Отчёт по лабораторной работе № 6

Волгина А. Д., ИУ5-33Б

Часть 1. Разработать программу, использующую делегаты.

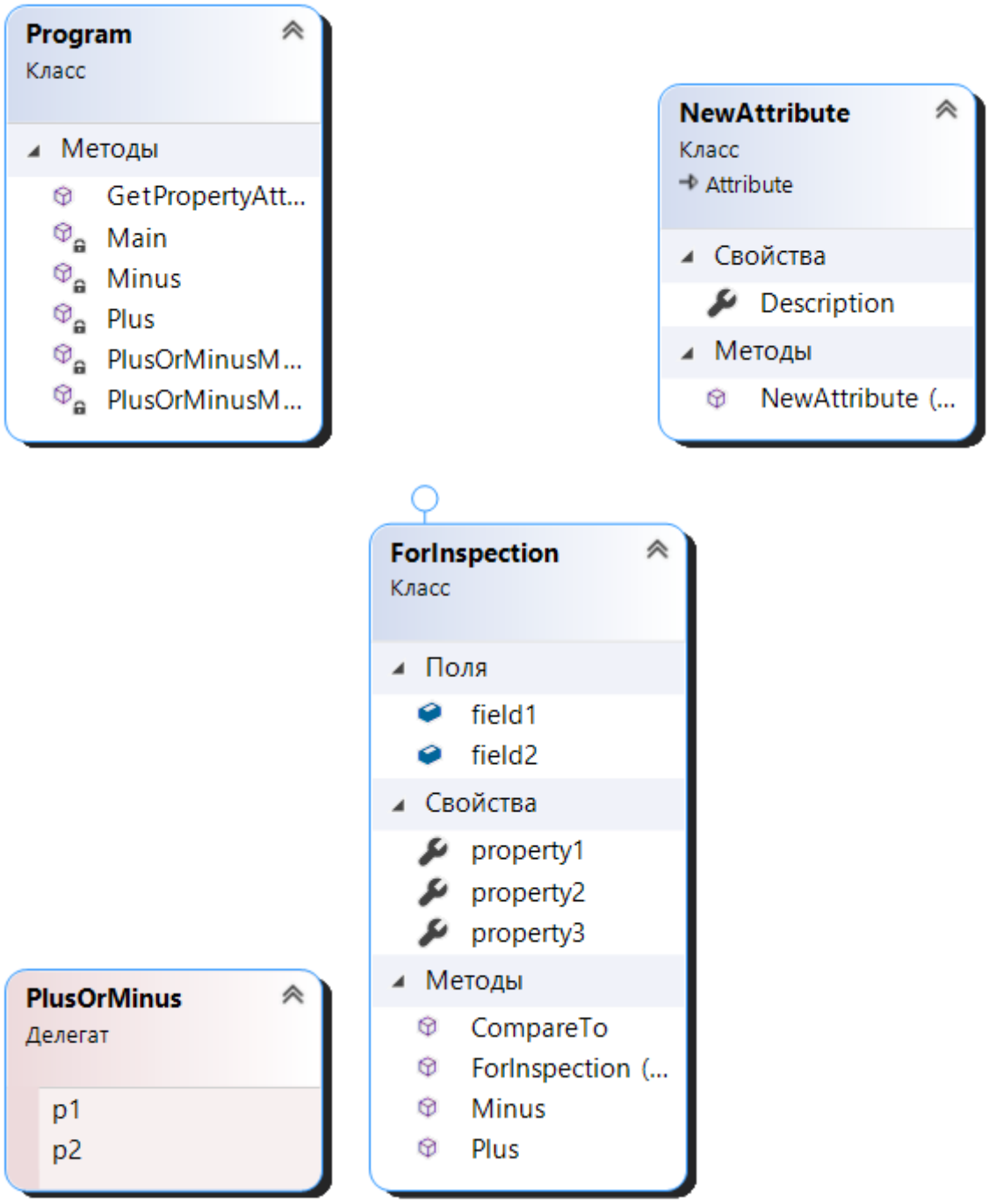
(В качестве примера можно использовать проект «Delegates»).

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.



Program.cs:

```
using System;
using System.Reflection;

namespace Lab6
{
    delegate void PlusOrMinus(int p1, int p2);
```

```
namespace Reflection
{
    public class ForInspection : IComparable
    {
        public int field1;
        public float field2;
        public ForInspection() { }
        public ForInspection(int i) { }
        public ForInspection(string s) { }
        [NewAttribute("Описание для property1")]
        public string property1 { get; set; }
        public int property2 { get; set; }
        [NewAttribute(Description = "Описание для property3")]
        public double property3 { get; set; }
        public int Plus(int x, int y) { return x + y; }
        public int Minus(int x, int y) { return x - y; }
        public int CompareTo(object obj)
        {
            return 0;
        }
    }
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)]
    public class NewAttribute : Attribute
    {
        public NewAttribute() { }
```

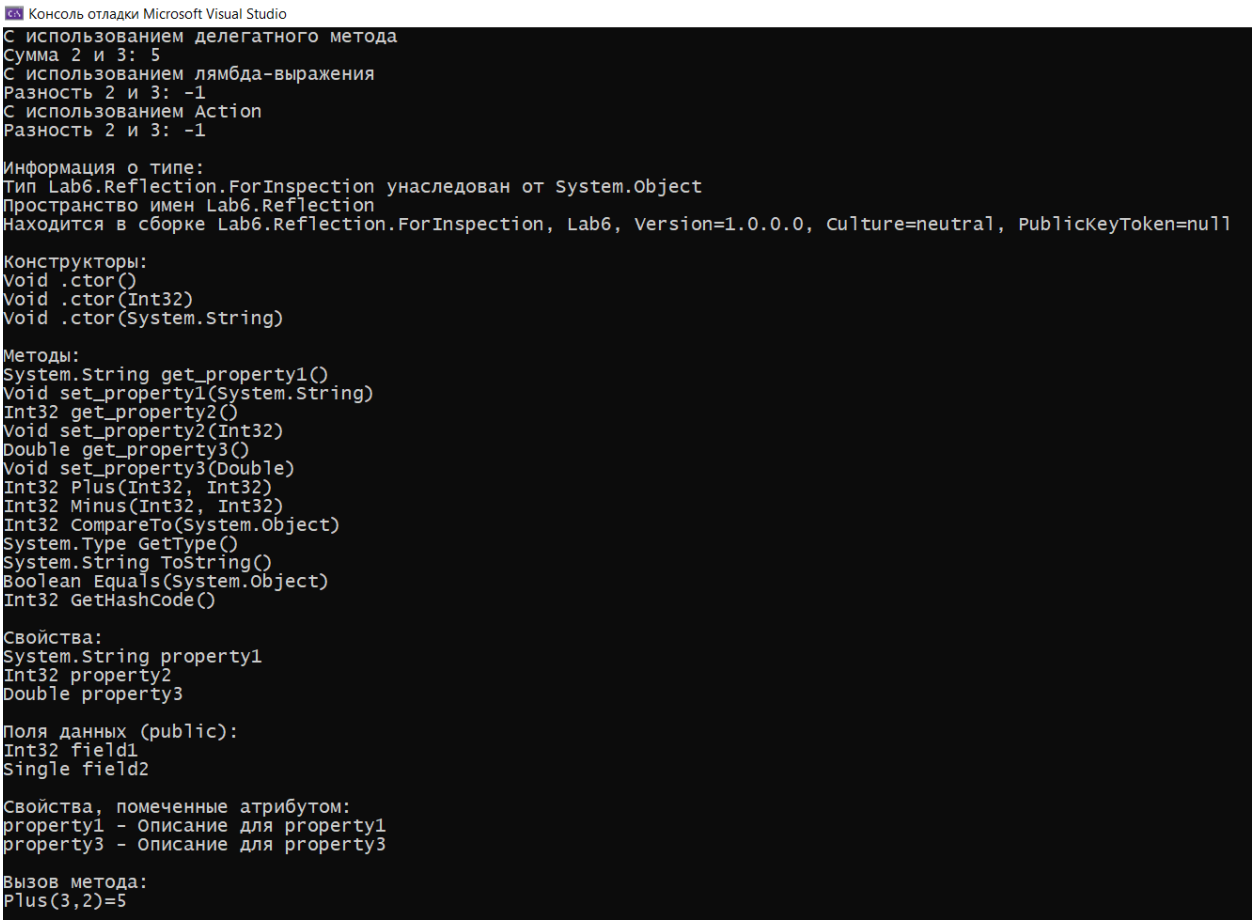
```
        public NewAttribute(string DescriptionParam)
        {
            Description = DescriptionParam;
        }
        public string Description { get; set; }
    }
}
class Program
{
    public static bool GetPropertyAttribute(PropertyInfo checkType, Type attributeType, out object attribute)
    {
        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 Plus(int p1, int p2) { Console.WriteLine("Сумма {0} и {1}: {2}", p1, p2, p1 + p2); }
```

```
static void Minus(int p1, int p2) { Console.WriteLine("Разность {0} и {1}: {2}", p1, p2, p1 - p2); }

static void PlusOrMinusMethod(int i1, int i2, PlusOrMinus PlusOrMinusParam)

{

   PlusOrMinusParam(i1, i2);

}

static void PlusOrMinusMethodAct(int i1, int i2, Action<int, int> PlusOrMinusParam)

{

   PlusOrMinusParam(i1, i2);

}

static void Main(string[] args)

{

   PlusOrMinus PlusOrMinusParam = (x, y) => { Console.WriteLine("Разность {0} и {1}: {2}", x, y, x - y); };

   int i1 = 2, i2 = 3;

   Console.WriteLine("С использованием делегатного метода");

   PlusOrMinusMethod(i1, i2, Plus);

   Console.WriteLine("С использованием лямбда-выражения");

   PlusOrMinusMethod(i1, i2, (x, y) => { Console.WriteLine("Разность {0} и {1}: {2}", x, y, x - y); });

   Console.WriteLine("С использованием Action");

   PlusOrMinusMethodAct(i1, i2, Minus);

   Reflection.ForInspection obj = new Reflection.ForInspection();

   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("\nСвойства, помеченные атрибутом:");
        foreach (var x in t.GetProperties())
```

```
        {
            object attrObj;
            if (GetPropertyAttribute(x, typeof(Reflection.NewAttribute), out attrObj))
            {
                Reflection.NewAttribute attr = attrObj as Reflection.NewAttribute;
                Console.WriteLine(x.Name + " - " + attr.Description);
            }
        }
        Console.WriteLine("\nВызов метода:");
        //Создание объекта
        //ForInspection fi = new ForInspection();
        //Можно создать объект через рефлексию
        Reflection.ForInspection fi =
         (Reflection.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);
    }
}
```

}


Консоль отладки Microsoft Visual Studio
С использованием делегатного метода
Сумма 2 и 3: 5
С использованием лямбда-выражения
Разность 2 и 3: -1
С использованием Action
Разность 2 и 3: -1
Информация о типе:
Тип Lab6.Reflection.ForInspection унаследован от System.Object
Пространство имен Lab6.Reflection
Находится в сборке Lab6.Reflection.ForInspection, Lab6, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Конструкторы:
Void .ctor()
Void .ctor(Int32)
Void .ctor(System.String)
Методы:
System.String get_property1()
Void set_property1(System.String)
Int32 get_property2()
Void set_property2(Int32)
Double get_property3()
Void set_property3(Double)
Int32 Plus(Int32, Int32)
Int32 Minus(Int32, Int32)
Int32 CompareTo(System.Object)
System.Type GetType()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()
Свойства:
System.String property1
Int32 property2
Double property3
Поля данных (public):
Int32 field1
Single field2
Свойства, помеченные атрибутом:
property1 - Описание для property1
property3 - Описание для property3
Вызов метода:
Plus(3,2)=5