**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты Интернет-технологий»

Отчет по лабораторной работе №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Брусникина Мария |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2018 г.

Описание задания

**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 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.

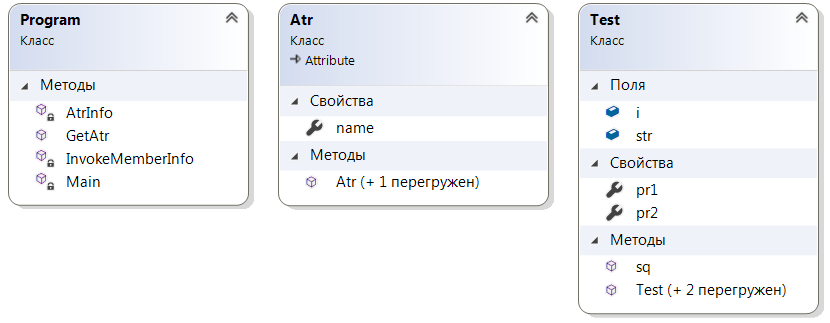
**Часть 2. Разработать программу, реализующую работу с рефлексией.**

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.

Вызовите один из методов класса с использованием рефлексии.

Диаграмма классов

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



Листинг

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

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

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 Print2(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)

{

Print(5, -10, true, "Умножение", Mult);

Print(-10, 5, 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);

};

Action<string, int, int, bool> chain = act1 + act2;

chain("Цепочка ", 3, -8, true);

Print2(5, -10, true, "Умножение ", act1);

Console.ReadKey();

}

}

}

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

using System;

using System.Reflection;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6\_v2

{

class Program

{

public static bool GetAtr(PropertyInfo check\_type, Type atr\_type, out object atr)

{

bool result = false;

atr = null;

var is\_atr = check\_type.GetCustomAttributes(atr\_type, false);

if (is\_atr.Length > 0)

{

result = true;

atr = is\_atr[0];

}

return result;

}

static void InvokeMemberInfo()

{

Type t = typeof(Test);

Console.WriteLine("\nВызов метода через рефлексию:");

//Создание объекта через рефлексию

Test test = (Test)t.InvokeMember(null, BindingFlags.CreateInstance, null, null, new object[] { });

object[] parameter = new object[] { 5 };

//Вызов метода

object result = t.InvokeMember("sq", BindingFlags.InvokeMethod, null, test, parameter);

Console.WriteLine("5 в степени 2 равно {0}", result);

}

static void AtrInfo()

{

Type t = typeof(Test);

Console.WriteLine("\nСвойства, помеченные атрибутом:");

foreach (var x in t.GetProperties())

{

object atr\_obj;

if (GetAtr(x, typeof(Atr), out atr\_obj))

{

Atr atr1 = atr\_obj as Atr;

Console.WriteLine(x.Name + " - " + atr1.name);

}

}

}

static void Main(string[] args)

{

Test obj = new Test();

Type t = obj.GetType();

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

Console.WriteLine("Пространство имён: " + t.Namespace);

Console.WriteLine("Информация о сборке: " + t.AssemblyQualifiedName);

Console.WriteLine("Конструкторы: ");

foreach (var x in t.GetConstructors()) Console.WriteLine(x);

Console.WriteLine("Методы: ");

foreach (var x in t.GetMethods()) Console.WriteLine(x);

Console.WriteLine("Свойства: ");

foreach (var x in t.GetProperties()) Console.WriteLine(x);

Console.WriteLine("Поля данных: ");

foreach (var x in t.GetFields()) Console.WriteLine(x);

AtrInfo();

InvokeMemberInfo();

Console.ReadKey();

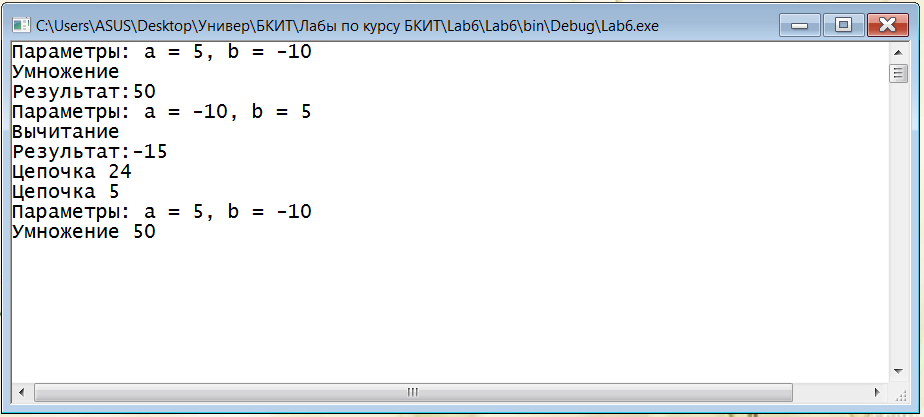
}

}

}

Пример выполнения программы

**Часть 1.**



**Часть 2.**

