Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Базовые компоненты интернет технологий»

Отчет по лабораторной работе № 6

«Использование делегатов и работа с рефлексией»

Выполнил:
студент группы ИУ5-33
Терентьев Владислав

Москва, 2019 г.

1. Постановка задачи

Часть 1. Разработать программу, использующую делегаты.

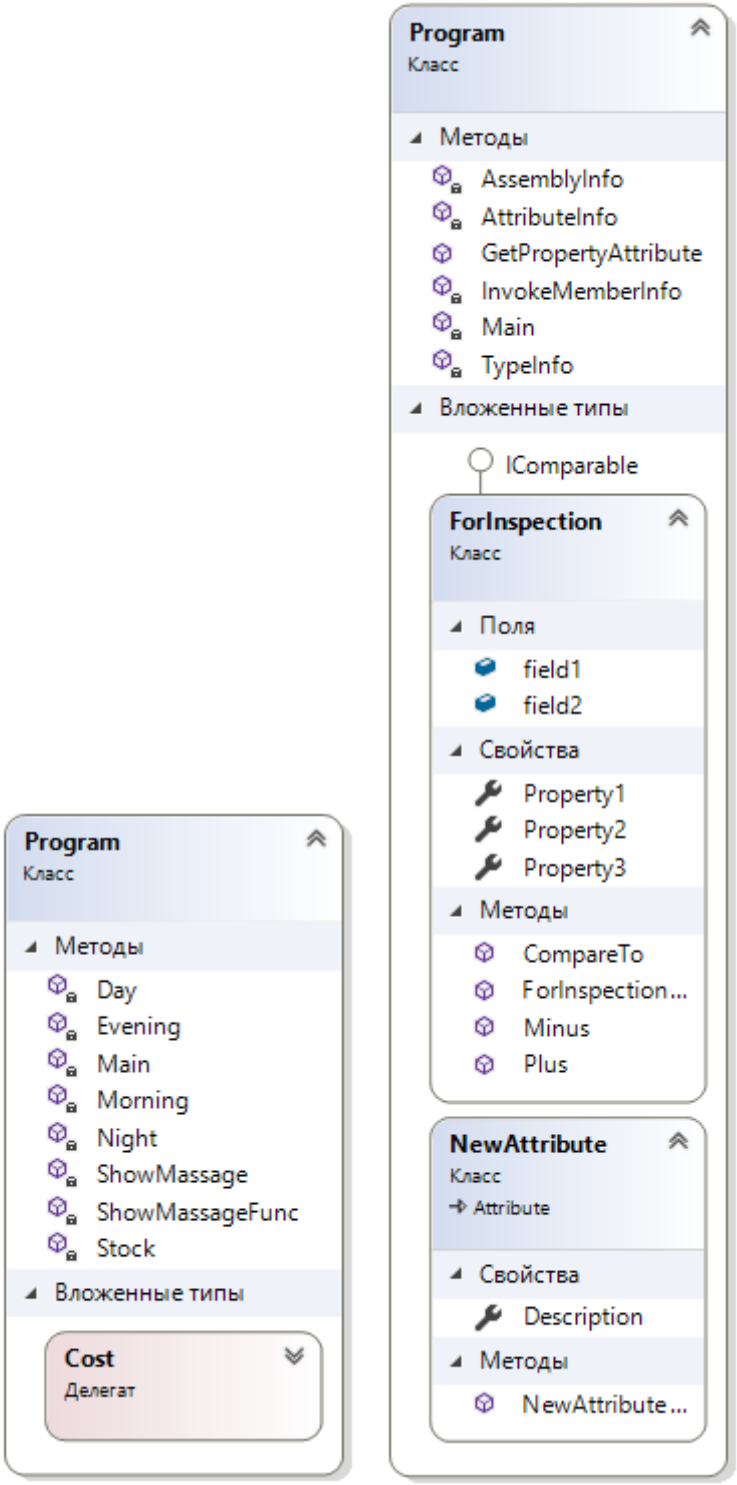
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат, в качестве одного из входным параметров. Осуществите вызов метода, передавая в качестве параметра-делегата:
    a. метод, разработанный в пункте 3;
    b. 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.

2. Диаграммы классов



3. Текст программы

3.1. Часть 1

```
using System;

namespace laba6
{

    class Program
```

```
    {
        delegate string Cost(string name, int basecost);

        private static string Day(string name, int basecost)
        {
            return "Стоимость билета на кино '" + name + "' в данный момент равняется " +
basecost * 1.1 + " рублей.";
        }

        private static string Evening(string name, int basecost)
        {
            return "Стоимость билета на кино '" + name + "' в данный момент равняется " +
basecost * 1.5 + " рублей.";
        }

        private static string Night(string name, int basecost)
        {
            return "Стоимость билета на кино '" + name + "' в данный момент равняется " +
basecost * 0.8 + " рублей.";
        }

        private static string Morning(string name, int basecost)
        {
            return "Стоимость билета на кино '" + name + "' в данный момент равняется " +
basecost * 1.0 + " рублей.";
        }

        private static string Stock(string name, int basecost)
        {
            return "Стоимость билета на кино '" + name + "' по скидочному купону
равняется " + basecost * 0.5 + " рублей.";
        }

        private static void ShowMassage(Cost CostParam, string NameParam, int
BasecostParam)
        {
            Console.WriteLine(CostParam(NameParam, BasecostParam));
        }

        private static void ShowMassageFunc(Func<string, int, string> CostParam, string
NameParam, int BasecostParam)
        {
            Console.WriteLine(CostParam(NameParam, BasecostParam));
        }

        static void Main(string[] args)
        {
            Console.Title = "Терентьев Владислав ИУ5-33";
            Cost cst;
            if (DateTime.Now.Hour < 6) { cst = Night; }
            else
            {
                if (DateTime.Now.Hour < 10) { cst = Morning; }
                else
                {
                    if (DateTime.Now.Hour < 16) { cst = Day; }
                    else { cst = Evening; }
                }
            }
            ShowMassage(cst, "Матрица 4", 200);
            ShowMassage((name, basecost) => "Стоимость детского билета на кино '" + name
+ "' равняется " + basecost * 0.5 + " рублей.", "Матрица 4", 200);
            ShowMassageFunc(Stock, "Матрица 4", 200);
            Console.ReadKey();
        }
```

```
    }
}
```

3.2. Часть 2

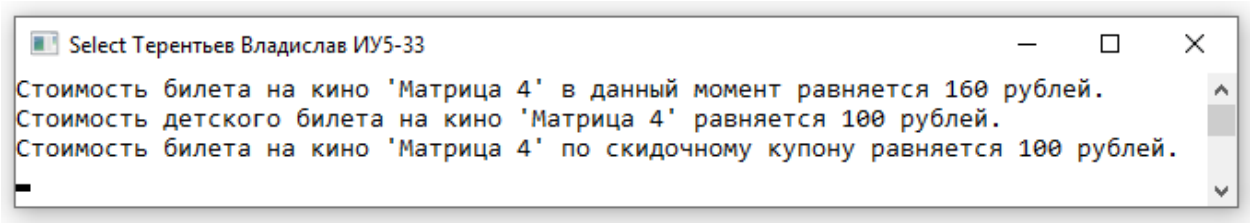
```
using System;
using System.Linq;
using System.Reflection;

namespace laba6_part2
{
    class Program
    {
        [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited =
false)]
        public class NewAttribute : Attribute
        {
            public NewAttribute() { }
            public NewAttribute(string DescriptionParam)
            {
                Description = DescriptionParam;
            }
            public string Description { get; set; }
        }

        public class ForInspection : IComparable
        {
            public ForInspection() { }
            public ForInspection(int i) { }
            public ForInspection(string str) { }
            public int Plus(int x, int y) { return x + y; }
            public int Minus(int x, int y) { return x - y; }

            [New("Описание для Property1")]
            public string Property1 { get; set; }

            public int Property2 { get; set; }

            [New(Description = "Описание для Property3")]
            public double Property3 { get; private set; }

            public int field1;
            public float field2;
            public int CompareTo(object obj)
            {
                return 0;
            }
        }

        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()
        {
            Type t = typeof(ForInspection);
            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(ForInspection);
            Console.WriteLine("\nВызов метода:");
            ForInspection fi = (ForInspection)t.InvokeMember(null,
BindingFlags.CreateInstance, null, null, new object[] { });
            object[] parameters = new object[] { 3, 2 };
            object Result = t.InvokeMember("Plus", BindingFlags.InvokeMethod, null, fi,
parameters);
            Console.WriteLine("Plus(3,2)={0}", Result);
        }

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

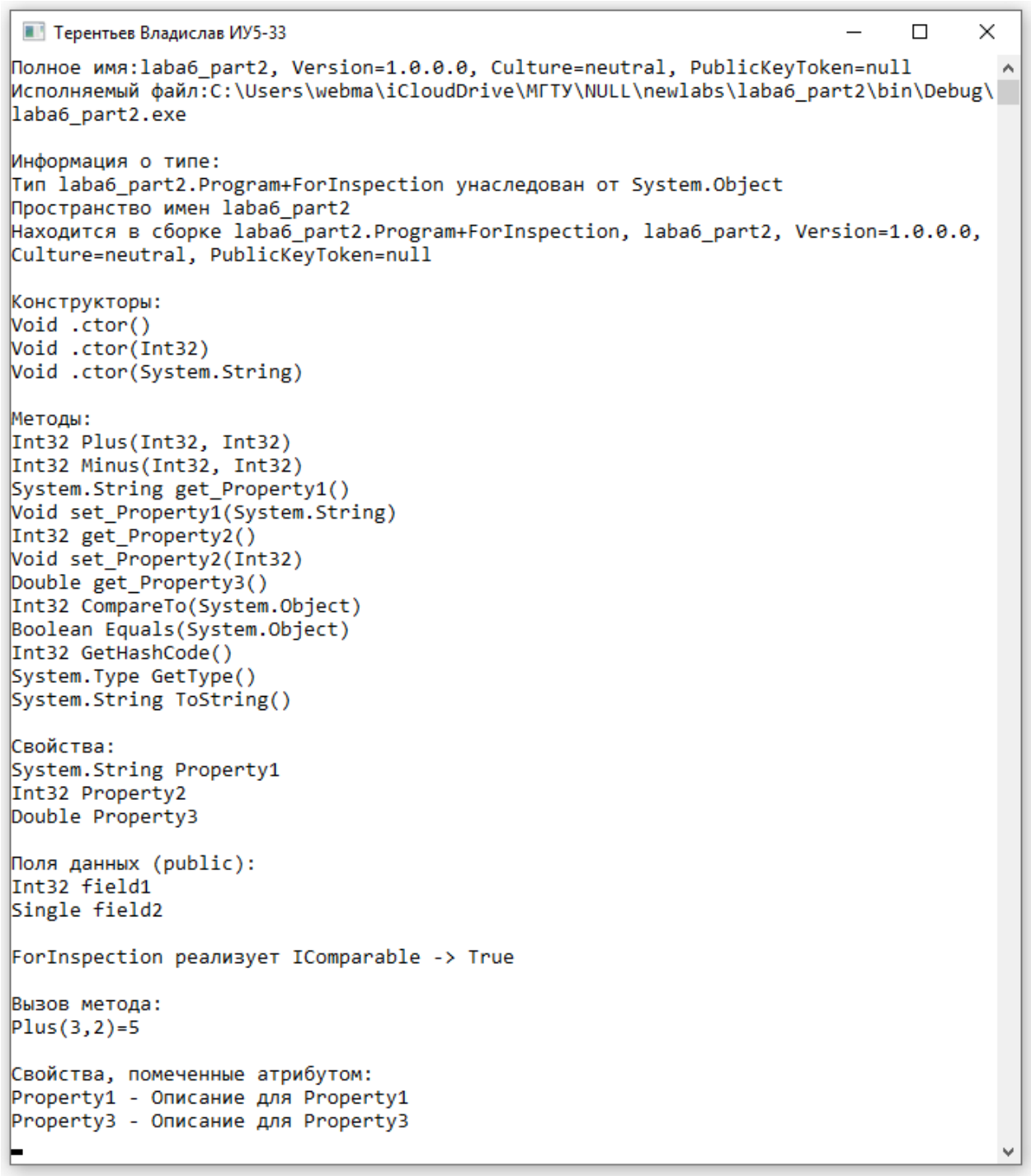
```
        }

        static void Main(string[] args)
        {
            Console.Title = "Терентьев Владислав ИУ5-33";
            AssemblyInfo();
            TypeInfo();
            InvokeMemberInfo();
            AttributeInfo();
            Console.ReadLine();
        }
    }
}
```

4. Анализ результатов

4.1. Часть 1



4.2. Часть 2



5. Ссылка на репозиторий

https://github.com/iYroglif/newlabs