**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: |  | Проверил: |
| студент группы ИУ5-35б |  | преподаватель каф. ИУ5 |
| Зубарева Антонина |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

# Задание.

**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.

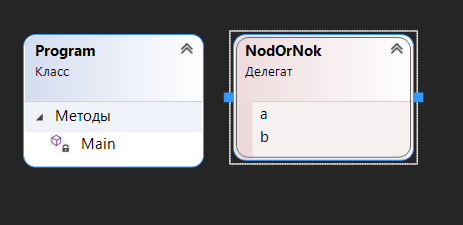
**Часть 2. Разработать программу, реализующую работу с рефлексией.**

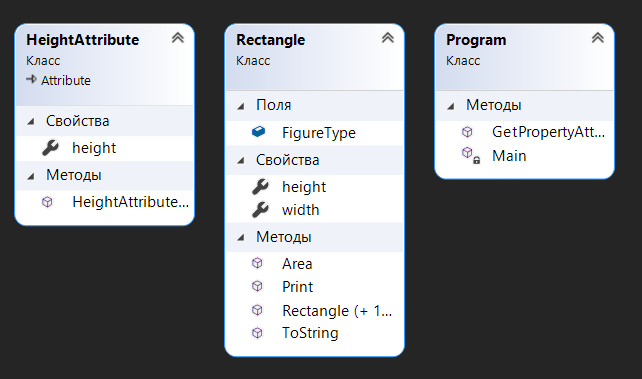
(В качестве примера можно использовать проект «Reflection»).

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.

Вызовите один из методов класса с использованием рефлексии

# Диаграмма классов





# Текст программы

# using System;

# namespace lab6

# {

# delegate int NodOrNok(int a, int b);

# class Program

# {

# static void Main(string[] args)

# {

# Console.WriteLine("Пример работы делегатов\n");

# 

# static int NOD(int a, int b)

# {

# int res = 0;

# for (int i = a; i > 0; i--)

# {

# if (a % i == 0 && b % i == 0)

# {

# res = i;

# break;

# }

# }

# return res;

# }

# 

# static int NOK(int a, int b)

# {

# int res = 0;

# res = a \* b / NOD(a, b);

# return res;

# }

# static void NodOrNokMethod(int a, int b, NodOrNok nonParam)

# {

# Console.WriteLine(nonParam(a, b));

# }

# //вызов метода с использованием метода, соответствующего делегату

# NodOrNokMethod(8, 6, NOD);

# //вызов метода с использованием лямбда-выражения

# NodOrNokMethod(8, 6, (int x, int y) => { return NOD(x, y); });

# //использование обобщённого делегата

# void NodOrNokMethodFunc(int a, int b, Func<int, int, int> nonParam)

# {

# Console.WriteLine(nonParam(a, b));

# }

# NodOrNokMethodFunc(6, 8, NOD);

# }

# }

# }

# using System;

# using System.Reflection;

# namespace lab6Reflection

# {

# class Program

# {

# static void Main(string[] args)

# {

# Console.WriteLine("Вывод информации о конструкторах, свойствах и методах\nс помощью рефлексии");

# Rectangle obj = new Rectangle(2, 3);

# Type t = obj.GetType();

# //другой способ

# //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("Вывод свойств с атрибутами");

# foreach (var x in t.GetProperties())

# {

# object attrObj;

# if (GetPropertyAttribute(x, typeof(HeightAttribute), out attrObj))

# {

# HeightAttribute attr = attrObj as HeightAttribute;

# Console.WriteLine(x);

# }

# }

# Console.WriteLine("Вызов метода класса с помощью рефлексии");

# // Параметры вызова метода

# object[] parameters = new object[] {};

# //Вызов метода

# object Result = t.InvokeMember("Area", BindingFlags.InvokeMethod, null, obj, parameters);

# Console.WriteLine("Area()={0}", Result);

# }

# ///<summary>///

# ///Проверка, что у свойства есть атрибут заданного типа///

# ///</summary>///

# ///<returns>Значение атрибута</returns>

# 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;

# }

# }

# public class HeightAttribute : System.Attribute

# {

# public int height { get; set; }

# public HeightAttribute()

# { }

# public HeightAttribute(int h)

# {

# this.height = h;

# }

# }

# /// <summary>

# /// Класс прямоугольник

# /// </summary>

# 

# class Rectangle

# {

# public double width { get; set; }

# [HeightAttribute(0)]

# public double height { get; set; }

# public string FigureType;

# public Rectangle()

# {

# this.width = 0;

# this.height = 0;

# this.FigureType = "Прямоугольник";

# }

# /// <summary>

# /// Конструктор класса прямоугольник, инициализирует сначала ширину, потом высоту

# /// </summary>

# public Rectangle(double w, double h)

# {

# this.width = w;

# this.height = h;

# this.FigureType = "Прямоугольник";

# }

# public double Area()

# {

# return this.width \* this.height;

# }

# public override string ToString()

# {

# return this.FigureType + " шириной " + this.width.ToString() + ", высотой " + this.height.ToString() + ", площадью " + this.Area().ToString();

# }

# public void Print()

# {

# Console.WriteLine(this.ToString());

# }

# }

# }

# Результаты

