Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

Выполнил:

студент группы ИУ5-31

Головацкий Андрей

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2020 г.

Описание задания:

Часть 1.

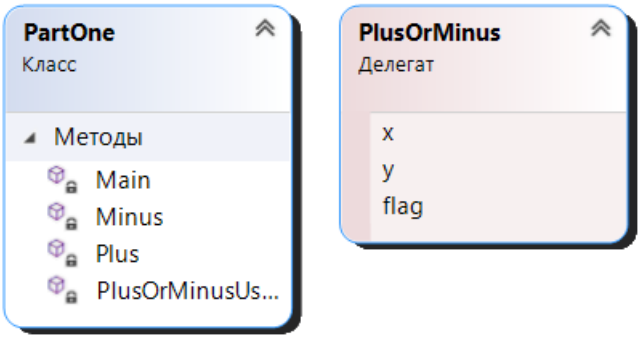
1) Программа должна быть разработана в виде консольного приложения на языке C#.
2) Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3) Напишите метод, соответствующий данному делегату.
4) Напишите метод, принимающий разработанный Вами делегат, 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.

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.

Часть 1:

Диаграмма классов:



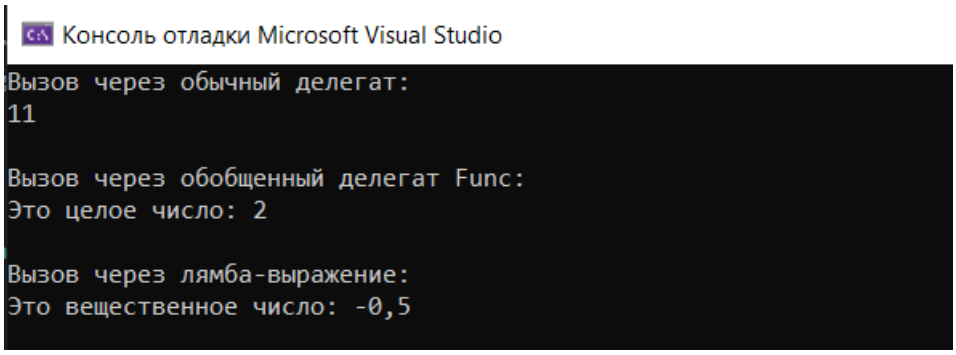
Текст программы:

1) Program.cs:

```
1. using System;
2. using System.Collections.Generic;
3. using System.Net.Cache;
4. using System.Runtime.CompilerServices;
5. using System.Security.Cryptography.X509Certificates;
6. using System.Reflection;
7.
8. namespace Lab6
9. {
10.         delegate object PlusOrMinus(int x, double y, bool flag);
11.         class PartOne
12.         {
13.             static object Plus(int x, double y, bool flag)
14.             {
15.                 if (flag == true)
16.                     return ((int)(x + y));
17.                 else
18.                     return (x + y);
19.             }
20.             static object Minus(int x, double y, bool flag)
21.             {
22.                 if (flag == true)
23.                     return ((int)(x - y));
24.                 else
25.                     return (x - y);
```

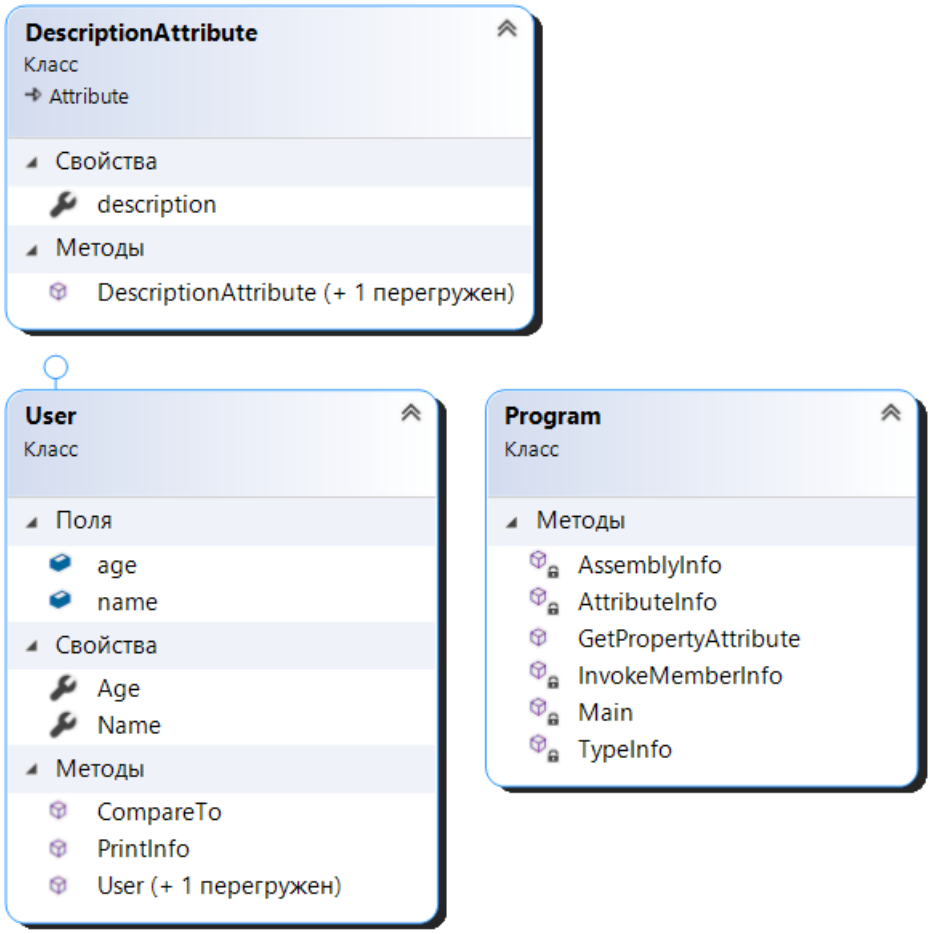
```
26.             }
27.             static void PlusOrMinusUse1(string str, int x, double y,
    bool flag, Func<int, double, bool, object> PlusOrMinusParam)
28.             {
29.                 object result = PlusOrMinusParam(x, y, flag);
30.                 Console.WriteLine(str + " " + result.ToString());
31.             }
32.
33.             static void Main(string[] args)
34.             {
35.                 PlusOrMinus pm = new PlusOrMinus(Plus);
36.                 Console.WriteLine("Вызов через обычный делегат: ");
37.                 Console.WriteLine(pm(1, 10, true).ToString() + "\n");
38.                 Console.WriteLine("Вызов через обобщенный делегат
    Func:");
39.                 PlusOrMinusUse1("Это целое число:", 1, 1.5, true,
    Plus);
40.                 Console.WriteLine("\nВызов через лямба-выражение:");
41.                 PlusOrMinusUse1("Это вещественное число:", 1, 1.5,
    false, (x, y, flag) =>
42.                 {
43.                     if (flag == true)
44.                         return ((int)(x - y));
45.                     else
46.                         return (x - y);
47.                 }
48.                 );
49.             }
50.
51.         }
52.     }
```

Пример выполнения программы:



Часть 2:

Диаграмма классов:



Текст программы:

1) Program.cs:

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Reflection;
5. using System.Text;
6. using System.Threading.Tasks;
7.
8. namespace Lab6Part2
9. {
10.          class Program
11.          {
12.              /// <summary>
13.              /// Проверка, что у свойства есть атрибут заданного типа
14.              /// </summary>
15.              /// <returns>Значение атрибута</returns>
16.              public static bool GetPropertyAttribute(PropertyInfo
   checkType, Type attributeType, out object attribute)
17.              {
18.                  bool Result = false;
19.                  attribute = null;
20.
21.                  //Поиск атрибутов с заданным типом
22.                  var isAttribute =
   checkType.GetCustomAttributes(attributeType, false);
23.                  if (isAttribute.Length > 0)
24.                  {
25.                      Result = true;
26.                      attribute = isAttribute[0];
27.                  }
28.
29.                  return Result;
30.              }
31.              /// <summary>
32.              /// Получение информации о текущей сборке
33.              /// </summary>
34.              static void AssemblyInfo()
35.              {
```

```
36.                     Console.WriteLine("Вывод информации о сборке:");
37.                     Assembly i = Assembly.GetExecutingAssembly();
38.                     Console.WriteLine("Полное имя:" + i.FullName);
39.                     Console.WriteLine("Исполняемый файл:" + i.Location);
40.                 }
41.                 /// <summary>
42.                 /// Получение информации о типе
43.                 /// </summary>
44.                 static void TypeInfo()
45.                 {
46.                     Type t = typeof(User);
47.
48.                     Console.WriteLine("\nИнформация о типе:");
49.                     Console.WriteLine("Тип " + t.FullName + " унаследован
   от " + t.BaseType.FullName);
50.                     Console.WriteLine("Пространство имен " + t.Namespace);
51.                     Console.WriteLine("Находится в сборке " +
   t.AssemblyQualifiedName);
52.
53.                     Console.WriteLine("\nКонструкторы:");
54.                     foreach (var x in t.GetConstructors())
55.                     {
56.                         Console.WriteLine(x);
57.                     }
58.
59.                     Console.WriteLine("\nМетоды:");
60.                     foreach (var x in t.GetMethods())
61.                     {
62.                         Console.WriteLine(x);
63.                     }
64.
65.                     Console.WriteLine("\nСвойства:");
66.                     foreach (var x in t.GetProperties())
67.                     {
68.                         Console.WriteLine(x);
69.                     }
70.
71.                     Console.WriteLine("\nПоля данных (public):");
72.                     foreach (var x in t.GetFields())
73.                     {
```

```
74.                     Console.WriteLine(x);
75.                 }
76.
77.                 Console.WriteLine("\nForInspection реализует
   IComparable -> " +
78.                 t.GetInterfaces().Contains(typeof(IComparable))
79.                 );
80.             }
81.
82.             /// <summary>
83.             /// Использования метода InvokeMember
84.             /// </summary>
85.             static void InvokeMemberInfo()
86.             {
87.                 Type t = typeof(User);
88.                 Console.WriteLine("\nВызов метода:");
89.
90.                 //Создание объекта
91.                 User fi = new User("Andrey", 19);
92.                 //User fi = (User)t.InvokeMember(null,
   BindingFlags.CreateInstance, null, null, new object[] { });
93.
94.                 //Вызов метода
95.                 t.InvokeMember("PrintInfo", BindingFlags.InvokeMethod,
   null, fi, new object[] { });
96.             }
97.
98.             /// <summary>
99.             /// Работа с атрибутами
100.            /// </summary>
101.            static void AttributeInfo()
102.            {
103.                Type t = typeof(User);
104.                Console.WriteLine("\nСвойства, помеченные
   атрибутом:");
105.                foreach (var x in t.GetProperties())
106.                {
107.                    object attrObj;
108.                    if (GetPropertyAttribute(x,
   typeof(DescriptionAttribute), out attrObj))
```

```
                    {
                        DescriptionAttribute attr = attrObj as
   DescriptionAttribute;
                        Console.WriteLine(x.Name + " - " +
   attr.description);
                    }
                }
            }
            static void Main(string[] args)
            {
                AssemblyInfo();
                TypeInfo();
                InvokeMemberInfo();
                AttributeInfo();
            }
        }
    }
```

2) ForInspection.cs:

```
1. using System;
2. using System.Collections.Generic;
3. using System.Net.Cache;
4. using System.Runtime.CompilerServices;
5. using System.Security.Cryptography.X509Certificates;
6. using System.Reflection;
7. using System.Text;
8. using System.Threading.Tasks;
9.
10.     namespace Lab6Part2
11.     {
12.         class DescriptionAttribute : Attribute
13.         {
14.             public string description { get; set; }
15.             public DescriptionAttribute() { }
16.             public DescriptionAttribute(string _description) {
   description = _description; }
17.         }
18.
19.         class User : IComparable
```

```
20.         {
21.             public User() { }
22.             public User(string _name, int _age) { name = _name; age =
    _age; }
23.             public string name;
24.             public string Name
25.             {
26.                 get { return name; }
27.                 set { name = value; }
28.             }
29.             public int age;
30.             [Description("Возраст")]
31.             public int Age
32.             {
33.                 get {return age; }
34.                 set { age = value; }
35.             }
36.             public void PrintInfo()
37.             {
38.                 Console.WriteLine("Имя пользователя: " + name +
    "\nВозраст пользователя: " + age.ToString());
39.             }
40.             public int CompareTo(object obj)
41.             {
42.                 if (obj.GetType().Name == "User")
43.                 {
44.                     Type t = obj.GetType();
45.                     foreach (var field in t.GetFields())
46.                     {
47.                         if (field.GetValue(obj) !=
    field.GetValue(this))
48.                             return (1);
49.                     }
50.                     return (0);
51.                 }
52.                 return (1);
53.             }
54.         }
55.     }
```

Пример выполнения программы:

```
Вывод информации о сборке:
Полное имя:Lab6Part2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Исполняемый файл:D:\Visual Studio\C#Labs\Lab6Part2\bin\Debug\netcoreapp3.1\Lab6Part2.dll

Информация о типе:
Тип Lab6Part2.User унаследован от System.Object
Пространство имен Lab6Part2
Находится в сборке Lab6Part2.User, Lab6Part2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

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

Методы:
System.String get_Name()
Void set_Name(System.String)
Int32 get_Age()
Void set_Age(Int32)
Void PrintInfo()
Int32 CompareTo(System.Object)
System.Type GetType()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()

Свойства:
System.String Name
Int32 Age

Поля данных (public):
System.String name
Int32 age

ForInspection реализует IComparable -> True

Вызов метода:
Имя пользователя: Andrey
Возраст пользователя: 19

Свойства, помеченные атрибутом:
Age - Возраст
```