**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Ветошкин Артём |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Постановка задачи**

**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.

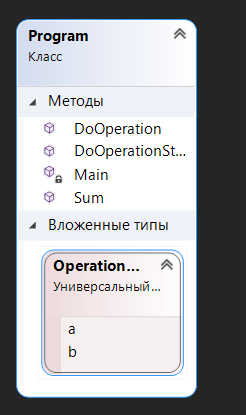
**Часть 2. Разработать программу, реализующую работу с рефлексией.**

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

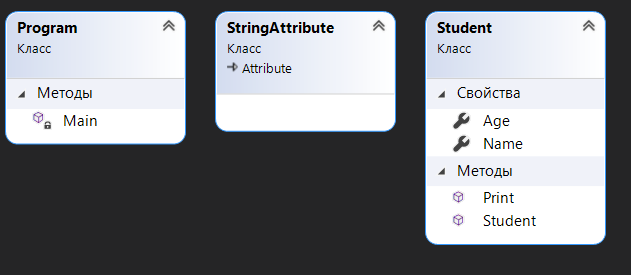
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс, содержащий конструкторы, свойства, методы.
3. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
4. Создайте класс атрибута (унаследован от класса System.Attribute).
5.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

**Диаграмма классов**

Delegates:



Reflection:



**Текст программы**

Delegates:

Файл Program.cs:

using System;

namespace IT6.\_1 {

class Program {

public delegate T Operation<T> (T a, T b);

static public int Sum(int a, int b) {

return a + b;

}

static public int DoOperation(int a, int b, Operation<int> operation) {

return operation(a, b);

}

static public int DoOperationStandart(int a, int b, Func<int, int, int> operation) {

return operation(a, b);

}

static void Main(string[] args) {

int a = 1;

int b = 30;

Console.WriteLine("Мой делегат: Делегат: " + DoOperation(a, b, Sum).ToString() + "\n" +

"Лямбда: " + DoOperation(a, b, (int a, int b) => a + b).ToString());

Console.WriteLine("Func: Делегат: " + DoOperationStandart(a, b, Sum).ToString() + "\n" +

"Лямбда: " + DoOperationStandart(a, b, (int a, int b) => a + b).ToString());

}

}

}

Reflection:

Файл Program.cs:

using System;

using System.Reflection;

namespace IT6.\_2 {

public class StringAttribute: Attribute {}

class Student {

public Student(string name, int age) {

Name = name;

Age = age;

}

[String]

public string Name { get; set; } = "";

public int Age { get; set; } = 0;

public void Print() {

Console.WriteLine("Имя студента: " + Name + ", возраст студента: " + Age);

}

}

class Program {

static void Main(string[] args) {

Type student = Type.GetType("IT6.\_2.Student");

Console.WriteLine("Конструкторы:");

foreach(var constructor in student.GetConstructors()) {

Console.WriteLine(constructor.Name);

}

Console.WriteLine("\nПоля:");

foreach (var constructor in student.GetFields(/\*BindingFlags.NonPublic | BindingFlags.Instance\*/)) {

Console.WriteLine(constructor.Name);

}

Console.WriteLine("\nМетоды");

foreach (var constructor in student.GetMethods()) {

Console.WriteLine(constructor.Name);

}

Console.WriteLine("\nСвойства");

foreach (var constructor in student.GetProperties()) {

Console.WriteLine(constructor.Name);

}

Console.WriteLine("\nЧлены класса с атрибутом String: ");

foreach (var member in student.GetMembers()) {

foreach (var attribute in member.GetCustomAttributes()) {

if (attribute is StringAttribute) {

Console.WriteLine(member.Name);

}

}

}

Console.WriteLine("\n");

MethodInfo method = student.GetMethod("Print", new Type[] {});

Student student\_class = new Student ( "Ivan", 25 );

method.Invoke(student\_class, new Object[] { });

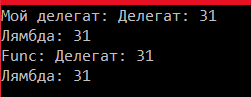
}

}

}

**Анализ результатов**

Delegates:



Reflection:

