**Министерство науки и высшего образования Российской Федерации** **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет** **имени Н. Э. Баумана** **(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Радиотехнический»**

**Кафедра «Системы обработки информации и управления»**

Отчет по лабораторной работе №6

по курсу БКИТ

Выполнила:

Студентка Группы РТ5-31Б   
Веденеева Екатерина Ильинична

подпись, дата

Проверил:

подпись, дата

2020г.

1. **Задание**

**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.
7. **Код**

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

namespace Lab6Sem3

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

Console.ReadLine();

}

}

}

Файл NewAtributes.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6Sem3

{

/// <summary>

/// Класс атрибута

/// </summary>

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

public class NewAttribute : Attribute

{

public NewAttribute() { }

public NewAttribute(string DescriptionParam)

{

Description = DescriptionParam;

}

public string Description { get; set; }

}

}

Файл Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

namespace Lab6Sem3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void SterterB\_Click(object sender, EventArgs e)

{

//Достаем данные из текствоксов

string str1 = textBox1.Text.Trim();

string str2 = textBox2.Text.Trim();

string intVal = textBox3.Text.Trim();

Delegates.TBConverter<string, string, int, string> Statement1 = Delegates.StrPlus;

Delegates.TBConverter<string, string, int, string> Statement2 = Delegates.StrPlus;

Delegates.TBConverter<Delegates.TBConverter<string, string, int, string>, string, string, string> Jora = Delegates.StatePlus;

int TxtInt = Convert.ToInt32(intVal); // Получаем числовое значение из последнего строкового

string sayThis;

//Две переменных, удобных для подстановки

string False = "ЛОЖЬ";

string True = "ИСТИНЫ";

// Какое-то условие для заключения, выводимого в конце строки

if (TxtInt == 0)

sayThis = " при коньюнкции дают 0.";

else if (TxtInt == 1)

sayThis = " при импликации дают 1";

else

sayThis = " <- это конец делегата, принимающего делегат";

// Вызываем и выводив первые два делегата

this.listBox1.Items.Add("Отдельные делегаты, созданные через шаблон и функцию:");

this.listBox1.Items.Add(Statement1(True, False, 0));

this.listBox1.Items.Add(Statement2(str1, str2, TxtInt));

////////////////// НЕРАБОТАЮЩАЯ ХРЕНЬ ////////////////////////////////

// Вызываем метод, принимающий делегат

// С помошью метода для другого делегата

this.listBox1.Items.Add("Метод, принимающий делегат в качетсве параметра - результат: ");

this.listBox1.Items.Add(Jora(Statement1, "КОТИКА", "КУСИ"));

//this.listBox1.Items.Add(Jora);

//С помощью лямбда-выражения

this.listBox1.Items.Add("Метод, принимающий делегат в качетсве параметра-делегата лямбда-выражение: ");

Delegates.TBConverter<Delegates.TBConverter<string, string, int, string>, string, string, string> TestStatement = (Delegates.TBConverter<string, string, int, string> f1, string f2, string f3) =>

{

string res = "ЭТО ПОЛНЫЙ БРЕД, НО НАДО ЧТО-ТО СЮДА ЗАПИСАТЬ: " + f1("ИСТИНЫ", "ЛОЖЬ", 1) + " и " + f2 + f3;

return res;

};

this.listBox1.Items.Add(TestStatement(Statement1, "ПРОВЕРКА 1", "ПРОВЕРКА 2"));

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void startRef\_Click(object sender, EventArgs e)

{

this.listBox2.Items.Add("Вывод информации о сборке:");

Assembly i = Assembly.GetExecutingAssembly();

this.listBox2.Items.Add("Полное имя:" + i.FullName);

this.listBox2.Items.Add("Исполняемый файл:" + i.Location);

this.listBox2.Items.Add("");

this.listBox2.Items.Add("");

Delegates obj = new Delegates();

Type File = obj.GetType();

//другой способ

//Type t = typeof(ForInspection);

this.listBox2.Items.Add("\nИнформация о типе:");

this.listBox2.Items.Add("Тип " + File.FullName + " унаследован от " + File.BaseType.FullName);

this.listBox2.Items.Add("Пространство имен " + File.Namespace);

this.listBox2.Items.Add("Находится в сборке " + File.AssemblyQualifiedName);

this.listBox2.Items.Add("");

this.listBox2.Items.Add("\nКонструкторы:");

foreach (var x in File.GetConstructors())

{

this.listBox2.Items.Add(x);

}

this.listBox2.Items.Add("");

this.listBox2.Items.Add("\nМетоды:");

foreach (var x in File.GetMethods())

{

this.listBox2.Items.Add(x);

}

this.listBox2.Items.Add("");

this.listBox2.Items.Add("\nСвойства:");

foreach (var x in File.GetProperties())

{

this.listBox2.Items.Add(x);

}

this.listBox2.Items.Add("");

this.listBox2.Items.Add("\nПоля данных (public):");

foreach (var x in File.GetFields())

{

this.listBox2.Items.Add(x);

}

this.listBox2.Items.Add("");

this.listBox2.Items.Add("\nForInspection реализует IComparable -> " + File.GetInterfaces().Contains(typeof(IComparable))

);

}

class Delegates : IComparable

{

public int CompareTo(object obj)

{

return 0;

}

//Делегат

public delegate T TBConverter<T0, T1, T2, T>(T0 d1, T1 d2, T2 int0);

public delegate string TBState(TBConverter<string, string, string, string> f1, TBConverter<string, string, string, string> f2);

//Методы, реализующие делегат (методы "типа" делегата)

public static string StrPlus(string str1, string str2, int i0)

{

return " из " + str1 + " следует " + str2 + " и поолучается " + i0.ToString();

}

public static string StatePlus(TBConverter<string, string, int, string> state1, string state2, string say)

{

return state1("АТЬ", "ДВА", 3) + state2 + say;

}

//Вызываем методы для заданных параметров

}

}

}

**3. Результат выполнения программы**

