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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

Выполнил:
студент группы ИУ5-32
Кудрявцев Сергей
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.С.
Подпись и дата:

г. Москва, 2018 г.

Описание задания:
Часть 1. Разработать программу, использующую делегаты.

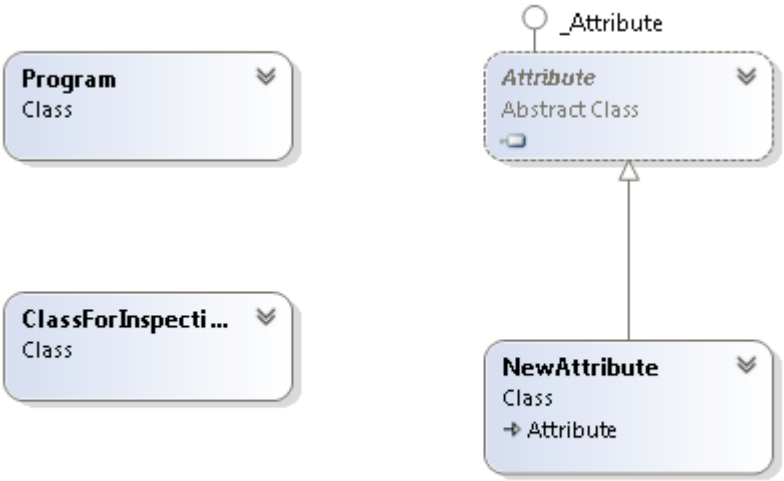
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат,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

Диаграмма классов:
Часть 1.

Часть 2.



Текст программы:

Часть 1.

Program.cs:

```
using System;
using System.Security.Cryptography.X509Certificates;

namespace Lab6
{
    delegate string StringJob(string p1, int p2);

    internal class Program
    {
        static void TestFunction(string desription, string p1, int p2, StringJob func)
        {
            Console.Write(desription + ": ");
            Console.WriteLine(func(p1, p2));
        }

        static void TestFunction1(string desription, string p1, int p2, Func<string, int, string>
func)
        {
            Console.Write(desription + ": ");
            Console.WriteLine(func(p1, p2));
        }

        static string StringJobFunc(string p1, int p2)
```

```
        {
            string tmp = "";
            for (int i = 0; i < p2; ++i)
            {
                tmp += p1;
            }

            return tmp;
        }

        public static void Main(string[] args)
        {
            string p1 = "Привет";
            int p2 = 3;

            Console.WriteLine("Делегат StringJob:");

            TestFunction("Передача функции", p1, p2, StringJobFunc);
            TestFunction("Передача лямбда-выражения", p1, p2, (x, y) => x + y.ToString());

            Console.WriteLine("\nОбобщенный делегат:");

            TestFunction1("Передача функции", p1, p2, StringJobFunc);
            TestFunction1("Передача лямбда-выражения", p1, p2, (x, y) => x + y.ToString());
        }
    }
}
```

Часть 2.

Program.cs:

```
using System;
using System.Reflection;

namespace Lab6_2
{
    internal class Program
    {
        public static bool GetPropertyAttribute(PropertyInfo checkType,
            Type attributeType, out object attribute)
        {
            bool Result = false;
            attribute = null;

            var isAttribute = checkType.GetCustomAttributes(attributeType, false);
            if (isAttribute.Length > 0)
            {
                Result = true;
                attribute = isAttribute[0];
            }

            return Result;
        }

        public static void Main(string[] args)
        {
            Type t = typeof(ClassForInspection);

            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);
            }

            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 с помощью рефлексии:");

            ClassForInspection cfi = new ClassForInspection();
            object[] parameters = new object[] { 3.0, 2.0 };

            object result = t.InvokeMember("Divide", BindingFlags.InvokeMethod, null, cfi,
parameters);
            Console.WriteLine($"Divide(3.0, 2.0) = {result}");
        }
    }
}
```

ClassForInspection.cs:

```
namespace Lab6_2
{
    public class ClassForInspection
    {
        public ClassForInspection() { }
        public ClassForInspection(string str) { }
        public ClassForInspection(double d) { }

        public double Divide(double a, double b)
        {
            return a / b;
        }

        [NewAttribute("Описание для prop1")]
        public int prop1 { get; set; }

        [NewAttribute("Описание для prop2")]
        public int prop2 { get; set; }

        public int prop3 { get; set; }

        public int[] data;
    }
```

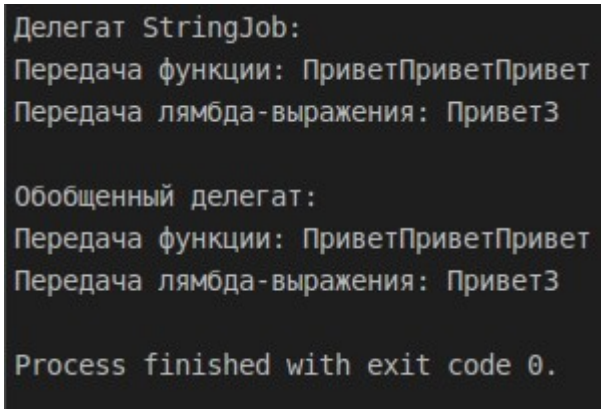
```
}
```

NewAttribute.cs:

```
using System;

namespace Lab6_2
{
    [AttributeUsage(AttributeTargets.Property,
        AllowMultiple = false,
        Inherited = false)]
    public class NewAttribute : Attribute
    {
        public NewAttribute() { }

        public NewAttribute(string DescriptionParam)
        {
            Description = DescriptionParam;
        }

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

Примеры:

Часть 1.



Часть 2.

```
Информация о типе:
Тип Lab6_2.ClassForInspection унаследован от System.Object
Пространство имен Lab6_2
Находится в сборке Lab6_2.ClassForInspection, Lab6_2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

Конструкторы:
Void .ctor()
Void .ctor(String)
Void .ctor(Double)

Методы:
Double Divide(Double, Double)
Int32 get_prop1()
Void set_prop1(Int32)
Int32 get_prop2()
Void set_prop2(Int32)
Int32 get_prop3()
Void set_prop3(Int32)
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
System.String ToString()

Свойства:
Int32 prop1
Int32 prop2
Int32 prop3

Поля данных (public):
System.Int32[] data

Свойства, помеченные атрибутом:
prop1 - Описание для prop1
prop2 - Описание для prop2

Вызов метода с помощью рефлексии:
Divide(3.0, 2.0) = 1.5

Process finished with exit code 0.
```