Защищено:
Гапанюк Ю.Е.

"__"_____2017 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2017 г.

Отчет по лабораторной работе № 6 по курсу Базовые компоненты интернет-технологий

9
(количество листов)

Студент группы ИУ5-33
Коционова Анна

Москва МГТУ 2017

СОДЕРЖАНИЕ

1. Описание задания лабораторной работы

Лабораторная работа №6

Часть 1. Разработать программу, использующую делегаты.

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.

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.

2. Текст программы

Main 6_1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab6
{
    // определение делегата
    delegate double SqrtOrPow(float p1, float p2);

    class Program
    {
        //метод, соответствующий делегату
        static double Sqrt(float a, float b)
        {
            return System.Math.Pow(a, (1 / b));
        }
        static double Power(float a, float b)
        {
            return System.Math.Pow(a, b);
        }

        //метод, принимающий делегат как один из параметров
        static void SqrtOrPowMethod(string str, float i1, float i2, SqrtOrPow
SqrtOrPowParam)
        {
            double Result = SqrtOrPowParam(i1, i2);
            Console.WriteLine(str + Result.ToString());
        }

        //метод, принимающий обобщенный делегат Func<>
        static void SqrtOrPowMethodFunc(string str, float i1, float i2,
            Func<float, float, double> SqrtOrPowParam)
        {
            double Result = SqrtOrPowParam(i1, i2);
            Console.WriteLine(str + Result.ToString());
        }

        static void Main(string[] args)
        {//вызов метода с праметром-делегатом
            int i1 = 4;
            int i2 = 2;
            SqrtOrPowMethod("Корень: ", i1, i2, Sqrt);
            SqrtOrPowMethod("Степень: ", i1, i2, Power);
            SqrtOrPowMethodFunc("Корень(обобщенным): ", i1, i2, Sqrt);
            SqrtOrPowMethodFunc("Степень(обобщенным): ", i1, i2, Power);
            // вызов метода с параметром - лямбда-выражением
            SqrtOrPowMethod("Создание экземпляра делегата на основе лямбда-выражения 1:
", i1, i2,
                (x, y) =>
                {
                    double z = Sqrt(x,y);
                    return z;
                }
           );
            SqrtOrPowMethod("Создание экземпляра делегата на основе лямбда-выражения 2:
", i1, i2,
```

```
                (float x, float y) =>
                {
                    double z = Power(x,y);
                    return z;
                }
            );
             //вызов метода с параметром - обобщенным делегатом(преимущество в том, что
             // его не надо объявлять в тексте программы(библиотечный)
             SqrtOrPowMethodFunc("Создание экземпляра делегата на основе лямбда-выражения
3: ", i1, i2,
                (float x, float y) =>
                {
                    double z = Sqrt(x,y);
                    return z;
                }
                );
             SqrtOrPowMethodFunc("Создание экземпляра делегата на основе лямбда-выражения
4: ", i1, i2,
                (float x, float y) =>
                {
                    double z = Power(x,y);
                    return z;
                }
                );
             Console.ReadLine();
        }
    }
}
```

Main 6_2

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

namespace Reflection
{
    class Program
    {
        /// <summary>
        /// Проверка, что у свойства есть атрибут заданного типа
        /// </summary>
        /// <returns>Значение атрибута</returns>
        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)
```

```
        {
            Type t = typeof(ForInspection);

            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);
            }

            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:");

            //Создание объекта
            //ForInspection fi = new ForInspection();

            //Можно создать объект через рефлексию
            ForInspection fi = (ForInspection)t.InvokeMember(null,
BindingFlags.CreateInstance, null, null, new object[] { });

            //Параметры вызова метода
            object[] parameters = new object[] { 3, 2 };

            //Вызов метода
            object Result = t.InvokeMember("Sqrt", BindingFlags.InvokeMethod, null, fi,
parameters);
            Console.WriteLine("Sqrt(3,2)={0}", Result);

            Console.ReadLine();
        }
    }
}
```

Класс NewAttribute

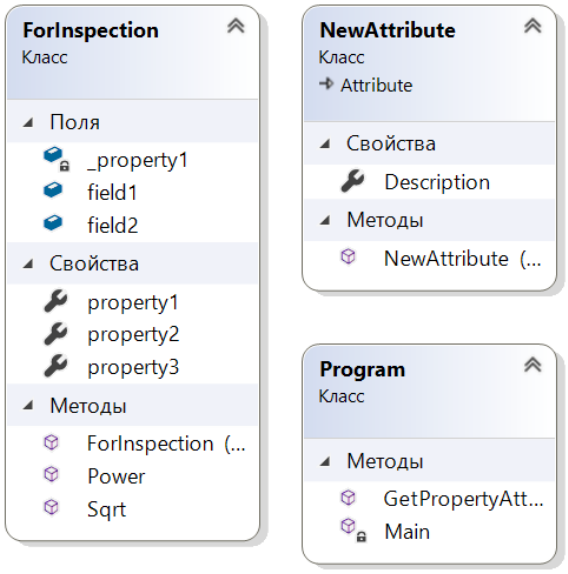
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Reflection
{
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]
    public class NewAttribute : Attribute
    {
        public NewAttribute() { }
        public NewAttribute(string DescriptionParam)
        {
            Description = DescriptionParam;
        }
        public string Description { get; set; }
    }
}
```

Класс ForInspection

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Reflection
{
    /// <summary>
    /// Класс для исследования с помощью рефлексии
    /// </summary>
    public class ForInspection
    {
        public ForInspection() { }
        public ForInspection(int i) { }
        public ForInspection(string str) { }
        public double Sqrt(float a, float b)
        { return System.Math.Pow(a, (1 / b)); }
            public double Power(float a, float b)
        { return System.Math.Pow(a, b); }

        [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;
    }
}
```

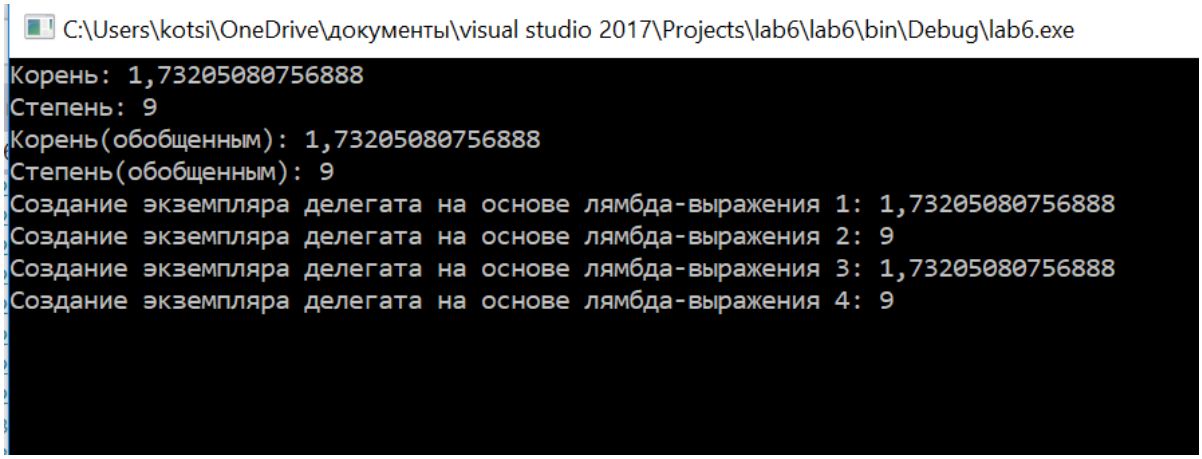
3. Диаграмма классов

Для 2-ой части ЛР

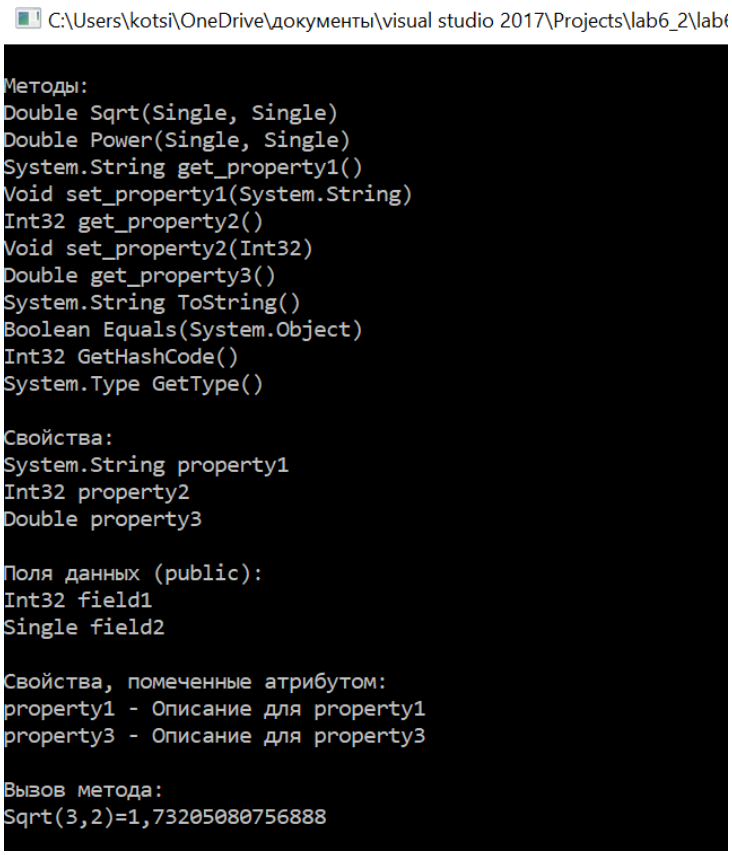


4. Результат выполнения программы

1-ая часть ЛР(работа с делегатами):



2-ая часть ЛР(работа с рефлексией):


C:\Users\kotsi\OneDrive\документы\visual studio 2017\Projects\lab6_2\lab
Методы:
Double Sqrt(Single, Single)
Double Power(Single, Single)
System.String get_property1()
Void set_property1(System.String)
Int32 get_property2()
Void set_property2(Int32)
Double get_property3()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
Свойства:
System.String property1
Int32 property2
Double property3
Поля данных (public):
Int32 field1
Single field2
Свойства, помеченные атрибутом:
property1 - Описание для property1
property3 - Описание для property3
Вызов метода:
Sqrt(3,2)=1,73205080756888