

**Отчет по лабораторной работе № 6 по курсу
«Базовые компоненты интернет технологий»**

Исполнил:
студент группы ИУ5-33
Ханмагомедов Ренат

Дата: 27.12.2017

Москва МГТУ 2017

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 входных 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

Для второй части:



Текст программы:

1 часть

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

namespace lab6
{
    delegate float mulordiv(int p1, int p2);

    class Program
    {
        static float mul(int p1, int p2) { return p1 * p2; }
        static float div(int p1, int p2) { return p1 / p2; }
        static void mulordivMethod(string str, int i1, int i2, mulordiv mulordivParam)
        {
            float Result = mulordivParam(i1, i2);
            Console.WriteLine(str + Result.ToString());
        }
        static void mulordivMethodFunc(string str, int i1, int i2, Func<int, int, float>
mulordivParam)
        {
            float Result = mulordivParam(i1, i2);
            Console.WriteLine(str + Result.ToString());
        }
        /* static void mulordivMethodAction(string str, int i1, int i2, Action<int,
int> mulordivParam)
        {
            void Result = mulordivParam(i1, i2);
            Console.WriteLine(str + Result.ToString());
        }
        */

        static void Main(string[] args)
        {
            int i1 = 6;
            int i2 = 2;

            mulordivMethod("Умножение: ", i1, i2, mul);
            mulordivMethod("Деление: ", i1, i2, div);

            mulordivMethod("Создание экземпляра делегата на основе лямбда-выражения 1: ",
i1, i2,
            (int x, int y) =>
            {
                int z = x * y;
                return z;
            }
            );
            mulordivMethod("Создание экземпляра делегата на основе лямбда-выражения 2: ",
i1, i2,
            (x, y) =>
            {
                return x * y;
            }
            );
            mulordivMethod("Создание экземпляра делегата на основе лямбда-выражения 3: ",
i1, i2, (x, y) => x*y );
        }
    }
}
```

```

        Console.WriteLine("\nИспользование обобщенного делегата Func<>");
        mulordivMethodFunc("Создание экземпляра делегата на основе метода:", i1, i2,
div);
        Console.WriteLine("\nИспользование обобщенного делегата Action<>");
        //mulordivMethodAction("Создание экземпляра делегата на основе метода:", i1,
i2, div);
    }
}
}

```

2 часть

```

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

namespace ConsoleApp8
{
    class Program
    {
        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 рефлексия
    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("Plus", BindingFlags.InvokeMethod,
null, fi, parameters);
    Console.WriteLine("Plus(3,2)={0}", Result);
    Console.ReadLine();
}
}
}

```

class NewAttribute

```
using System;
```

```
namespace ConsoleApp8
```

```
{
```

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

```
public class NewAttribute : Attribute
```

```
{
```

```
    public NewAttribute() { }
```

```
    public NewAttribute(string DescriptionParam)
```

```
    {
```

```
        Description = DescriptionParam;
```

```
    }
```

```
    public string Description { get; set; }
```

```
}
```

```
}
```

class ForInspection

```
namespace ConsoleApp8
```

```
{
```

```
    public class ForInspection
```

```
    {
```

```
        public ForInspection() { }
```

```
        public ForInspection(int i) { }
```

```
        public ForInspection(string str) { }
```

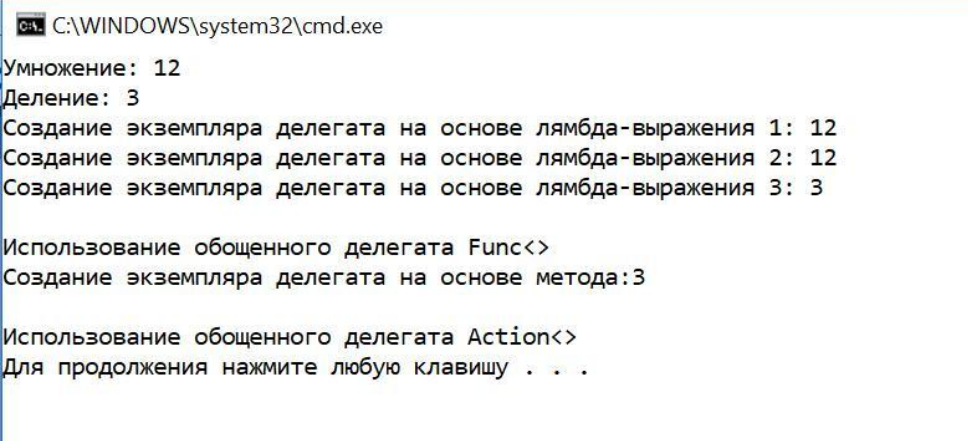
```

        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;
    }
}

```

Экранные формы с примерами выполнения программы

1 часть



```

C:\WINDOWS\system32\cmd.exe
Умножение: 12
Деление: 3
Создание экземпляра делегата на основе лямбда-выражения 1: 12
Создание экземпляра делегата на основе лямбда-выражения 2: 12
Создание экземпляра делегата на основе лямбда-выражения 3: 3

Использование обобщенного делегата Func<>
Создание экземпляра делегата на основе метода:3

Использование обобщенного делегата Action<>
Для продолжения нажмите любую клавишу . . .

```

2 часть

 Выбрать C:\WINDOWS\system32\cmd.exe

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()
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 property
property3 - Описание для property
```

Вызов метода:

```
Plus(3,2)=5
```



