**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»

отчёт

по лабораторной работе №6.

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-32 |  | преподаватель каф. ИУ5 |
| Честнова Е. А. |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2020 г.

Описание задания

Часть 1. Разработать программу, использующую делегаты.

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

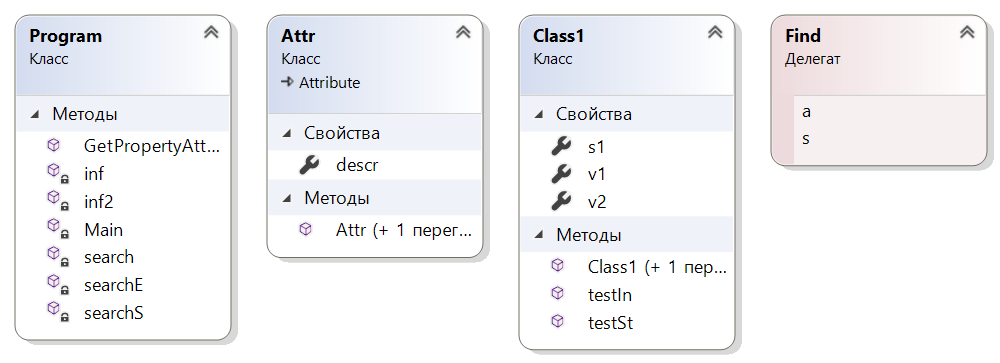
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат,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

{

delegate string Find(char a, string[] s);

class Program

{

static string searchS(char a, string[] s) {

if (Array.Exists(s, element => element.Trim().StartsWith(a)|| element.Trim().StartsWith(a.ToString().ToUpper()) )) return "Есть слова, начинающиеся на '" + a.ToString() +"'.";

else return "Нет слов, начинающихся на '" + a.ToString() + "'.";

}

static string searchE(char a, string[] s)

{

if (Array.Exists(s, element => element.EndsWith(a) ||element.EndsWith(a.ToString().ToUpper()) )) return "Есть слова, оканчивающиеся на '" + a.ToString() + "'.";

return "Нет слов, оканчивающихся на '" + a.ToString() + "'.";

}

static string search(char a, string[] s)

{

foreach (string str in s) {

if (str.ToLower().Contains(a)) return "Есть слова, содержащие '" + a.ToString() + "'.";

}

return "Нет слов, содержащих '" + a.ToString() + "'.";

}

static void inf(string i, char a, string [] s, Find FindPar) {

Console.WriteLine("\nСписок слов:");

foreach (string str in s) {

Console.WriteLine(str.Trim());

}

Console.WriteLine( i + '\n' + FindPar(a, s));

}

//использует обобщенный делегат, делит массив пополам

static void inf2(string i, char a, string[] s, Func<char, string[], string> FindPar) {

Console.WriteLine("\nСписок слов:");

Array.Resize(ref s, (int)s.Length/2);

foreach (string str in s)

{

Console.WriteLine(str.Trim());

}

Console.WriteLine(i + '\n' + FindPar(a, s));

}

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

{

bool Result = false;

attribute = null;

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

if (isAttribute.Length > 0)

{

Result = true;

attribute = isAttribute[0];

}

return Result;

}

static void Main(string[] args)

{

//ЧАСТЬ 1

string[] words = { " romE ", "mouse", "hEy", "DARK", " ice " };

string[] words2 = { "house", "bang", "Room", " light" };

char a;

Console.WriteLine("Введите букву:");

a=Char.ToLower((char)Console.Read());

inf("поиск по началу", a, words, searchS);

inf("поиск", a, words, search);

inf("поиск по концу", a, words2, searchE);

inf("поиск по концу ", a, words2, (char a, string [] s) =>

{

if (Array.Exists(s, element => element.EndsWith(a) || element.EndsWith(a.ToString().ToUpper()))) return "Есть слова, оканчивающиеся на '" + a.ToString() + "' .";

return "Нет слов, оканчивающихся на '" + a.ToString() + "'.";

});

inf2("поиск по началу", a, words, searchS);

inf2("поиск по началу", a, words, (char a, string[] s) =>

{

if (Array.Exists(s, element => element.Trim().StartsWith(a) || element.Trim().StartsWith(a.ToString().ToUpper()))) return "Есть слова, начинающиеся на '" + a.ToString() + "' .";

else return "Нет слов, начинающихся на '" + a.ToString() + "'.";

});

//ЧАСТЬ 2

Class1 ex = new Class1();

ex.v1 = 8;

Type info = ex.GetType();

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

Console.WriteLine("\nКонструкторы:");

foreach (var x in info.GetConstructors())

{

Console.WriteLine(x);

}

Console.WriteLine("\nМетоды:");

foreach (var x in info.GetMethods())

{

Console.WriteLine(x);

}

Console.WriteLine("\nСвойства:");

foreach (var x in info.GetProperties())

{

Console.WriteLine(x);

}

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

foreach (var x in info.GetProperties())

{

object attrObj;

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

{

Attr attr = attrObj as Attr;

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

}

}

object Res1 = info.InvokeMember("testSt", BindingFlags.InvokeMethod, null, ex, null);

Console.WriteLine("Вызов метода testSt \n" + Res1);

object Res2 = info.InvokeMember("testIn", BindingFlags.InvokeMethod, null, ex, null);

Console.WriteLine("Вызов метода testIn \n" + Res2.ToString());

}

}

class Class1

{

public Class1() { }

public Class1(string str) { }

public string testSt() {return "test";}

public double testIn() { return Math.Sqrt(v2); }

[Attr()]

public int v1 { get; set; }

public int v2 { get { return v1 \* 2; } }

[Attr("описание свойства s1")]

public string s1 { get; set; } = "propetry";

}

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

class Attr:Attribute

{

public Attr() { }

public Attr(string par) { descr = par; }

public string descr { get; set; } = "описание";

}

}

Анализ результатов

