Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

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

Выполнил:
студент группы РТ5-31Б

Сысоев Александр

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

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.

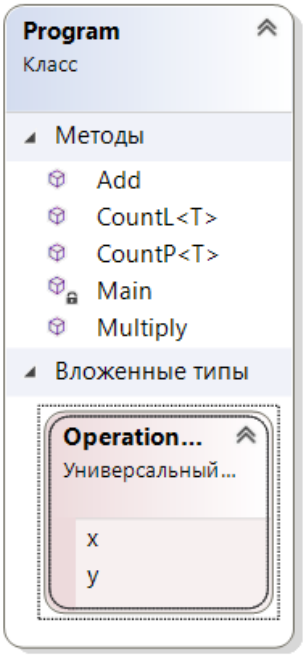
Часть 2. Разработать программу, реализующую работу с рефлексией.

(В качестве примера можно использовать проект «Reflection»).

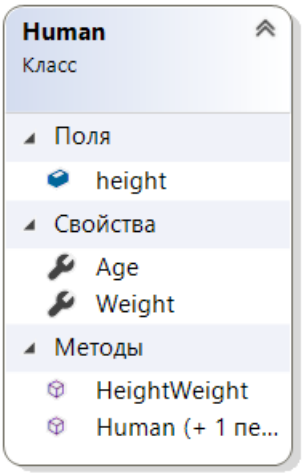
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс, содержащий конструкторы, свойства, методы.
3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
4. Создайте класс атрибута (унаследован от класса System.Attribute).
5. Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.
6. Вызовите один из методов класса с использованием рефлексии.

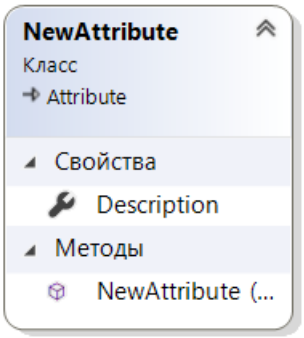
Диаграмма классов

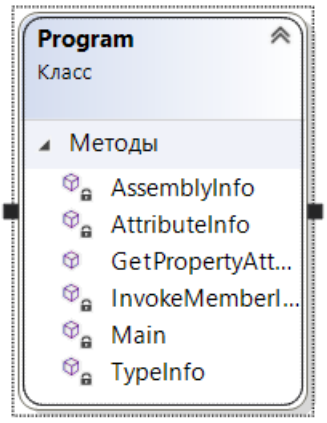
Часть 1



Часть 2







Текст программы

Часть 1

```
using System;

namespace Lab6_1
{
    class Program
    {
        public delegate T Operation<T>(T x, T y);

        public static int Add(int x, int y)
        {
            return x + y;
        }

        public static double Multiply(double x, double y)
        {
            return x * y;
        }

        public static T CountP<T>(T x, T y, Operation<T> operation)
        {
            return operation(x, y);
        }

        public static T CountL<T>(T x, T y, Func<T, T, T> operation)
        {
            return operation(x, y);
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Введите 4 числа:");
            int x = int.Parse(Console.ReadLine()), y = int.Parse(Console.ReadLine());
            double z = double.Parse(Console.ReadLine()), k =
double.Parse(Console.ReadLine());
            Console.WriteLine("Сумма = " + CountP(x, y, Add));
            Console.WriteLine("Сумма = " + CountL(x, y, Add));
            Console.WriteLine("Произведение = " + CountP(z, k, Multiply));
            Console.WriteLine("Произведение = " + CountP(z, k, Multiply));
            Console.ReadKey();
        }
    }
}
```

Часть 2

Файл Human.cs

```
namespace Reflection
{
    public class Human
    {
        public Human() { }

        public Human(double height, double weight)
        {
            this.height = height;
            Weight = weight;
        }

        public double HeightWeight(double height, double weight)
```

```
        {
            return weight / ((height / 100) * (height / 100));
        }

        [New("Возраст человека")]
        public int Age { get; set; }

        public double height;

        [New(Description = "Вес человека")]
        public double Weight { get; set; }
    }
}
```

Файл NewAttribute.cs

```
using System;

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; }
    }
}
```

Файл Program.cs

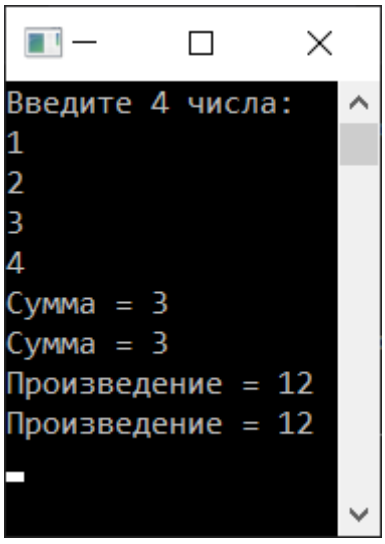
```
using System;
using System.Linq;
using System.Reflection;

namespace Reflection
{
    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;
        }

        static void AssemblyInfo()
        {
            Console.WriteLine("Вывод информации о сборке:");
            Assembly i = Assembly.GetExecutingAssembly();
            Console.WriteLine("Полное имя:" + i.FullName);
            Console.WriteLine("Исполняемый файл:" + i.Location);
        }

        static void TypeInfo()
```

```
        {
            Human obj = new Human();
            Type t = obj.GetType();
            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("\nForInspection реализует IComparable -> " +
            t.GetInterfaces().Contains(typeof(IComparable))
            );
        }

        static void InvokeMemberInfo()
        {
            Type t = typeof(Human);
            Console.WriteLine("\nВызов метода:");
            Human h = (Human)t.InvokeMember(null, BindingFlags.CreateInstance, null,
null, new object[] { });
            object[] parameters = new object[] { 180, 70 };
            object Result = t.InvokeMember("HeightWeight", BindingFlags.InvokeMethod,
null, h, parameters);
            Console.WriteLine("HeightWeight(180,70)={0}", Result);
        }

        static void AttributeInfo()
        {
            Type t = typeof(Human);
            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);
                }
            }
        }

        static void Main(string[] args)
        {
            AssemblyInfo();
            TypeInfo();
            InvokeMemberInfo();
```

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

Результат работы программы

Часть 1



Часть 2