**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет технологий»

Отчет по лабораторной работе №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б |  | преподаватель каф. ИУ5 |
| Грызин Алексей |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

# Описание 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

# 

# Текст программы

using System;

using System.Reflection;

namespace lab6

{

// Делегат производящий математические операции над двумя числами

delegate string MathOperation(double num\_1, double num\_2);

partial class Program

{

// Часть 1

// Метод сложения (соответствующий нашему делегату)

static string Addition(double num\_1, double num\_2)

{

return (num\_1 + num\_2).ToString();

}

// Метод вычитания (соответствующий нашему делегату)

static string Subtraction(double num\_1, double num\_2)

{

return (num\_1 - num\_2).ToString();

}

// Метод деления (соответствующий нашему делегату)

static string Division(double num\_1, double num\_2)

{

return (num\_1 / num\_2).ToString();

}

// Метод с делегатным параметром

static string MethodMathOperation(double num\_1, double num\_2, MathOperation operation)

{

return operation(num\_1, num\_2);

}

// Метод принимающий в качестве одного из параметров обобщенный делегат Func

static string MethodMathOperationFunc(double num\_1, double num\_2, Func<double, double, string> operation)

{

return operation(num\_1, num\_2);

}

// Часть 2

// Проверка, что у свойства есть атрибут заданного типа

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;

}

// Получение информации о текущей сборке

static void AssemblyInfo()

{

Console.WriteLine("Вывод информации о сборке:");

Assembly i = Assembly.GetExecutingAssembly();

Console.WriteLine("Полное имя:" + i.FullName);

Console.WriteLine("Исполняемый файл:" + i.Location);

}

// Получение информации о типе

static void TypeInfo()

{

MyTestClass obj = new MyTestClass();

Type t = obj.GetType();

//другой способ

//Type t = typeof(MyTestClass);

Console.WriteLine("\nИнформация о типе:");

Console.WriteLine("Тип " + t.FullName + " унаследован от " + t.BaseType.FullName);

Console.WriteLine("Пространство имен " + t.Namespace);

Console.WriteLine("Находится в сборке " + t.AssemblyQualifiedName);

Console.WriteLine("\nКонструкторы:");

foreach (var x in t.GetConstructors())

{

Console.WriteLine(x);

}

Console.WriteLine("\nМетоды:");

foreach (var x in t.GetMethods())

{

Console.WriteLine(x);

}

Console.WriteLine("\nСвойства:");

foreach (var x in t.GetProperties())

{

Console.WriteLine(x);

}

Console.WriteLine("\nПоля данных (public):");

foreach (var x in t.GetFields())

{

Console.WriteLine(x);

}

}

// Пример использования метода InvokeMember

static void InvokeMemberInfo()

{

Type t = typeof(MyTestClass);

Console.WriteLine("\nВызов метода:");

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

//Параметры вызова метода

object[] parameters = new object[] { 19, 2020};

//Вызов метода

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

Console.WriteLine("YearOfBirth(20, 2020) = {0}", Result);

}

// Работа с атрибутами

static void AttributeInfo()

{

Type t = typeof(MyTestClass);

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

foreach (var x in t.GetProperties())

{

object attrObj;

if (GetPropertyAttribute(x, typeof(MyAttribute), out attrObj))

{

MyAttribute attr = attrObj as MyAttribute;

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

}

}

}

static void Main(string[] args)

{

Console.WriteLine("Алиев Тимур РТ5-31Б");

// Тестирование первой части лабораторной работы

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

double x = 5, y = 2;

Console.WriteLine($"x = {x}, y = {y}");

// Создание экземпляра нашего делегата

MathOperation add;

// Присваевыем в него нужный нам метод

add = Addition;

// И вызываем созанный нами объект делегата передав в него параметры

Console.WriteLine($"x + y = {add(x,y)}");

// Вызваем метод, который принимает делегат

Console.WriteLine($"x + y = {MethodMathOperation(x, y, Addition)}");

Console.WriteLine($"x - y = {MethodMathOperation(x, y, Subtraction)}");

// Передадим в этот же метод Лямбда-выражение

Console.WriteLine($"x \* y = {MethodMathOperation(x, y, (x, y) => (x \* y).ToString())}");

// Обобщенный делегат Func

Console.WriteLine($"x / y = {MethodMathOperationFunc(x, y, Division)}");

// Тестирование второй части лабораторной работы

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

// Создаю экземпляр тестового класса (MyTestClass), третьим параметром в конструктор передаю bool значение (true - муж., false - жен.)

MyTestClass person\_1 = new MyTestClass("Клава", "19", false);

MyTestClass person\_2 = new MyTestClass("Жорик", "15", true);

person\_1.GetInfo();

Console.WriteLine();

AssemblyInfo();

TypeInfo();

InvokeMemberInfo();

AttributeInfo();

}

}

}

# Результат работы программы

