запись и считывание информации в это поле, с учетом всех ограничений.

3. Откорректировать обработчики событий, программы лабораторной работы №2, так, что бы они работали с объектами, измененного класса.

4. Дополнить класс двумя конструкторами, один из которых должен быть без параметров, другой с параметрами.

5. Создать еще несколько долгоживущих объектов Вашего класса, число объектов определяется индивидуальным заданием. Хотя бы один их них должен быть создан конструктором с параметрами.

6. Создать поле формы, которое является ссылкой на объекты Вашего класса, назовем ее буквой А (в Ваших программах она должна иметь другое имя).

7. На форму поместить столько радио-кнопок (компонентов RadioButton) сколько определено долгоживущих объектов. Названия радио-кнопок должны отражать связь с объектами, они могут называться так же, как и объекты.

8. В программе несколько объектов Вашего класса, но в каждый момент времени пользователь работает только с одним. Выбор объекта 21 производится с помощью радио-кнопок. Выбранный объект называется активным. Можно сделать так, что бы ссылка А указывала на активный объект (это требование не является обязательным).

9. Откорректировать обработчики событий Click кнопок, выполняющих ввод и вывод информации в долгоживущий объект так, что бы они работали с активным объектом.

10. Если ни одна радио-кнопка не выбрана, а пользователь нажимает кнопки ввода или вывода, то программа должна сообщать об этом. Для вывода сообщений можно использовать MessageBox . Например. MessageBox.Show("Объект не выбран","Внимание!");

11. Отладить программу.

Распечатка программы

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 lab2

{

public partial class Form1 : Form

{

okr o1 = new okr(10); // поля главной формы

okr o2 = new okr(5);

okr o3 = new okr(5);

okr o4 = new okr();

okr o5 = new okr();

okr A;

public Form1()

{

InitializeComponent();

}

private Boolean radioButtonChoise()

{

if (radioButton1.Checked)

{

A = o1;

}

else if (radioButton2.Checked)

{

A = o2;

}

else if (radioButton3.Checked)

{

A = o3;

}

else if (radioButton4.Checked)

{

A = o4;

}

else if (radioButton5.Checked)

{

A = o5;

} else

{

return false;

}

return true;

}

//ввод

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "") { MessageBox.Show("Не введено число"); return; }

double r = (double)int.Parse(textBox1.Text);

if (!radioButtonChoise()) {

MessageBox.Show("Объект не выбран", "Внимание!");

return;

}

A.setR(r);

}

//вывод

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

{

if (!radioButtonChoise())

{

return;

}

textBox2.Text = A.DlinaOkr().ToString();

textBox3.Text = A.Sokr().ToString();

textBox1.Text = A.getR().ToString();

}

// кнопка расчет

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

{

double r = (double)int.Parse(textBox1.Text);

okr b = new okr(r);

button3.Text = "Длина окружности:\n" + b.DlinaOkr().ToString() + "\nПлощадь окружности:\n" + b.Sokr().ToString();

}

}

//4) Класс, описывающий окружность. Класс должен иметь методы, решающие следующие задачи:

// a.расчет длины окружности;

// b.расчет площади окружности.

class okr

{

private double R;

public okr() {

Random rnd = new Random();

int value = rnd.Next(5, 15);

this.R = value; // значение по умолчанию

}

public okr(double r)

{

this.R = r;

}

public double DlinaOkr()

{

return 2\*Math.PI \* this.R;

}

public double Sokr()

{

return Math.PI \* this.R \* this.R;

}

public bool setR(double r)

{

if (r>4 && r<16) {

this.R = r;

return true;

}

MessageBox.Show("Ошибка");

return false;

}

public double getR()

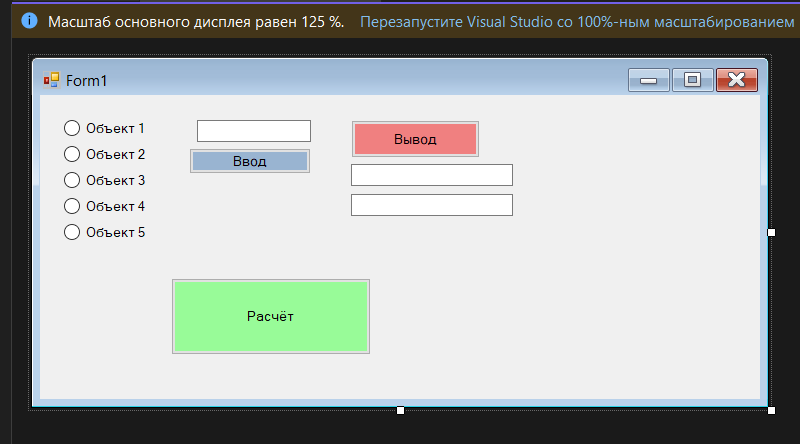
{

return this.R;

}

}

}



Результат работы программы

