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

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Базовые компоненты интернет-технологий»

Отчет по ЛР6

Выполнил:

студент группы ИУ5-31Б

Вардумян Арсен

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Подпись и дата:

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).

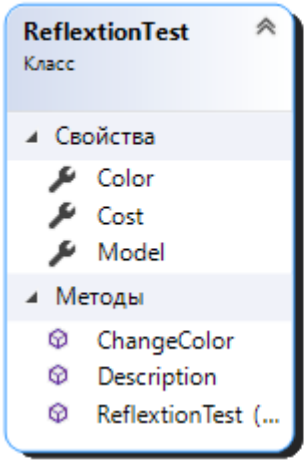
5.

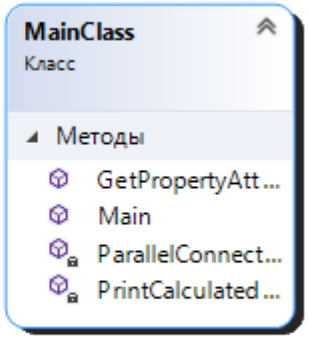
Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.

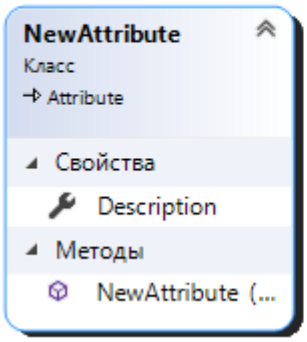
6.

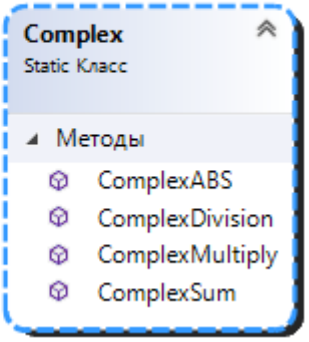
Вызовите один из методов класса с использованием рефлексии.

Разработка интерфейса класса









Листинг программы

//Program.cs

```
using System;
using System.Reflection;

namespace Lab_6
{
    using complex = System.ValueTuple<double, double>;
    class MainClass
    {
        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;
        }

        //delegate string DetermineResistance((double, double) complex1,
(double,double) complex2, string units);

        static string ParallelConnection(complex complex1, complex complex2,
string units)
        {
            var sum = Complex.ComplexSum(complex1, complex2);
            var mult = Complex.ComplexMultiply(complex1, complex2);
            var div = Complex.ComplexDivision(mult, sum);

            return Complex.ComplexABS(div).ToString() + "\t" + units;
        }

        //здесь исползуем обобщенный делегат Func<>
        static void PrintCalculatedResistance(complex complex1, complex complex2,
            string units, Func<complex, complex, string, string> func)
        {
            Console.Write(func(complex1, complex2, units));
            Console.WriteLine("\tCalculations finished...");
        }

        public static void Main()
        {
            //передаем функцию через делегат
            PrintCalculatedResistance((3.1f, 4.3f), (2.23f, -0.12f), "mA",
ParallelConnection);

            //передаем лямбда-функцию в качестве параметра
            PrintCalculatedResistance((23.17f, 0f), (0f, 0.002f), "A", (complex
complex1,
                complex complex2, string units) =>
            {
```

```
                return Complex.ComplexABS(Complex.ComplexSum(complex1,
complex2)).ToString() + "\t" + units;
            });

            Console.WriteLine("\n_________________________\n");

            var car = new ReflextionTest("Kia");
            Type t = car.GetType();

            Console.WriteLine("\nConstructors:");
            foreach (var x in t.GetConstructors())
            {
                Console.WriteLine(x);
            }

            Console.WriteLine("\nMethods:");
            foreach (var x in t.GetMethods())
            {
                Console.WriteLine(x);
            }

            Console.WriteLine("\nProperties:");
            foreach (var x in t.GetProperties())
            {
                Console.WriteLine(x);
            }

            Console.WriteLine("\nProperties with attribute:");
            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);
                }
            }

            object Result = t.InvokeMember("Description",
System.Reflection.BindingFlags.InvokeMethod, null, car, new object[] { });
            Console.WriteLine("\nDescription ->\n" + Result);
        }
    }
}
```

//Complex.cs

```
using System;

namespace Lab_6
{
    using complex = System.ValueTuple<double, double>;
    public static class Complex
    {
        public static complex ComplexSum(complex x, complex y)
        {
            return ((x.Item1 + y.Item1), (x.Item2 + y.Item2));
        }

        public static complex ComplexMultiply(complex x, complex y)
        {
```

```
            return ((x.Item1 * y.Item1 + x.Item2 * y.Item2), (x.Item1 * y.Item2 +
x.Item2 * y.Item1));
        }

        public static complex ComplexDivision(complex x, complex y)
        {
            var num = ComplexMultiply(x, (y.Item1, -y.Item2));
            double del = y.Item1 * y.Item1 + y.Item2 * y.Item2;
            return (num.Item1 / del, num.Item2 / del);
        }

        public static double ComplexABS(complex complex)
        {
            return Math.Sqrt(complex.Item1 * complex.Item1 + complex.Item2 *
complex.Item2);
        }
    }
}
```

//ReflextionTest.cs

```
using System;
namespace Lab_6
{
    public class ReflextionTest
    {
        public ReflextionTest() { }

        public ReflextionTest(string model)
        {
            Model = model;
        }

        public ReflextionTest(string color, string model)
        {
            Color = color;
            Model = model;
        }

        public ReflextionTest(string color, string model, int cost)
        {
            Color = color;
            Model = model;
            Cost = cost;
        }

        [NewAttribute("The color of a car")]
        public string Color { get; set; } = "";

        [NewAttribute("The model of a car")]
        public string Model { get; set; } = "";

        public int Cost { get; set; } = 0;

        public string Description()
        {
            string result;
            result = "Model: " + Model + "\nColor:" + Color + "\nCost: " + Cost +
"\n";
            return result;
```

```
        }

        public void ChangeColor(string color)
        {
            Color = color;
        }
    }

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

Анализ результатов

Терминал – Lab_6

```
1.20248738144474        mA      Calculations finished...
23.1700001626125        A       Calculations finished...


_____________________________


Constructors:
Void .ctor()
Void .ctor(System.String)
Void .ctor(System.String, System.String)
Void .ctor(System.String, System.String, Int32)

Methods:
System.String get_Color()
Void set_Color(System.String)
System.String get_Model()
Void set_Model(System.String)
Int32 get_Cost()
Void set_Cost(Int32)
System.String Description()
Void ChangeColor(System.String)
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
System.String ToString()

Properties:
System.String Color
System.String Model
Int32 Cost

Properties with attribute:
Color - The color of a car
Model - The model of a car

Description ->
Model: Mercedes
Color:
Cost: 0
```

Терминал – Lab_6

```
0.49740209102675        mA      Calculations finished...
23.1700001626125        A       Calculations finished...


_____________________________


Constructors:
Void .ctor()
Void .ctor(System.String)
Void .ctor(System.String, System.String)
Void .ctor(System.String, System.String, Int32)

Methods:
System.String get_Color()
Void set_Color(System.String)
System.String get_Model()
Void set_Model(System.String)
Int32 get_Cost()
Void set_Cost(Int32)
System.String Description()
Void ChangeColor(System.String)
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
System.String ToString()

Properties:
System.String Color
System.String Model
Int32 Cost

Properties with attribute:
Color - The color of a car
Model - The model of a car

Description ->
Model: Kia
Color:
Cost: 0
```