Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»   
(МГТУ им. Н.Э. Баумана)

**Отчет**

**по дисциплине**

**“Базовые компоненты интернет технологий”**

**по лабораторной работе №6**

Выполнил: Окопный Марк Олегович

Группа: ИУ5 – 34

**Задачи:**

**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.

**Текст программы:**

**Programm.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using pryamougo;

namespace lab6

{

class Program

{

static double Plus(double p1, int p2)

{

return p1 + p2;

}

static double Multiply(double p1, int p2)

{

return p1 \* p2;

}

delegate double PlusOrMultiply(double p1, int p2);

static void PlusOrMultiplyMethod(string str, double p1, int p2, PlusOrMultiply PlusOrMultiplyParam)

{

double Result = PlusOrMultiplyParam(p1, p2); Console.WriteLine(str + Result.ToString());

}

static void PlusOrMultiplyMethod2(string str , double a, int b , Func<double,int,double> f )

{

double Result = f(a,b);

Console.WriteLine(str+Result);

}

static void Main(string[] args)

{

double m = 5.1;

int n = 4;

Console.WriteLine("m = " + m + " , n = " + n);

PlusOrMultiplyMethod(" Плюс:", m , n , Plus);

PlusOrMultiplyMethod(" Плюс:", m, n, Multiply);

PlusOrMultiplyMethod2(" Создание экземпляра делегата c Func<> " , m, n,

(double x, int y) =>

{

double z = x + y;

return z;

}

);

PlusOrMultiplyMethod2("Plus: ", m, n, Plus);

Console.WriteLine("\nПример группового делегата");

Action<double, int> a1 = (x, y) =>

{

Console.WriteLine("{0} + {1} = {2}",

x, y, x + y);

};

Action<double, int> a2 = (x, y) =>

{

Console.WriteLine("{0} - {1} = {2}",

x, y, x - y);

};

Action<double, int> a3 = (x, y) =>

{

Console.WriteLine("{0} \* {1} = {2}",

x, y, x\*y);

};

Action<double, int> group = a1 + a2;

group += a3;

Console.WriteLine("добавление вызова метода из группового делегата\n");

group(5.2, 3);

Console.WriteLine("\nудаление вызова метода из группового делегата\n");

group -= a3;

group(5.2, 3);

pryamoug p1 = new pryamoug(2, 3);

//System.Type tp1 = p1.GetType();

//System.Reflection.Assembly tp2 = typeof(System.Int32).Assembly;

//Mes MessageBoxsageBox.Show(tp1.ToString()+"\n"+tp2.ToString());

Console.WriteLine(Reflect.MethodReflectInfo<pryamoug>(p1));

Console.WriteLine("НАжмите любую клавишу для завершения");

Console.ReadKey();

}

}

}

**Pryamoyg.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pryamougo

{

class pryamoug

{

public double shir;

public double dlin;

public pryamoug(double dlin, double shir)

{

this.dlin = dlin;

this.shir = shir;

}

public override string ToString()

{

return base.ToString() + "\n" + "Длина прямоугольника: " + dlin + "\n" + "Ширина прямоугольника: " + shir + "\n" + "Площадь прямоугольника: " + shir \* dlin + "\n";

}

public void print()

{ Console.WriteLine(this.ToString()); }

}

}

**Reflect.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6

{

class Reflect

{

public static string MethodReflectInfo<T>(T element) where T : class

{

Type t = typeof(T);

List<string> constructors = new List<string>();

foreach (var x in t.GetConstructors())

{ constructors.Add(x.ToString()); }

List<string> metody = new List<string>();

foreach (var x in t.GetMethods())

{ metody.Add(x.ToString()); }

List<string> polya = new List<string>();

foreach (var x in t.GetFields())

{ polya.Add(x.ToString()); }

List<string> svva = new List<string>();

foreach (var x in t.GetProperties())

{ svva.Add(x.ToString()); }

string retstring = "Конструкторы: \n";

foreach (string c in constructors)

{ retstring += c + Environment.NewLine; }

retstring += "Методы: \n";

foreach (string c in metody)

{ retstring += c + Environment.NewLine; }

retstring += "Поля: \n";

foreach (string c in polya)

{ retstring += c + Environment.NewLine; }

retstring += "Свойства: \n";

foreach (string c in svva)

{ retstring += c + Environment.NewLine; }

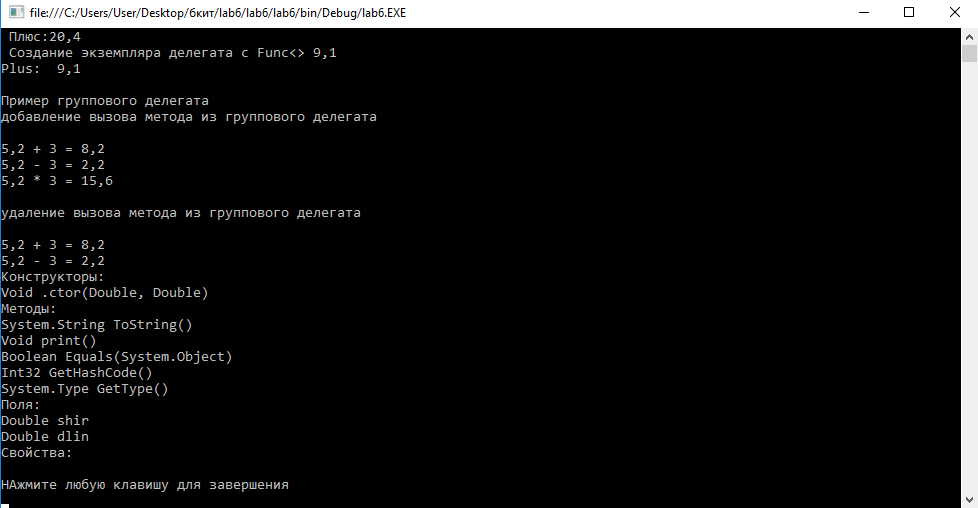
return retstring;

}

}

}

**Результат выполнения программы**

****