МГТУ имени Баумана

Факультет «Информатика и Системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»

Дисциплина «Базовые компоненты интернет технологий»

Отчет по лабораторной работе №6

Выполнила

студентка группы

ИУ5-34б

Слободчикова Юлия

Москва 2020

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.

Текст программы:

Первая часть

```
using System;

namespace LAB_6
{
    class Program
    {

        delegate double Oper(double a, int b); //делегат, принимающий несколько
параметров различных
```

```
                                                    //типов и возвращающий значение
произвольного типа.
        static void Main()
        {

            Console.WriteLine("Вызов через метод");
            //Вызов через метод
            PlusMinus("Плюс", 1.2, 3, Plus);
            Console.WriteLine("Вызов через лямбда");
            // вызов через лямбда
            PlusMinus("Плюс", 4.5, 6, (a, b) => a + b);
            //Func
            Console.WriteLine("Вызов через Func");

            PlusMinus1("Плюс", 7.8, 9, Plus);
            Console.ReadLine();
        }
        static double Plus(double a, int b) //метод, соответствующий данному делегату.
        {
            return a + b;
        }
        static void PlusMinus(        //метод, принимающий разработанный делегат
            string str, double a, int b,
            Oper param)
        {
            double Result = param(a, b);
            Console.WriteLine(str + ' ' + '=' + ' ' + Result.ToString());
        }
        static void PlusMinus1(            //вместо обычного делегата передаем обобщенный
                string str, double a, int b,
                Func<double, int, double> param)
        {
            double Result = param(a, b);
            Console.WriteLine(str + ' ' + '=' + ' ' + Result.ToString());
        }
    }
     }
```

Вторая часть

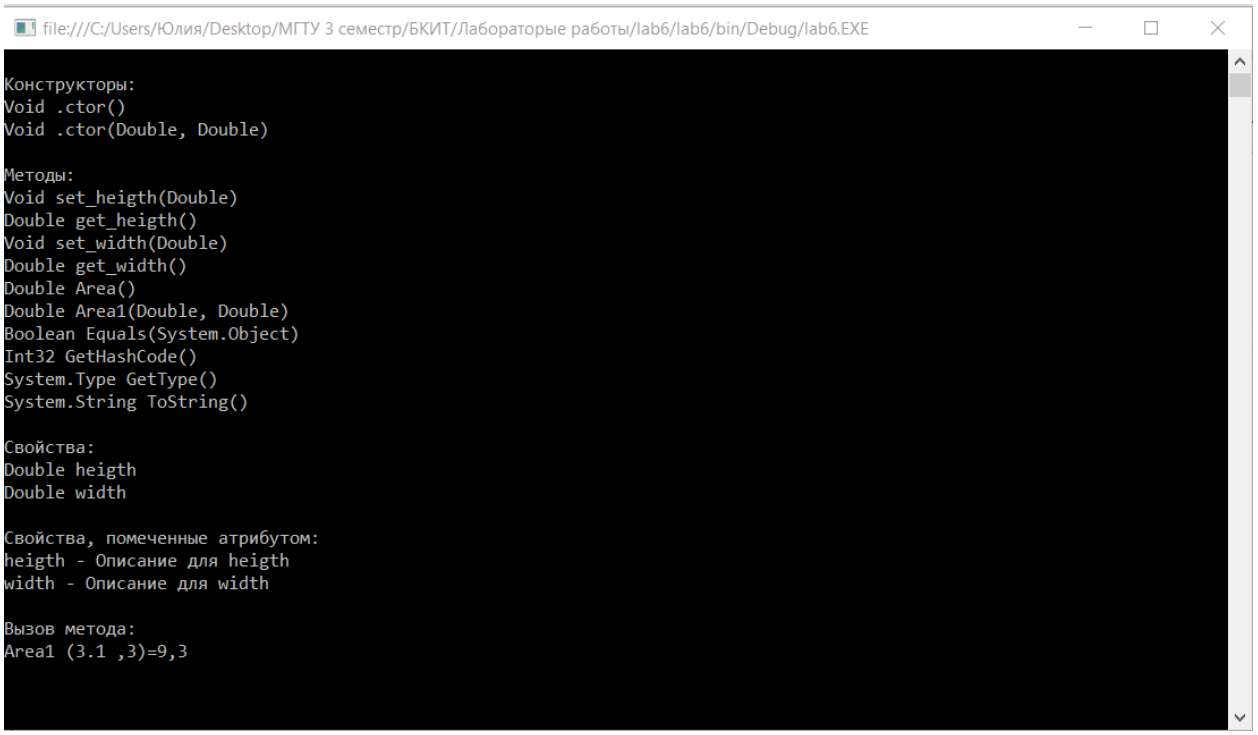
```
using System;
using System.Reflection;

namespace Reflectiom
{
    class Program
    {
        static void Main(string[] args)
        {
            Rectangle obj = new Rectangle(3.1, 3.2);
            Type t = obj.GetType();
            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);
```

```
            }

            //содержат ли свойства  атрибут NewAttribute
            Type t1 = typeof(Rectangle);
            Console.WriteLine("\nСвойства, помеченные атрибутом:");
            foreach (var x in t1.GetProperties())
            {
                object attrObj;
                if (GetPropertyAttribute(x, typeof(NewAttribute), out attrObj))
                {
                    NewAttribute attr = attrObj as NewAttribute;// Для приведения
полученного значения типа object к требуемому типу NewAttribute
                    Console.WriteLine(x.Name + " - " + attr.Description);
                }
            }

            //вызов метода с использованием рефлексии
            Type t2 = typeof(Rectangle);
            Console.WriteLine("\nВызов метода:");
            //Создание объекта
            //ForInspection fi = new ForInspection();
            //Можно создать объект через рефлексию
            Rectangle fi = (Rectangle)t.InvokeMember(null, BindingFlags.CreateInstance,
null, null, new object[] { });
            //Параметры вызова метода
            object[] parameters = new object[] { 3.1, 3 };
            //Вызов метода
            object Result = t2.InvokeMember("Area1", BindingFlags.InvokeMethod, null, fi,
parameters);
            Console.WriteLine("Area1 (3.1 ,3)={0}", Result);
            // Метод InvokeMember принимает различные
            Console.ReadLine();

        }

        public class Rectangle
        {

            [NewAttribute("Описание для heigth")] //пункт 5  Назначьте атрибут некоторым
свойствам классам

            public double heigth
            {
                set
                {
                    if (value < 0)
                        Console.WriteLine("Введено некоректное значение высоты");
                    else _heigth = value;
                }
                get { return _heigth; }

            }
            private double _heigth;


            [NewAttribute(Description = "Описание для width")]
            public double width
            {
                set
                {
                    if (value < 0)
                        Console.WriteLine("Введено некоректное значение ширины");
                    else
                        _width = value;
                }
```

```
            get { return _width; }
        }
        private double _width;
        public Rectangle() { }
        public Rectangle(double width, double heigth)
        {
            this.width = width;
            this.heigth = heigth;

        }

        public double Area()
        {
            return this.heigth * this.width;
        }

        public double Area1(double heigth, double width)
        {
            return heigth * width;
        }


    }

    /// 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; }
    }

    public static bool GetPropertyAttribute(PropertyInfo checkType, Type// информация
о проверяемом свойстве,
attributeType, out object attribute)//тип проверяемого атрибута
    {
        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;
    }


}
 }
```

Экранные формы с примерами выполнения программы:


file:///C:/Users/Юлия/Desktop/МГТУ 3 семестр/БКИТ/Лабораторые работы/lab6/lab6/bin/Debug/lab6.EXE
Вызов через метод
Плюс = 4,2
Вызов через лямбда
Плюс = 10,5
Вызов через Func
Плюс = 16,8


file:///C:/Users/Юлия/Desktop/МГТУ 3 семестр/БКИТ/Лабораторые работы/lab6/lab6/bin/Debug/lab6.EXE
Конструкторы:
Void .ctor()
Void .ctor(Double, Double)
Методы:
Void set_heigth(Double)
Double get_heigth()
Void set_width(Double)
Double get_width()
Double Area()
Double Area1(Double, Double)
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
System.String ToString()
Свойства:
Double heigth
Double width
Свойства, помеченные атрибутом:
heigth - Описание для heigth
width - Описание для width
Вызов метода:
Area1 (3.1 ,3)=9,3