Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

Выполнил:
студент группы ИУ5-31Б

Гришин Станислав
Подпись и дата:
21.12.2020

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:
21.12.2020

г. Москва, 2020 г.

Постановка задачи

Часть 1. Разработать программу, использующую делегаты.

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

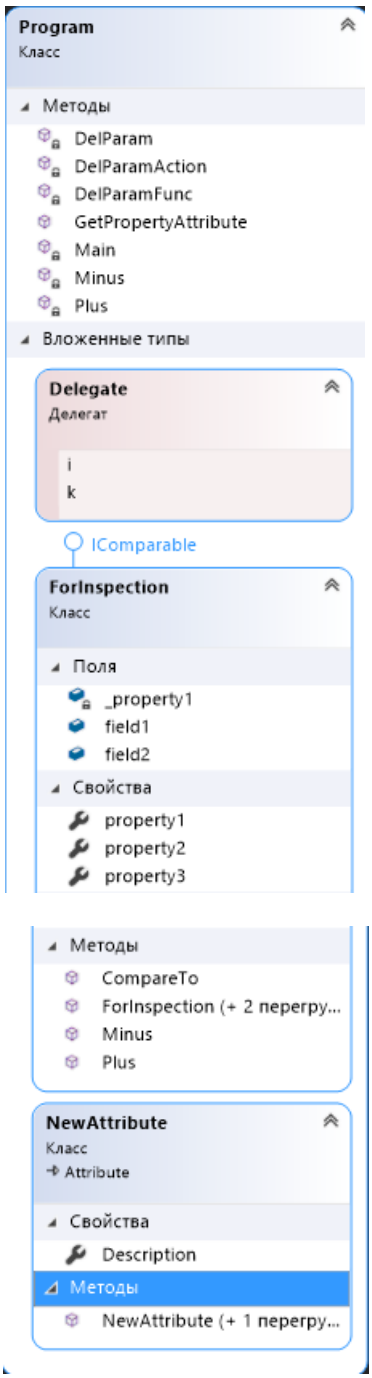
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат,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

Разработка интерфейса класса



Листинг программы

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Reflection;

namespace Laba6
{
    class Program
    {
        delegate object Delegate(int i, double k);

        static object Plus(int i, double k)
        {
            object m = i + k;
            return m;
```

```
        }
        static object Minus(int i, double k)
        {
            object m = i - k;
            return m;
        }
        static object DelParam(int i,double k,Delegate Param)
        {
            object par = Param(i, k);
            return par;
        }
        static object DelParamFunc(int i, double k, Func<int, double, object> Param)
        {
            object par = Param(i, k);
            return par;
        }
        static void DelParamAction(int i, double k, Action<int, double> Param)
        {
            Param(i, k);
        }
        public class ForInspection : IComparable
        {
            public ForInspection() { }
            public ForInspection(int i) { }
            public ForInspection(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; }
            public int field1;
            public float field2;
            public int CompareTo(object obj)
            {
                return 0;
            }
        }
        [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited =
false)]
        public class NewAttribute : Attribute
        {
            public NewAttribute() { }
            public NewAttribute(string DescriptionParam)
            {
                Description = DescriptionParam;
            }
            public string Description { get; set; }

        }
        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;
        }

        static void Main(string[] args)
        {
            Console.WriteLine("1 ЧАСТЬ");
            object A = DelParam(5, 4.5, Plus);
            Console.WriteLine("Сумма двух чисел равна " + A.ToString());
            A = DelParam(5, 4.5, Minus);
            Console.WriteLine("Разность двух чисел равна " + A.ToString());
            A = DelParam(5, 4.5, (x,y)=>x*y);
            Console.WriteLine("Использование лямбда-выражения " + A.ToString());
            A = DelParamFunc(5, 4.5, Plus);
            Console.WriteLine("Использование обобщенного делегата Func " + A.ToString());
            A = DelParamFunc(5, 4.5, (x, y) => x * y);
            Console.WriteLine("Использование обобщенного делегата Func и лямбда-выражения
" + A.ToString());
            Console.WriteLine("****************************************");
            Console.WriteLine("2 ЧАСТЬ");
            ForInspection obj = new ForInspection();
            Type t = obj.GetType();
            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);
            }
            t = typeof(ForInspection);
            Console.WriteLine("\nСвойства, помеченные атрибутом:");
            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);
                }
                Console.WriteLine("\nВызов метода:");
                ForInspection fi = (ForInspection)t.InvokeMember(null,
BindingFlags.CreateInstance, null, null, new object[] { });
                object[] parameters = new object[] { 3, 2 };
```

```
                object Result = t.InvokeMember("Plus", BindingFlags.InvokeMethod,null,
fi, parameters);
                Console.WriteLine("Plus(3,2)={0}", Result);
            }
        }
    }
                                        }
```

Анализ результатов

```
Консоль отладки Microsoft Visual Studio
1 ЧАСТЬ
Сумма двух чисел равна 9,5
Разность двух чисел равна 0,5
Использование лямбда-выражения 22,5
Использование обобщенного делегата Func 9,5
Использование обобщенного делегата Func и лямбда-выражения 22,5
**************************************
2 ЧАСТЬ

Информация о типе:
Тип Laba6.Program+ForInspection унаследован от System.Object
Пространство имен Laba6
Находится в сборке Laba6.Program+ForInspection, Laba6, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

Конструкторы:
Void .ctor()
Void .ctor(Int32)
Void .ctor(System.String)

Методы:
Int32 Plus(Int32, Int32)
Int32 Minus(Int32, Int32)
System.String get_property1()
Void set_property1(System.String)
Int32 get_property2()
Void set_property2(Int32)
Double get_property3()
Int32 CompareTo(System.Object)
System.Type GetType()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()

Свойства:
System.String property1
Int32 property2
Double property3

Поля данных (public):
Int32 field1
Single field2
```

```
Свойства, помеченные атрибутом:
property1 - Описание для property1

Вызов метода:
Plus(3,2)=5

Вызов метода:
Plus(3,2)=5
property3 - Описание для property3

Вызов метода:
Plus(3,2)=5
```