**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

**«**Работа с делегатами и рефлексией»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35б |  | доцент каф. ИУ5 |
| Кожиев Т.К. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

# Постановка задачи:

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:

//Program.cs (делегаты)

using System;

namespace lab6

{

public delegate string delegate1(int name1, string name2); // тип\_возвщаемого\_значения имя\_делегата(тип\_возващ\_аргум агумент)

class Program

{

static void Main(string[] args)

{

string str = Console.ReadLine();

int a = Convert.ToInt32(Console.ReadLine());

delegate1 mydelegate = method1;

mydelegate(a, str);

Console.WriteLine("----------------- Секция обычных делегатов ------------------");

method2(a, str, mydelegate);

Console.WriteLine("-----------------------------------");

delegate1 delegate2 = (x, y) => y + " " + x.ToString();

Console.WriteLine(delegate2(a, str));

Console.WriteLine("------------- Секция Action -------------");

Action<int, string> action\_delegate = (x, y) => Console.WriteLine(x.ToString() + " " + str);

Action<int, string> action\_delegate2 = method3;

action\_delegate(a, str);

action\_delegate2(a, str);

Console.WriteLine("------------------- Секция Func ----------------------");

Func<int, string, string> funcdelegat = method1;

method\_func(str, a, funcdelegat);

int x = 5;

bool y = false;

string z = "SOMEsTRING";

Func<int, string, bool, string> func\_delegat = (x, y, z) => x.ToString() + y.ToString() + z;

func\_delegat(x, z, y);

}

public static string method1(int a, string str)

{

string result = str + " " + a.ToString();

Console.WriteLine(result);

return result;

}

public static string method4(int a, string str1)

{

return str1;

}

public static void method3(int a, string str)

{

string result = str + " " + a.ToString();

Console.WriteLine(result);

}

public static void method2(int a, string str1, delegate1 mydeleg)

{

mydeleg(a, str1);

}

public static void method\_func(string str, int a, Func<int, string, string> func)

{

string i = str + func(a, str);

}

}

}

// Program.cs (Рефлексия)

using System;

using System.Reflection;

namespace lab6\_2

{

class Program

{

static void Main(string[] args)

{

ConstructorInfo[] constr = typeof(MyClass).GetConstructors();

for(int i = 0; i < constr.Length; i++)

{

Console.WriteLine(constr[i]);

}

MyClass myclass1 = new MyClass(1, 6);

MyClass myClass = new MyClass(6, 1);

bool myClassIsValid = ValidXorY(myClass);

bool myClass1IsValid = ValidXorY(myclass1);

Console.WriteLine($"Реультат валидации myClass: {myClassIsValid}");

Console.WriteLine($"Реультат валидации myclass1: {myClass1IsValid}");

}

static bool ValidXorY(MyClass myclass)

{

Type t = typeof(MyClass);

object[] attrs = t.GetCustomAttributes(false);

foreach (MyClassAttribute attr in attrs)

{

if (myclass.x >= attr.x) return true;

else return false;

}

return true;

}

}

}

MyClass.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_2

{

[MyClassAttribute(5)]

class MyClass

{

public int x { get; private set; }

public int y { get; private set; }

public MyClass()

{

this.x = 0;

this.y = 0;

}

public MyClass(int x)

{

this.x = x;

this.y = 0;

}

public MyClass(int x, int y)

{

this.x = x;

this.y = y;

}

public string getXY()

{

return x.ToString() + " " + y.ToString();

}

public void setX(int X)

{

this.x = X;

}

public void setY(int Y)

{

this.y = Y;

}

}

}

MyClassAttribute.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_2

{

public class MyClassAttribute : System.Attribute

{

public int x { get; set; }

public MyClassAttribute()

{ }

public MyClassAttribute(int X)

{

x = X;

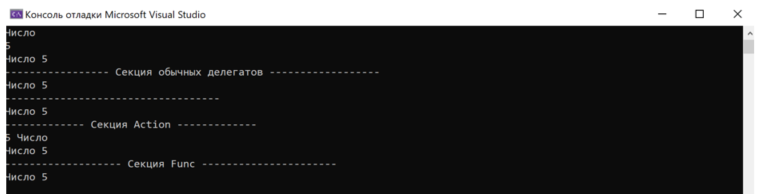
}

}

}

Тесты программы:

Делегаты



24.12.2020  
Кожиев Т.К.