

**Московский государственный технический  
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

«Разработать программу, использующую делегаты и реализующую работу с рефлексией»

Выполнил:

студент группы ИУ5-31Б  
Алехин Сергей

Подпись и дата:

Проверил:

преподаватель каф. ИУ5  
Осликов С.П.

Подпись и дата:

г. Москва, 2019 г.

## Задание лабораторной работы

### 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.

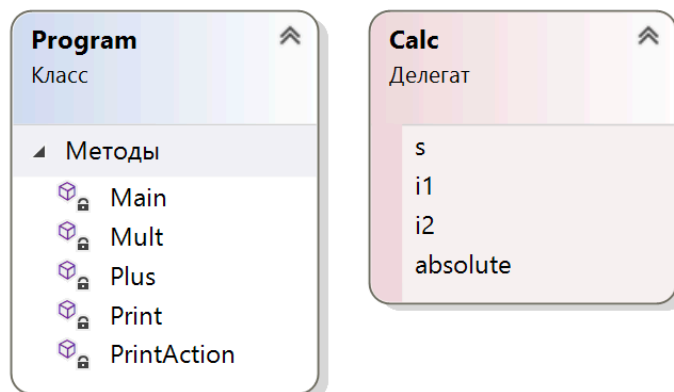
### Часть 2. Разработать программу, реализующую работу с рефлексией.

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

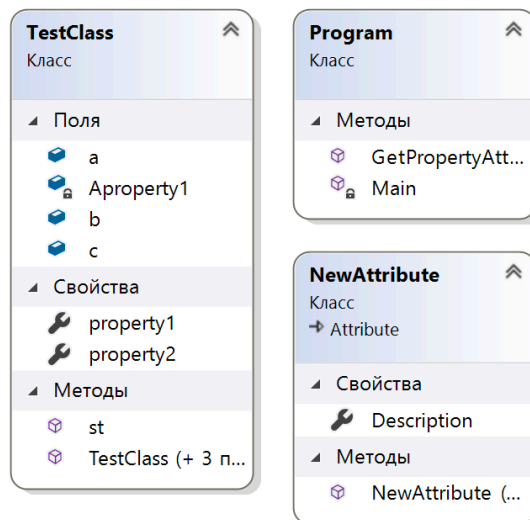
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс, содержащий конструкторы, свойства, методы.
3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
4. Создайте класс атрибута (унаследован от класса `System.Attribute`).
5.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

## Диаграмма классов

### 1 часть:



### 2 часть:



# Текст программы

## 1 часть:

### *Program.cs*

```
using System;
namespace Lab6_part1
{
    delegate int Calc(string s, int i1, int i2, bool absolute);
    class Program
    {
        static int Mult(string str, int a, int b, bool absolute)
        {
            Console.Write(str);
            if (absolute) return Math.Abs(a * b);
            else return a * b;
        }
        static int Plus(string str, int a, int b, bool absolute)
        {
            Console.Write(str);
            if (absolute) return Math.Abs(a + b);
            else return a + b;
        }
        static void Print(int a, int b, bool ab, string str, Calc func)
        {
            Console.WriteLine("Параметры: a = " + a.ToString() + ", b = " + b.ToString());
            Console.WriteLine("\nРезультат:" + (func(str, a, b, ab)).ToString());
        }
        static void PrintAction(int a, int b, bool ab, string str, Action<string, int, int, bool> act_param)
        {
            Console.WriteLine("Параметры: a = " + a.ToString() + ", b = " + b.ToString());
            act_param(str, a, b, true);
        }
        static void Main(string[] args)
        {
            Console.Title = "Алехин Сергей ИУ5-31Б";
            int a = 5;
            int b = -2;
            Print(a, b, true, "Умножение", Mult);
            Print(a, b, true, "Сложение", Plus);
            Print(a, b, false, "Вычитание",
                (str, x, y, absolute) =>
                {
                    Console.Write(str);
                    if (absolute) return Math.Abs(x - y);
                    else return x - y;
                }
            );
            Action<string, int, int, bool> act1 = (str, x, y, ab) =>
            {
                Console.Write(str);
                if (ab) Console.WriteLine(Math.Abs(x * y));
                else Console.WriteLine(x * y);
            };
            Action<string, int, int, bool> act2 = (str, x, y, ab) =>
            {
                Console.Write(str);
                if (ab) Console.WriteLine(Math.Abs(x + y));
                else Console.WriteLine(x + y);
            };
            PrintAction(a, b, true, "Умножение ", act1);
            PrintAction(a, b, true, "Сложение ", act2);
            Action<string, int, int, bool> act12 = act2 + act1;
```

```

        PrintAction(a, b, true, "Последовательные действия ", act12);
        Console.ReadKey();
    }
}
}

```

## 2 часть:

### *Attribute.cs*

```

using System;
namespace Lab6_part2
{
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]
    public class NewAttribute : Attribute
    {
        public string Description { get; set; }
        public NewAttribute() { }
        public NewAttribute(string y)
        {
            Description = y;
        }
    }
}

```

### *TestClass.cs*

```

namespace Lab6_part2
{
    class TestClass
    {
        public int a;
        public string b;
        public bool c;
        public TestClass() { }
        public TestClass(int x, string y, bool z) { a = x; b = y; c = z; }
        public TestClass(int x) { a = x; }
        public TestClass(int x, bool z) { a = x; c = z; }
        public int st (int x, bool z)
        {
            if (z)
            {
                return x * x;
            }
            else
            {
                return x * x * x;
            }
        }
        int Aproperty1;
        [NewAttribute("Описание для property1")]
        public int property1
        {
            get { return Aproperty1; }
            set { Aproperty1 = value; }
        }
        [NewAttribute("Описание для property2")]
        public string property2 { get; private set; }
    }
}

```

### *Program.cs*

```

using System;
using System.Reflection;
namespace Lab6_part2

```

```


{
class Program
{
    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 Main(string[] args)
{
    Console.Title = "Алехин Сергей ИУ5-31Б";
    Type t = typeof(TestClass);
    Console.WriteLine("Тип " + t.FullName + " унаследован от " + t.BaseType.FullName);
    Console.WriteLine("Пространство имен " + t.Namespace);
    Console.WriteLine("Находится в сборке " + t.AssemblyQualifiedName);
    Console.WriteLine("\nКонструкторы:");
    foreach (var x in t.GetConstructors())
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("\nМетоды:");
    foreach (var x in t.GetMethods())
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("\nСвойства:");
    foreach (var x in t.GetProperties())
    {
        Console.WriteLine(x);
    }
    Console.WriteLine("\nПоля данных (public):");
    foreach (var x in t.GetFields())
    {
        Console.WriteLine(x);
    }
    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);
            }
        }
    }
    Console.WriteLine("\nВызов метода:");
    TestClass fi = (TestClass)t.InvokeMember(null, BindingFlags.CreateInstance, null, null, new object[] { });
    object[] parameters = new object[] { 3, false };
    object Result = t.InvokeMember("st", BindingFlags.InvokeMethod, null, fi, parameters);
    Console.WriteLine("st(3,false)={0}", Result);
    Console.ReadKey();
}
}
}

```


## Примеры работы программы

### 1 часть:

 Алехин Сергей ИУ5-31Б

```
Параметры: a = 5, b = -2
Умножение
Результат:10
Параметры: a = 5, b = -2
Сложение
Результат:3
Параметры: a = 5, b = -2
Вычитание
Результат:7
Параметры: a = 5, b = -2
Умножение 10
Параметры: a = 5, b = -2
Сложение 3
Параметры: a = 5, b = -2
Последовательные действия 3
Последовательные действия 10
```

### 2 часть:

 Выбрать Алехин Сергей ИУ5-31Б

```
Тип Lab6_part2.TestClass унаследован от System.Object
Пространство имен Lab6_part2
Находится в сборке Lab6_part2.TestClass, Lab6_part2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

Конструкторы:
Void .ctor()
Void .ctor(Int32, System.String, Boolean)
Void .ctor(Int32)
Void .ctor(Int32, Boolean)

Методы:
Int32 st(Int32, Boolean)
Int32 get_property1()
Void set_property1(Int32)
System.String get_property2()
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
System.String ToString()

Свойства:
Int32 property1
System.String property2

Поля данных (public):
Int32 a
System.String b
Boolean c

Свойства, помеченные атрибутом:
property1 - Описание для property1
property2 - Описание для property2

Вызов метода:
st(3,false)=27
```

### Ссылка на репозиторий GitHub

<https://github.com/SergeyBMSTU2018/AlekhinSergeyLabs.git>