**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

«Работа с делегатами»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Перова Анна Евгеньевна |  | Гапанюк Юрий  Евгеньевич |
| Подпись и дата: |  | Подпись и дата:  10.12.20 |

Москва, 2020 г.

Задание

**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.**

(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.

Текст программы

using System;

namespace laba6

{

delegate void PlusDelegate(string s, int p1, int p2);

class Program

{

static void Plus(string a, int b, int c)

{

int Result = b + c;

Console.WriteLine(a + Result.ToString());

}

static void PlusMethod1(string str, int p1, int p2, PlusDelegate pr)

{

pr(str, p1, p2);

}

static void PlusMethod2(string str, int p1, int p2, Action<string, int, int> pr)

{

pr(str, p1, p2);

}

static void Main(string[] args)

{

Console.WriteLine("Вызов метода с передачей в качестве параметра-делагата метод");

PlusMethod1("Плюс: ", 1, 2, Plus);

Console.WriteLine("Вызов метода с передачей в качестве параметра-делагата лямбда-выражение");

PlusMethod1("Плюс: ", 1, 2, (string a, int b, int c) => {

int Result = b + c;

Console.WriteLine(a + Result.ToString());

});

Console.WriteLine();

Console.WriteLine("Использование обобщенного делегата Action< >");

Console.WriteLine("Вызов метода с передачей в качестве параметра-делагата метод");

PlusMethod2("Плюс: ", 1, 2, Plus);

Console.WriteLine("Вызов метода с передачей в качестве параметра-делагата лямбда-выражение");

PlusMethod2("Плюс: ", 1, 2, (string a, int b, int c) => {

int Result = b + c;

Console.WriteLine(a + Result.ToString());

});

Console.ReadLine();

Console.ReadLine();

}

}

}

using System;

using System.Reflection;

namespace laba6\_2

{

class Program

{

static void Main(string[] args)

{

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

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

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

foreach (ConstructorInfo i in constructor)

Console.WriteLine(i);//выводим

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

MethodInfo[] Meth = 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 i in Meth)

Console.WriteLine(i);//выводим

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

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

foreach (PropertyInfo i in property)

Console.WriteLine(i);//выводим

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

foreach (PropertyInfo i in property)

{

object attributes = i.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(i);//выводим

}

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

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

}

}

class ReflClass

{

public ReflClass()

{

X = 1;

Y = 2;

}

public ReflClass(int a, int b)

{

X = a;

Y = b;

}

[ReflClass]

public int X { get; set; }

public int S { get; set; }

[ReflClass]

public int Y { get; set; }

public int Choose(bool f)

{

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

S = F;

return S;

}

private int Sum()

{

return X + Y;

}

private int Mult()

{

return X \* Y;

}

}

class ReflClassAttribute : System.Attribute//атрибут

{

}

}

Скриншоты с примером работы программы

