МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Радиотехнический факультет (РТ)

Лабораторная работа № 6

По дисциплине: «Базовые компоненты интернет технологий»

Тема: «Разработать программу, использующую делегаты. Разработать программу, реализующую работу с рефлексией.»

Выполнил: Наврузов Э.Р.

студент группы РТ5-31

Проверил: Гапанюк Ю.Е.,

Преподаватель каф. ИУ5

г. Москва 2017 г.

Описание задания лабораторной работы:

Часть 1. Разработать программу, использующую делегаты

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 входных 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

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

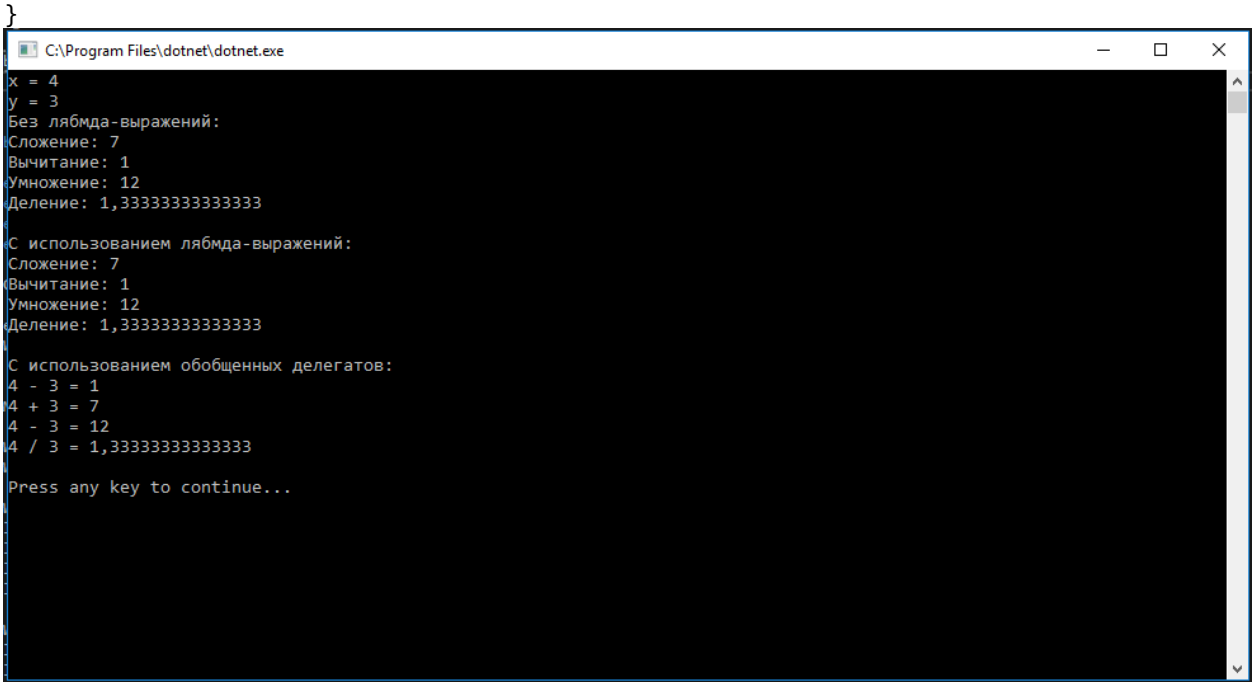
6. Вызовите один из методов класса с использованием рефлексии.

Текст программы:

Делегаты:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DelegateLab6
{
    class Program
    {
        delegate double CalcDeleg(double a, double b);

        static double Sum(double a, double b) { return a + b; }
        static double Dif(double a, double b) { return a - b; }
        static double Mul(double a, double b) { return a * b; }
        static double Div(double a, double b) { return a / b; }

        static void CalcResult(string str, double arg1, double arg2, CalcDeleg
DelParametr)
        {
            double result = DelParametr(arg1, arg2);
            Console.WriteLine(str + result.ToString());
        }

        static void Main(string[] args)
        {
            Console.Write("x = "); double x = Convert.ToDouble(Console.ReadLine());
            Console.Write("y = "); double y = Convert.ToDouble(Console.ReadLine());

            Console.WriteLine("Без лябмда-выражений:");
            CalcResult("Сложение: ", x, y, Sum);
            CalcResult("Вычитание: ", x, y, Dif);
            CalcResult("Умножение: ", x, y, Mul);
            CalcResult("Деление: ", x, y, Div);

            Console.WriteLine("\nС использованием лябмда-выражений:");
            CalcResult("Сложение: ", x, y, (arg1, arg2) => arg1 + arg2);
            CalcResult("Вычитание: ", x, y, (arg1, arg2) => arg1 - arg2);
            CalcResult("Умножение: ", x, y, (arg1, arg2) => arg1 * arg2);
            CalcResult("Деление: ", x, y, (arg1, arg2) => arg1 / arg2);

            Console.WriteLine("\nС использованием обобщенных делегатов:");
            Action<double, double> Sumaction = (arg1, arg2) =>
            { Console.WriteLine("{0} + {1} = {2}", arg1, arg2, arg1 + arg2); };

            Action<double, double> Difaction = (arg1, arg2) =>
            { Console.WriteLine("{0} - {1} = {2}", arg1, arg2, arg1 - arg2); };

            Action<double, double> Mulaction = (arg1, arg2) =>
            { Console.WriteLine("{0} - {1} = {2}", arg1, arg2, arg1 * arg2); };

            Action<double, double> Divaction = (arg1, arg2) =>
            { Console.WriteLine("{0} / {1} = {2}", arg1, arg2, arg1 / arg2); };

            Action<double, double> AllDelegates = Difaction + Sumaction + Mulaction +
Divaction;
            AllDelegates(x, y);
```

```
            Console.Write("\nPress any key to continue...");
            Console.ReadKey();
        }
    }
}
```



Рефлексия:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;

namespace Reflect
{
    class Program
    {
        class MyTestClass
        {
            MyTestClass() { }
            double d, f;

            public MyTestClass(double d, double f)
            {
                this.d = d;
                this.f = f;
            }

            [NewAttribute("Первый параметр")]
            public double D { get; set; }
            [NewAttribute("Второй параметр")]
            public double F { get; set; }

            public double Sum() { return d + f; }
            public double Mul() { return d * f; }
        }

        [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited =
false)]
        public class NewAttribute : Attribute
```

```
        {
            public NewAttribute() { }
            public NewAttribute(string argument1)
            {
                Description = argument1;
            }

            public string Description { get; set; }
        }

        static void AboutClass()
        {
            Type t = typeof(MyTestClass);

            Assembly info = Assembly.GetExecutingAssembly();
            Console.WriteLine("Текущее имя сборки: " + info.FullName);
            Console.WriteLine("Исполняемый файл: " + info.Location);

            Console.WriteLine("Пространство имен: " + t.Namespace);
            Console.WriteLine("Тип: " + t.FullName);

            Console.WriteLine("Методы:");
            foreach (var x in t.GetMethods()) { Console.WriteLine(x); }
            Console.WriteLine("Конструкторы:");
            foreach (var x in t.GetConstructors()) { Console.WriteLine(x); }
            Console.WriteLine("Методы:");
            foreach (var x in t.GetProperties()) { Console.WriteLine(x); }
        }

        public static bool GetPropertyAttribute(PropertyInfo checkType, Type
attributeType, out object attribute)
        {
            bool Result = false;
            attribute = null;

            var isAttribute = checkType.GetCustomAttributes(attributeType, false);
            if (isAttribute.Length > 0)
            {
                Result = true;
                attribute = isAttribute[0];
            }
            return Result;
        }

        static void AboutAttribute()
        {
            Type t = typeof(MyTestClass);
            Console.WriteLine("\nСвойства, помеченные атрибутом:");

            foreach (var x in t.GetProperties())
            {
                object attrObj;

                if (GetPropertyAttribute(x, typeof(NewAttribute), out attrObj))
                {
                    NewAttribute attr = attrObj as NewAttribute;
                    Console.WriteLine(x.Name + " - " + attr.Description);
                }
            }
        }

        static void Main()
        {
            AboutClass();
            AboutAttribute();
```

```
            Type t = typeof(MyTestClass);

            Console.WriteLine("\nВызов метода:");
            MyTestClass tc = (MyTestClass)t.InvokeMember(null,
BindingFlags.CreateInstance, null, null, new object[] { });
            object[] parameters = new object[] { };
            object Result = t.InvokeMember("AboutAttribute", BindingFlags.InvokeMethod,
null, tc, parameters);

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
    }
}
```

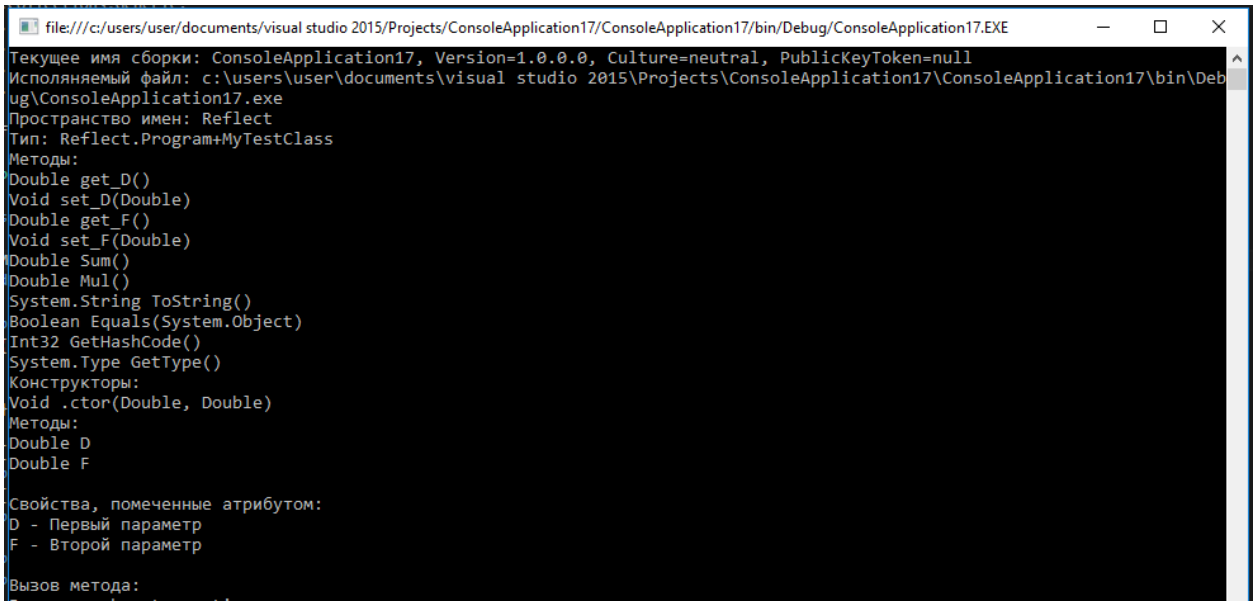
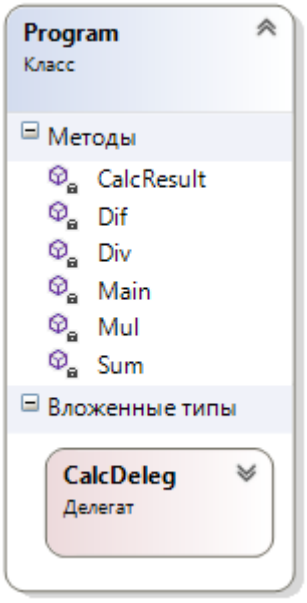
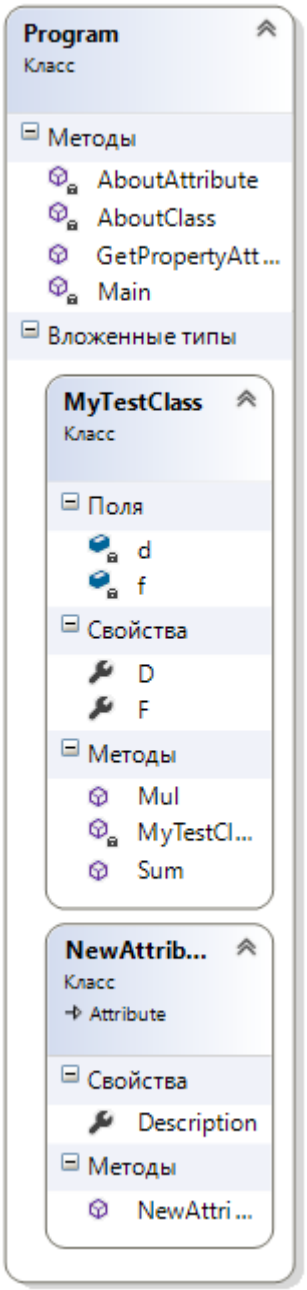


Диаграмма классов:

Делегаты:



Рефлексия:


Program
Класс
Методы
AboutAttribute
AboutClass
GetPropertyAtt...
Main
Вложенные типы
MyTestClass
Класс
Поля
d
f
Свойства
D
F
Методы
Mul
MyTestCl...
Sum
NewAttrib...
Класс
Attribute
Свойства
Description
Методы
NewAttri...