**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Радиотехнический»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-11 |  | преподаватель каф.ИУ5 |
| Стадник Елена |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2020 г.

**Описание задания**

**Часть 1. Разработать программу, использующую делегаты.**

(В качестве примера можно использовать проект «Delegates»).

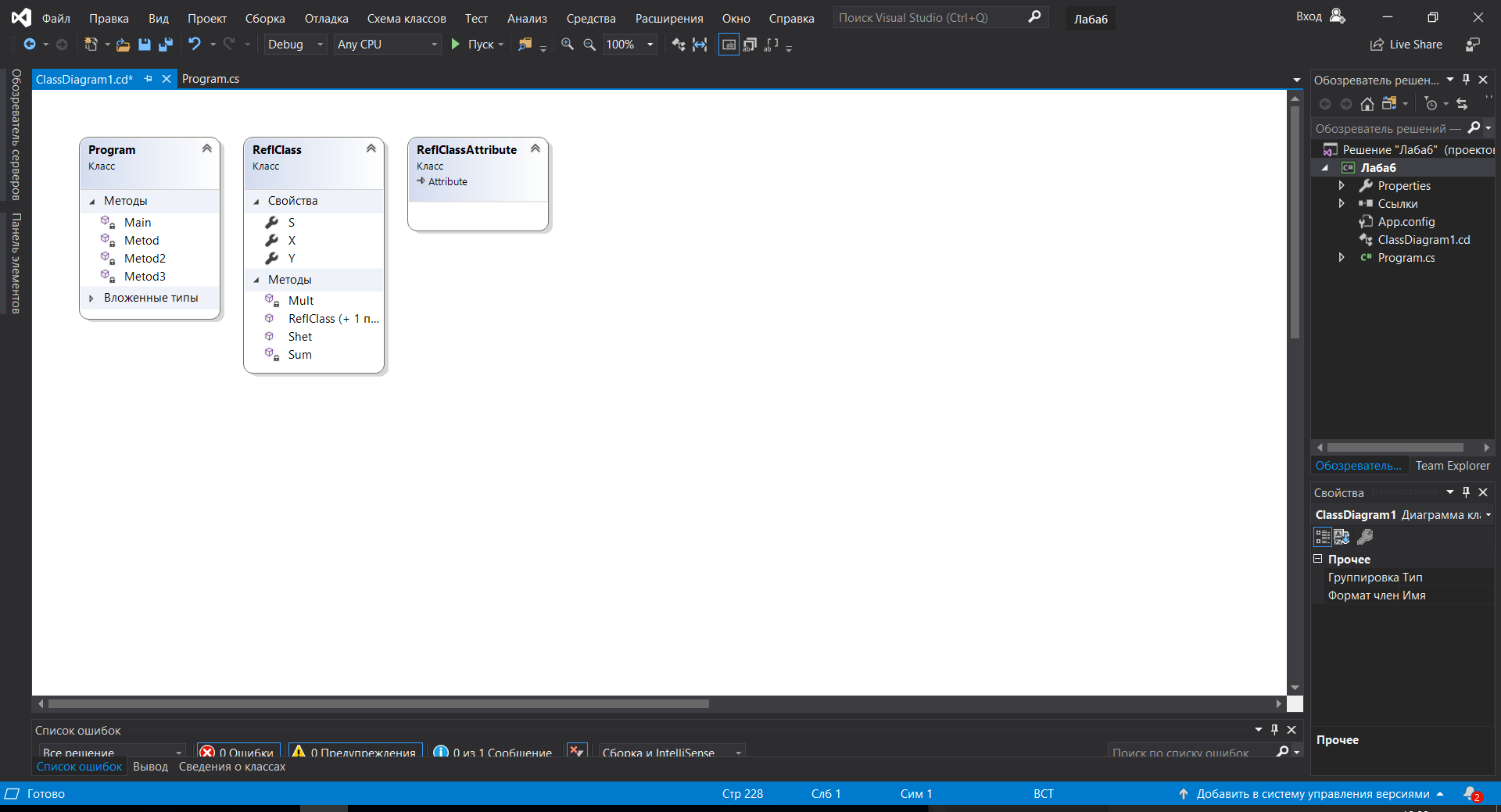
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат, в качестве одного из входным параметров. Осуществите вызов метода, передавая в качестве параметра-делегата:
   * метод, разработанный в пункте 3;
   * лямбда-выражение.
5. Повторите пункт 4, используя вместо разработанного Вами делегата, обобщенный делегат Func< > или Action< >, соответствующий сигнатуре разработанного Вами делегата.

**Часть 2. Разработать программу, реализующую работу с рефлексией.**

(В качестве примера можно использовать проект «Reflection»).

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс, содержащий конструкторы, свойства, методы.
3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
4. Создайте класс атрибута (унаследован от класса System.Attribute).
5. Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.
6. Вызовите один из методов класса с использованием рефлексии.

**Диаграмма классов**



**Текст программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace Лаба6

{

class Program

{

delegate int MetodDel(int x, string str, bool b);

static void Main(string[] args)

{

MetodDel del = Metod;//экз делегата

Console.WriteLine($"метод1, соответствующий созданному делегату:\t\t\t\t\t{del(12, "zazaz", true)}");//выводим metod

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - метод1):\t\t{Metod2(del, 7)}");//выводим metod2 передавая ему делегат

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - лямда-выражение:\t{Metod2((x, str, b) => x + str.Length, 7)}");//выводим metod2 передавая ему лямда-выражение

Func<int, string, bool, int> dFub = Metod;

Console.WriteLine($"Обобщенный метод, принимающий разработанный делегат(параметр-делегата - метод1:\t{Metod3(dFub, 5)}");//выводим metod3 передавая ему об. делегат

Console.WriteLine($"метод, принимающий разработанный делегат(параметр-делегата - лямда-выражение:\t{Metod3((x, str, b) => x + str.Length, 5)}");//выводим лямда-выражение

Console.ReadKey();

Type myType = typeof(ReflClass);//получаем тип

#region Информация о конструкторах

Console.WriteLine("\nИнформация о конструкторах");

ConstructorInfo[] cInfo = myType.GetConstructors();//заносим инфу в ConstructorInfo

foreach (ConstructorInfo o in cInfo)

Console.WriteLine(o);//выводим

#endregion

#region Информация о методах

Console.WriteLine("\nИнформация о методах");

MethodInfo[] cMeth = myType.GetMethods(BindingFlags.Instance | BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.DeclaredOnly);//заносим инфу в MethodInfo, используя флаги, нам подходят: (Instance-Указывает, что члены экземпляров следует включить в поиск) (методы public), (NonPublic(частные)) и (только те, что находятся нашем классе)

foreach (MethodInfo o in cMeth)

Console.WriteLine(o);//выводим

#endregion

#region Информация о свойствах

Console.WriteLine("\nИнформация о свойствах");

PropertyInfo[] cProp = myType.GetProperties();//заносим инфу в PropertyInfo

foreach (PropertyInfo o in cProp)

Console.WriteLine(o);//выводим

#endregion

#region Информация только о свойствах с атрибутами

Console.WriteLine("\nИнформация только о свойствах с атрибутами");

foreach (PropertyInfo o in cProp)

{

object attributes = o.GetCustomAttribute(typeof(ReflClassAttribute), false);//извлекаем настраив. атрибут(который указали)

if (attributes != null)//если ничего не достали, значит его нет, а значит его не выводим

Console.WriteLine(o);//выводим

//object[] attributes = o.GetCustomAttributes(typeof(ReflClassAttribute), false);//как вариант

//if(attributes.Length>0)

// Console.WriteLine(o);

}

#endregion

#region Вызов метода

dynamic st = Activator.CreateInstance(myType);//создаем экз класса с помощью рефлексии

Console.WriteLine("\n" + st.Shet(true));//вызываем нужный метод

#endregion

}

static int Metod(int x, string str, bool b)//первый метод

{

int n = 0;

foreach (char o in str)

if (o == 'z')

n++;

int M = b ? n + x + str.Length : n + x;

return M;

}

static int Metod2(MetodDel f, int x)//второй метод принимающий в качестве параметра делегат

{

return x + f(x, "zez", false); ;

}

static int Metod3(Func<int, string, bool, int> F, int x)//третий метод принимающий в качестве параметра обобщенный делегат

{

return x + F(x, "zez", false); ;

}

}

class ReflClass

{

#region Конструкторы

public ReflClass()

{

X = 5;

Y = 6;

}

public ReflClass(int a, int b)

{

X = a;

Y = b;

}

#endregion

#region Свойства

[ReflClass]

public int X { get; set; }

public int S { get; set; }

[ReflClass]

public int Y { get; set; }

#endregion

#region Методы

public int Shet(bool f)

{

int M = f ? Sum() : Mult();

S = M;

return S;

}

private int Sum()

{

return X + Y;

}

private int Mult()

{

return X \* Y;

}

#endregion

}

class ReflClassAttribute : System.Attribute//атрибут

{

}

}