

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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «БКИТ»

**Отчет по лабораторной работе №6**

Выполнил: студент группы ИУ5-31И Кареникс Артёмс

Москва, 2017 г

### **Задание:**

#### **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.

### **Код программы:**

#### **Box.cs**

```
using System;  
using System.Collections.Generic;  
using System.IO;
```

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

namespace LAB_6
{
    class Box
    {
        int width;
        int height;
        int lenght;

        public Box(int a, int b, int c)
        {
            width = a;
            height = b;
            lenght = c;
        }

        [AttributeClass("Details for width")]
        public int propertywidth
        {
            get { return width; }
            set { this.width = value; }
        }

        [AttributeClass("Details for height")]
        public int propertyheight
        {
            get { return height; }
            set { this.height = value; }
        }

        public int propertylenght
        {
            get { return lenght; }
            set { this.lenght = value; }
        }
    }
}
```

```

    }

    public int Volume()
    {
        return this.width * this.height * this.lenght;
    }

    public int AreaOfSurface()
    {
        return 2*(this.width * this.height + this.lenght* this.height+
this.width* this.lenght);
    }
}
}

```

### AttributeClass.cs

```

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

namespace LAB_6
{
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited
= false)]
    public class AttributeClass: Attribute
    {
        public AttributeClass(){}
        public AttributeClass(string str)
        {
            Description= str;
        }
        public string Description
        {

```

```

        set;

        get;
    }
}

}

Program.cs

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

namespace LAB_6
{

    class Program
    {
        delegate void AreaOrPerimeter(int a, int b, string units);

        static void Area(int width, int height, string units)
        {
            Console.WriteLine("Area of rectangle: " + width * height + " (" +
units + ")^2");
        }

        static void FindAreaOrPerimeter(int a, int b, string un,
AreaOrPerimeter p)
        {
            p(a, b, un);
        }

        static void FindAreaOrPerimeterAction(int a, int b, string un,
Action<int, int, string> f)

```

```

    {
        f(a, b, un);
    }

    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 Main(string[] args)
    {
        Console.WriteLine("Using of delegate");
        FindAreaOrPerimeter(2, 4, "m", Area);
        FindAreaOrPerimeter(2, 4, "m", (int a, int b, string un)=>{
Console.WriteLine("Perimeter of rectangle: "+((a+b)*2)+ ' ' + un); } );

        Console.WriteLine("\n\nUsing of action");
        FindAreaOrPerimeterAction(3, 8, "cm", Area);
        FindAreaOrPerimeterAction(3, 8, "cm", (int a, int b, string un) =>
{ Console.WriteLine("Perimeter of rectangle: " + ((a + b) * 2) + ' ' + un); });

        Type objType = typeof(Box);
        Console.WriteLine("\n\n\nInformation about class 'Box':\n");
    }
}

```

```

Console.WriteLine("Constructors:");
foreach ( var constr in objType.GetConstructors())
Console.WriteLine(constr);

Console.WriteLine("\nMethods:");
foreach (var meth in objType.GetMethods())
Console.WriteLine(meth);

Console.WriteLine("\nProperties:");
foreach (var prop in objType.GetProperties())
Console.WriteLine(prop);

Console.WriteLine("\nProperties with attribute:");
foreach (var prop in objType.GetProperties())
{
    object attrObj;
    if (GetPropertyAttribute(prop, typeof(AttributeClass), out
attrObj))
    {
        AttributeClass attr = attrObj as AttributeClass;
        Console.WriteLine(prop.Name + " - " + attr.Description);
    }
}

Console.WriteLine("\nRun method");
Console.WriteLine("Write method name:");
string methodName=Console.ReadLine();
Box box1 = new Box(1,2,3);
Type t = box1.GetType();
object[] parameters = new object[] { };
Console.WriteLine(t.InvokeMember(methodName,
BindingFlags.InvokeMethod, null, box1, parameters));

```

```

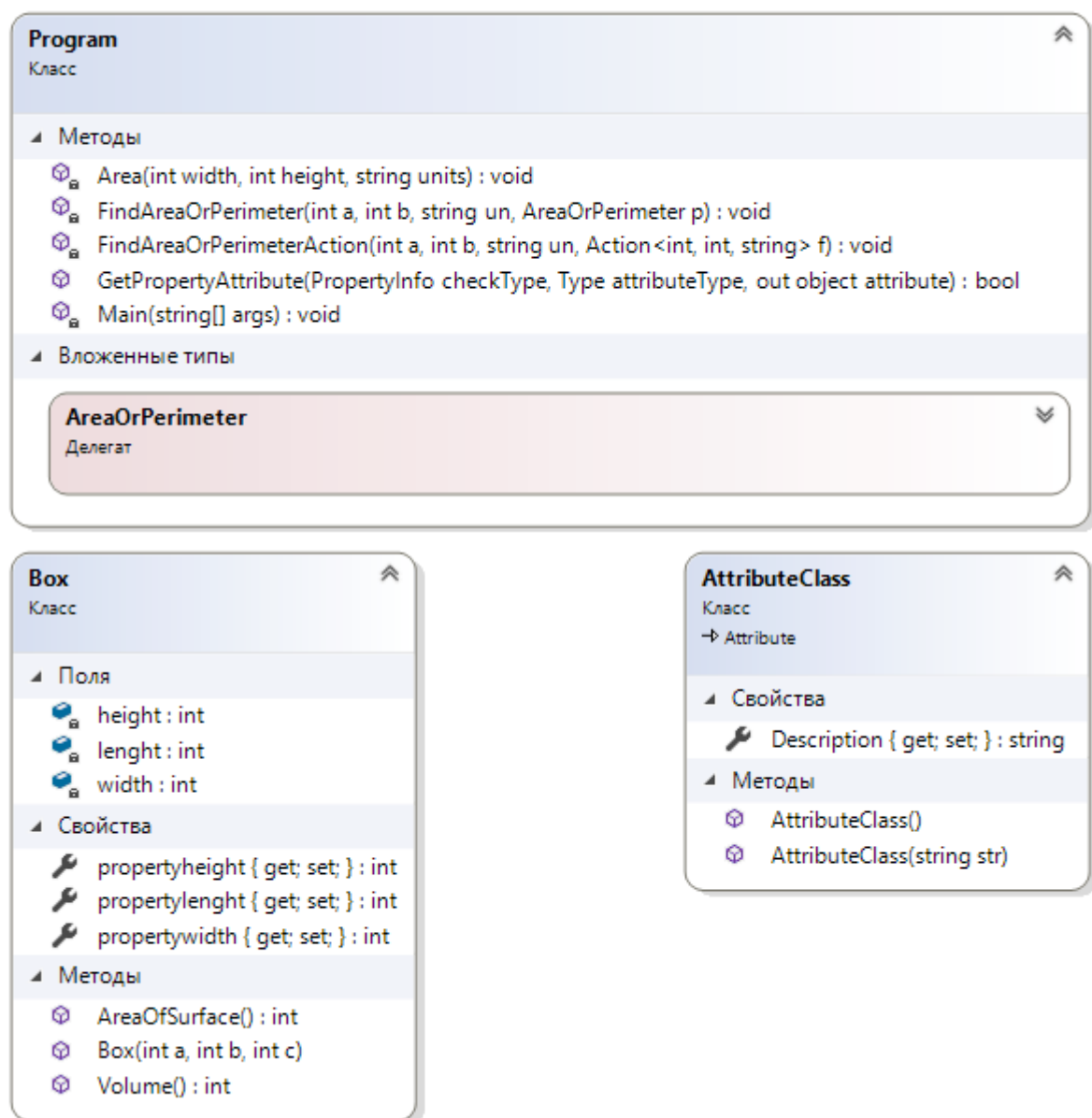
        Console.ReadLine();

    }

}

```

**Диаграмма классов:**





## Результаты



```
E:\Univer\BKIT\LAB_6\LAB_6\bin\Debug\LAB_6.exe
Using of delegate
Area of rectangle: 8 (m)^2
Perimeter of rectangle: 12 m

Using of action
Area of rectangle: 24 (cm)^2
Perimeter of rectangle: 22 cm

Information about class 'Box':
Constructors:
Void .ctor(Int32, Int32, Int32)
Methods:
Int32 get_propertywidth()
Void set_propertywidth(Int32)
Int32 get_propertyheight()
Void set_propertyheight(Int32)
Int32 get_propertylenght()
Void set_propertylenght(Int32)
Int32 Volume()
Int32 AreaOfSurface()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()
Properties:
Int32 propertywidth
Int32 propertyheight
Int32 propertylenght
Properties with attribute:
propertywidth - Details for width
propertyheight - Details for height
Run method
Write method name:
Volume
6
```