**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»

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

Москва, 2020 г.

**Условие задания:**

**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.

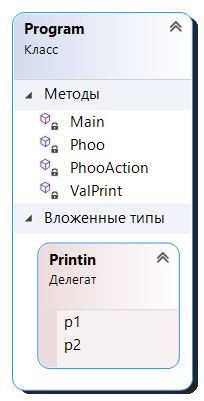
**Часть 2. Разработать программу, реализующую работу с рефлексией.**

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

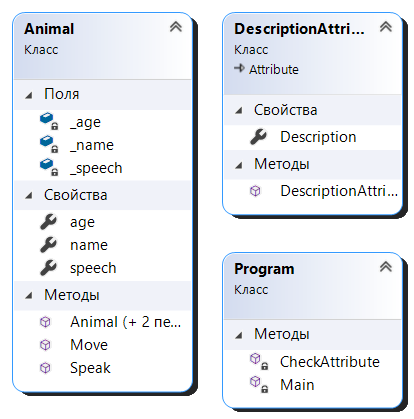
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс, содержащий конструкторы, свойства, методы.
3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
4. Создайте класс атрибута (унаследован от класса System.Attribute).
5.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

**Диаграмма классов:**

1)



2)



**Текст программы:**

1)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab06

{

class Program

{

// 1

delegate void Printin(int p1, double p2);

// 2

static void ValPrint(int p1, double p2)

{

Console.WriteLine("Your integer: " + p1);

Console.WriteLine("Your double: " + p2);

}

// 3.1

static void Phoo(Printin func, int p1, double p2)

{

func(p1, p2);

}

// 4.1

static void PhooAction(Action<int, double> func, int p1, double p2)

{

func(p1, p2);

}

static void Main(string[] args)

{

// 3.2

Phoo(ValPrint, 1, 2.1);

// 3.3

Printin lambda = (p1, p2) =>

{

Console.WriteLine("Sum of integer and double: " + (p1 + p2));

};

Phoo(lambda, 1, 1.1);

// 4.2

Action<int, double> lmbd = (p1, p2) =>

{

Console.WriteLine("Sum of integer and double: " + (p1 + p2));

};

PhooAction(ValPrint, 1, 1.1);

PhooAction(lmbd, 1, 2.1);

}

}

}

2)

using System;

using System.Reflection;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab06.Reflection

{

// 4

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]

class DescriptionAttribute : Attribute

{

public DescriptionAttribute() { }

public DescriptionAttribute(string description)

{

Description = description;

}

public string Description { get; set; }

}

// 2

class Animal

{

public Animal() { }

public Animal(int Age) { }

public Animal(int Age, string Name, string Speech) { age = Age; name = Name; speech = Speech; }

public void Speak()

{

Console.WriteLine(speech);

}

public void Move() { }

public int age

{

get { return \_age; }

set { \_age = value; }

}

private int \_age;

[DescriptionAttribute(Description = "Название этого животного.")]

public string name

{

get { return \_name; }

set { \_name = value; }

}

private string \_name;

[DescriptionAttribute("Звук, который издает это животное.")]

public string speech

{

get { return \_speech; }

set { \_speech = value; }

}

private string \_speech;

}

class Program

{

static bool CheckAttribute(PropertyInfo checkType, Type attributeType, out object attribute)

{

bool Res = false;

attribute = null;

var isAttribute = checkType.GetCustomAttributes(attributeType, false);

if (isAttribute.Length > 0)

{

Res = true;

attribute = isAttribute[0];

}

return Res;

}

static void Main(string[] args)

{

// 3

Type t = typeof(Animal);

Console.WriteLine("Конструкторы: ");

foreach (var i in t.GetConstructors())

{

Console.WriteLine(i);

}

Console.WriteLine("\nМетоды: ");

foreach (var i in t.GetMethods())

{

Console.WriteLine(i);

}

Console.WriteLine("\nСвойства: ");

foreach (var i in t.GetProperties())

{

Console.WriteLine(i);

}

// 5

Console.WriteLine("\nСвойства с атрибутом DescriptionAttribute:");

foreach (var i in t.GetProperties())

{

object attrObj;

if (CheckAttribute(i, typeof(DescriptionAttribute), out attrObj))

{

DescriptionAttribute attr = attrObj as DescriptionAttribute;

Console.WriteLine(i.Name + " - " + attr.Description);

}

}

// 6

Animal Cow = new Animal(5, "Cow", "Moooooo");

t.InvokeMember("Speak", BindingFlags.InvokeMethod, null, Cow, null);

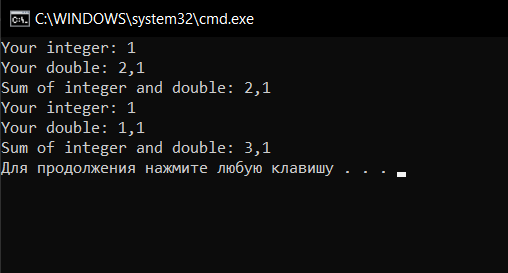
}

}

}

**Результаты:**

1)



2)

