Московский государственный технический университет

им. Н.Э. Баумана

Отчет по лабораторной работе № 6

|  |  |
| --- | --- |
|  | Выполнила:  Персиянова В.  ИУ5-33 |

г. Москва

2017 г.

**Описание задания лабораторной работы:**

**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:

* 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.**

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 кла сс 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.

**Текст программы на языке C#.**

**Часть 1.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

public delegate int d1(int x1, string x2);

public static List<int> SumArrays(List<int> x, List<string> str, d1 func)

{

List<int> t = new List<int>();

for (int i = 0; i < x.Count; i++)

{

t.Add(func(x[i], str[i]));

}

return t;

}

public static List<int> SumArraysFunc(List<int> x, List<string> str, Func<int, string, int> func)

{

List<int> t = new List<int>();

for (int i = 0; i < x.Count; i++)

{

t.Add(func(x[i], str[i]));

}

return t;

}

public static int Sum(int x, string str)

{

int result = Int32.Parse(str);

return x + result;

}

static void Main(string[] args)

{

List<int> x = new List<int>();

x.Add(10);

x.Add(20);

x.Add(30);

List<string> str = new List<string>();

str.Add("5");

str.Add("10");

str.Add("15");

d1 sum = Sum;

Console.WriteLine("Вызов метода через обычный делегат");

List<int> l1 = SumArrays(x, str, sum);

foreach (var i in l1)

{

Console.WriteLine(i);

}

Console.WriteLine();

Console.WriteLine("Вызов метода через лямбда-варажение");

List<int> l2 = SumArrays(x, str, (x\_1, str\_1) => x\_1 \* Int32.Parse(str\_1));

foreach (var i in l2)

{

Console.WriteLine(i);

}

Console.WriteLine();

Console.WriteLine("Вызов метода через обобщённый делегат Func();");

Func<int, string, int> sum1 = Sum;

List<int> l3 = SumArraysFunc(x, str, sum1);

foreach (var i in l3)

{

Console.WriteLine(i);

}

Console.WriteLine();

Console.ReadKey();

}

}

}

**Часть 2.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6\_Path2

{

class Human

{

[Attr("Имя")]

private int age { get; set; }

public string name { get; set; }

[Attr("Питомец")]

public string pet { get; set; }

public Human(string name, int age)

{

this.age = age;

this.name = name;

}

public Human(string name)

{

this.age = 22;

this.name = name;

}

public Human(int age)

{

this.name = "Тимофей";

this.age = age;

}

public void human(int type)

{

switch (type)

{

case 1:

Console.WriteLine("Собака.");

break;

case 2:

Console.WriteLine("Кот.");

break;

case 3:

Console.WriteLine("Жираф.");

break;

default:

Console.WriteLine("Нет такого животного.");

break;

}

}

}

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

class Attr : Attribute

{

public string description { get; set; }

public Attr(string str\_description)

{

description = str\_description;

}

}

class Program

{

static void Main(string[] args)

{

Type t = typeof(Human);

Console.WriteLine("Все конструкторы:");

foreach (var x in t.GetConstructors())

{

Console.WriteLine(x);

}

Console.WriteLine("\nВсе свойства:");

foreach (var x in t.GetProperties())

{

Console.WriteLine(x);

}

Console.WriteLine("\nВсе методы:");

foreach (var x in t.GetMethods())

{

Console.WriteLine(x);

}

Console.WriteLine("\nАтрибуты свойств:");

foreach (var x in t.GetProperties())

{

var isAttr = x.GetCustomAttributes(typeof(Attr), false);

if (isAttr.Length > 0)

{

Console.WriteLine(isAttr[0]);

}

}

Human Human1 = new Human(18);

object[] param = new object[] { 3 };

Console.WriteLine();

t.InvokeMember("human", System.Reflection.BindingFlags.InvokeMethod, null, Human1, param);

//1 - string, содержащий имя конструктора;

//2 - invokeAttr . Как ведётся поиск. Указывать необязательно;

//3 - binder . объект, определяющий набор свойств и разрешающий привязку;

//4 - target.

//5 - args. массив с аргументами, который передаётся вызываевому члену.

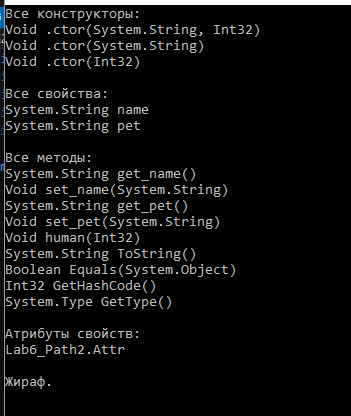
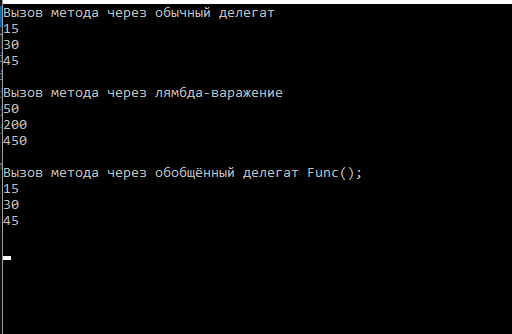
Console.ReadKey();

}

}

}

**Результаты выполнения программы:**



**Диаграмма классов:**

