**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 |
| Ширшов Артём |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

# Задание.

**Часть 1. Разработать программу, использующую делегаты.**

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

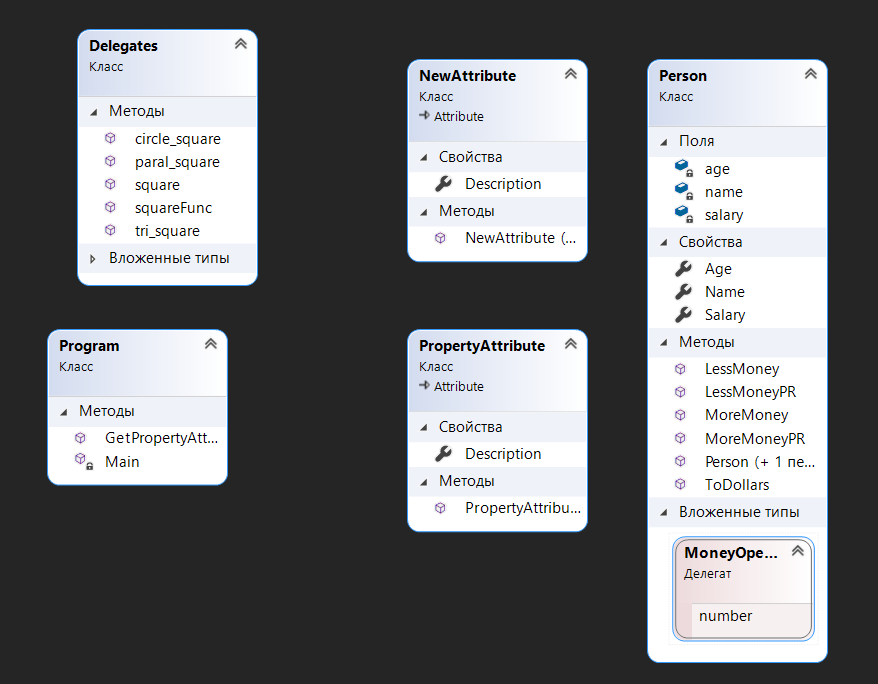
1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
3. Напишите метод, соответствующий данному делегату.
4. Напишите метод, принимающий разработанный Вами делегат, 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.
6. Вызовите один из методов класса с использованием рефлексии.

# Диаграмма классов



# Текст программы

Delegates.cs

using System;

using System.Collections.Generic;

namespace lab6

{

class Delegates

{

public delegate string square\_count(double first, double second, string name);

static public string tri\_square(double length, double height, string name)

{

double result = 0.5\*length\*height;

return ($"Треугольник {name} со параметрами {length} и {height}. Площадь равна:{result}");

}

static public string paral\_square(double length,double height, string name)

{

double result =length\*height;

return ($"Параллелограмм {name} со параметрами {length} и {height}. Площадь равна:{result}");

}

static public string circle\_square(double radius,double pi,string name)

{

pi=Math.PI;

double result=pi\*radius\*radius;

return ($"Круг {name} с радиусом {radius}. Площадь равна:{result}");

}

static public string square (double length, double height, string name, square\_count result)

{

return result(length,height,name);

}

static public string squareFunc (double length, double height, string name, Func <double,double,string,string> result)

{

return result(length,height, name);

}

}

}

Reflection.cs

using System;

using System.Reflection;

namespace lab6

{

[AttributeUsage(AttributeTargets.Method,AllowMultiple=false, Inherited= false)]

public class NewAttribute :Attribute

{

public NewAttribute() {}

public NewAttribute(string param)

{

Description=param;

}

public string Description {get; set;}

}

[ AttributeUsage(AttributeTargets.Property,AllowMultiple=false, Inherited= false)]

public class PropertyAttribute:Attribute

{

public PropertyAttribute() {}

public PropertyAttribute(string param)

{

Description=param;

}

public string Description {get; set;}

}

class Person

{

delegate void MoneyOperations(double number);

private int age;

[PropertyAttribute("Возраст сотрудника")]

public int Age {

get{

return age;

}

set{

age=value;

}

}

private string name;

public string Name {

get{

return name;

}

set{

name=value;

}

}

private double salary;

[PropertyAttribute("Зарплата сотрудникам")]

public double Salary {

get{

return salary;

}

set{

salary=value;

}

}

public Person(string name, int age){

Age=age;

Name=name;

Salary =50000;

}

public Person(string name, int age, double salary){

Age=age;

Name=name;

Salary=salary;

}

[NewAttribute("Увеличить зарплату в процентах сотруднику")]

public void MoreMoneyPR(double percent){

Salary = Salary/100\*(100+percent);

}

public void LessMoneyPR(double percent){

Salary = Salary/100\*(100-percent);

}

[NewAttribute("Увеличить зарплату сотруднику на определённую сумму")]

public void MoreMoney(double number){

Salary = Salary+number;

}

public void LessMoney(double number){

Salary = Salary+number;

}

public double ToDollars(){

return Math.Round(Salary/77.18,2);

}

}

}

Program.cs

using System;

using System.Reflection;

namespace lab6

{

class Program

{

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 Main(string[] args)

{

double a=10;

double b=5;

string name ="ARTYOM";

System.Console.WriteLine(Delegates.square(a,b,"OR",Delegates.circle\_square));

Delegates.square\_count lbd\_square\_count = (x,y,z) =>

{

double result = (x\*y)\*2/5;

return $"Фигура {z} со параметрами {x} и {y}. Площадь равна:{result}. Лямбда-выражение";

};

System.Console.WriteLine(lbd\_square\_count(20,15.3,"FRETY"));

System.Console.WriteLine(Delegates.square(a,b,name,(x,y,z) =>

{

double result = (x\*y)\*2/5;

return $"Фигура {z} со параметрами {x} и {y}. Площадь равна:{result}. Лямбда-выражение в качестве аргумента";

}));

System.Console.WriteLine(Delegates.squareFunc(a,b,"ABC",Delegates.tri\_square));

//-----------------------------------------------------------------------------------------

Person obj =new Person("Artyom",18);

Type t =obj.GetType();

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("\nСвойства, помеченные атрибутом:");

foreach (var x in t.GetProperties())

{

object attrObj;

if (GetPropertyAttribute(x, typeof(PropertyAttribute), out attrObj))

{

PropertyAttribute attr = attrObj as PropertyAttribute;

Console.WriteLine(x.Name + " - " + attr.Description);

}

}

Console.WriteLine("\nВызов метода:");

Person fi = new Person("Artyom",19);

object[] parametrs = new object[] {};

object Result = t.InvokeMember("ToDollars", BindingFlags.InvokeMethod, null, fi, parametrs);

System.Console.WriteLine("ToDollars={0}$", Result);

}

}

}

# Результаты

