**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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: |  |  |  |  |
| студент группы ИУ5-33Б |  |  |  |  | |
|  |  |  |  |  |
| Тарновский Даниил |  |  |  |  |

Москва, 2020 г.

**Условие Лабораторной работы:**

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

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

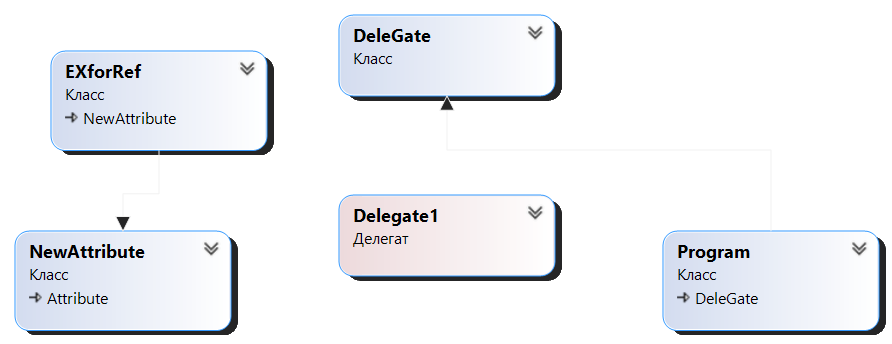
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат, 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:**



**Текст программы:**

DeleGate.cs

using System;

namespace ConsoleApp1

{

delegate bool Delegate1(double p1, float p2, int p3);

class DeleGate

{

public static bool AllPositive\_YorN(double p1, float p2, int p3)// проверка на подложительность всех чисел

{

if (p1 > 0 && p2 > 0 && p3 > 0)

return true;

else

return false;

}

public static bool CompositionPos\_YorN(double p1, float p2, int p3)// проверка на положительность произведения

{

if (p1 \* p2 \* p3 > 0)

return true;

else

return false;

}

public static void ActionTest(double i1, float i2, int i3)

{

Console.WriteLine("Сумма чисел: " + (i1 + i2 + i3));

}

public static void PosFun(string str, double p1, float p2, int p3, Delegate1 Param)

{

bool result = Param(p1, p2, p3);

Console.WriteLine(str + result.ToString());

}

public static void PosFunc1(string str, double i1, float i2,int i3, Func<double, float, int, bool> param)

{

bool Result = param(i1, i2,i3);

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

}

public static void PosFunc2(double i1, float i2, int i3, Action<double, float, int> param)

{

if (Math.Abs(i1) > 100 || Math.Abs(i2) > 100 || Math.Abs(i3) > 100)

param(i1, i2, i3);

else Console.WriteLine("Все значения меньше 100. Считайте сумму сами!");

}

}

}

EXforRef.cs

using System;

namespace ConsoleApp1

{

class EXforRef :NewAttribute

{

public EXforRef() { }

public EXforRef(int i) { }

public EXforRef(string str) { }

public int Plus(int x, int y) { return x + y; }

public int Minus(int x, int y) { return x - y; }

[NewAttribute("Описание для property1")]

public string property1

{

get { return \_property1; }

set { \_property1 = value; }

}

private string \_property1;

public int property2 { get; set; }

[NewAttribute(Description = "Описание для property3")]

public double property3 { get; private set; }

}

}

NewAttribute.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

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

public class NewAttribute : System.Attribute

{

public NewAttribute() { }

public NewAttribute(string DescriptionParam)

{

Description = DescriptionParam;

}

public string Description { get; set; }

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program : DeleGate

{

public static void TypeInfo()

{

EXforRef obj = new EXforRef();

Type t = obj.GetType();

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nКонструкторы:");

Console.ResetColor();

foreach (var x in t.GetConstructors())

{

Console.WriteLine(x);

}

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nМетоды:");

Console.ResetColor();

foreach (var x in t.GetMethods())

{

Console.WriteLine(x);

}

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nСвойства:");

Console.ResetColor();

foreach (var x in t.GetProperties())

{

Console.WriteLine(x);

}

}

/// <summary>

/// Пример использования метода InvokeMember

/// </summary>

static void InvokeMemberInfo()

{

Type t = typeof(EXforRef);

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nВызов метода:");

Console.ResetColor();

//Создание объект через рефлексию

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

//Параметры вызова метода

object[] parameters = new object[] { 3, 2 };

//Вызов метода

object Result = t.InvokeMember("Plus", BindingFlags.InvokeMethod, null, fi, parameters);

Console.WriteLine("Plus(3,2)={0}", Result);

}

/// <summary>

/// Проверка, что у свойства есть атрибут заданного типа

/// </summary>

/// <returns>Значение атрибута</returns>

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

{

bool Result = false;

attribute = null;

//Поиск атрибутов с заданным типом

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

if (isAttribute.Length > 0)

{

Result = true;

attribute = isAttribute[0];

}

return Result;

}

/// <summary>

/// Работа с атрибутами

/// </summary>

static void AttributeInfo()

{

Type t = typeof(EXforRef);

Console.ForegroundColor = ConsoleColor.Blue;

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

Console.ResetColor();

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);

}

}

}

static void Main(string[] args)

{

//ПРИМЕР ИСПОЛЬЗОВАНИЯ "Delegate"//

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"\n----------ПРИМЕР ИСПОЛЬЗОВАНИЯ Delegate----------");

Console.ResetColor();

#region

double p1 = -12.44;

float p2 = -14.3f;

int p3 = 200;

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Возьмем: p1= {p1} p2= {p2} p3= {p3}");

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine($"\nПример с методами в виде переменных:");

Console.ResetColor();

PosFun("Все ли числа положительны? ", p1, p2, p3, AllPositive\_YorN);

PosFun("Положительно ли произведение? ", p1, p2, p3, CompositionPos\_YorN);

Delegate1 dg1 = (double x, float y, int z) =>// проверим на положительность сумму

{

if (x + y + z > 0)

return true;

else return false;

};

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine($"\nПример с лямбда-выражением в виде переменной:");

Console.ResetColor();

PosFun("Положительна ли сумма?: ", p1, p2, p3, dg1);

double f1=-123.238723;

float f2=-1.1f;

int f3=-987;

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"\nВозьмем: f1= {f1} f2= {f2} f3= {f3}");

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nИспользование обощенного делегата Func<>");

Console.ResetColor();

PosFunc1("Все ли числа положительны?", f1, f2, f3, AllPositive\_YorN);

PosFunc1("Положительно ли произведение?", f1, f2, f3, CompositionPos\_YorN);

double a1 = -12.238723;

float a2 = -0.5f;

int a3 = -98;

double A1 = -72.333333;

float A2 = -111.1f;

int A3 = -7;

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"\nВозьмем: a1= {a1} a2= {a2} a3= {a3} A1= {A1} A2= {A2} A3= {A3}");

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("\nИспользование обощенного делегата Action<>");

Console.ResetColor();

Action<double, float, int> param;

param = ActionTest;

Console.WriteLine("При использование значений а1, а2, а3:");

PosFunc2( a1, a2, a3, param);

Console.WriteLine("\nПри использование значений А1, А2, А3:");

PosFunc2(A1, A2, A3, param);

#endregion

//ПРИМЕР ИСПОЛЬЗОВАНИЯ "REFLECTION"//

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine($"\n----------ПРИМЕР ИСПОЛЬЗОВАНИЯ Reflection----------");

Console.ResetColor();

TypeInfo();

InvokeMemberInfo();

AttributeInfo();

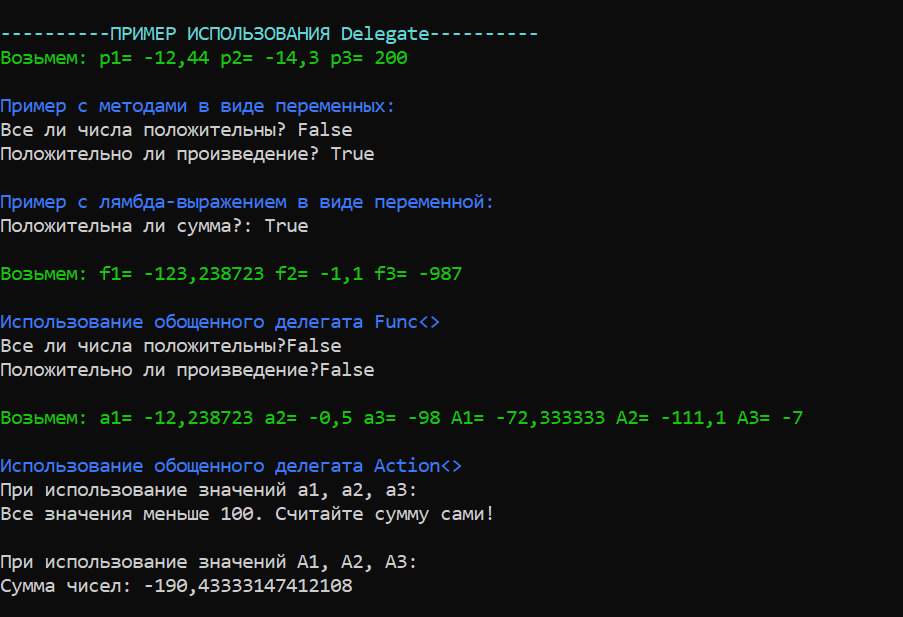
Console.ReadLine();

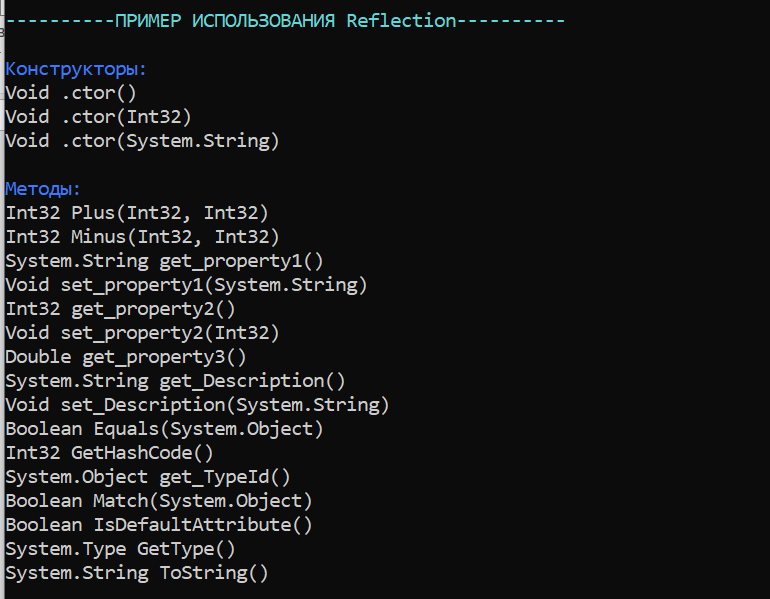
}

}

}

**Примеры выполнения программы:**

****

****

