**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5.  
Курс «Базовые компоненты интернет технологий»

Отчет по лабораторной̆ работе №6

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  | преподаватель каф. ИУ5 |
| Рябова В |  | Гапанюк Ю. Е. |
| Подпись и дата: |  | Подпись и дата: |

г. 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:

Lab6\_1

Изображение выглядит как текст, снимок экрана, монитор, внутренний

Описание создано с очень высокой степенью достоверности

Lab6\_2

Изображение выглядит как текст, снимок экрана, компьютер, монитор

Описание создано с очень высокой степенью достоверности

Изображение выглядит как текст, снимок экрана, компьютер, монитор

Описание создано с очень высокой степенью достоверности

Текст программы:

Lab6\_1

//delegates

using System;

using System.Runtime.InteropServices.ComTypes;

namespace lab6\_1

{

delegate double CknorAkn(int a, int b);

class Program

{

static int Factorial(int num)

{

if (num <= 1)

{

return 1;

}

else

return num \* Factorial(num - 1);

}

static double Ckn(int k, int n) //Сочетания из n по k

{

return (Factorial(n))/((Factorial(n-k))\*(Factorial(k)));

}

static double Akn(int k, int n) //Размещения из n по k

{

return (Factorial(n)) / (Factorial(n - k));

}

static void CknorAknMethod(int a, int b, CknorAkn nonParam)

{

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

}

static void Main(string[] args)

{

//вызов метода с использованием метода, соответствующего делегату

CknorAknMethod(2, 5, Ckn);

CknorAknMethod(2, 5, Akn);

//вызов метода с использованием лямбда-выражения

CknorAknMethod(2, 5, (int x, int y) => { return Ckn(x, y); });

CknorAknMethod(2, 5, (int x, int y) => { return Akn(x, y); });

//использование обобщённого делегата

void CknorAknMethodFunc(int a, int b, Func<int, int, double> nonParam)

{

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

}

CknorAknMethodFunc(2, 5, Ckn);

CknorAknMethodFunc(2, 5, Akn);

}

}

}

Lab6\_2

using System;

using System.Reflection;

namespace lab6\_2

{

class Program

{

public class AgeValidationAttribute : System.Attribute

{

public int Age { get; set; }

public AgeValidationAttribute()

{

}

public AgeValidationAttribute(int age)

{

this.Age = age;

}

}

[AgeValidation(18)]

public class Person

{

public string Name { get; set; }

public uint Age { get; set; }

public uint Size { get; set; }

public double Money { get; set; }

public Person(string name, uint size, double money, uint age)

{

this.Size = size;

this.Age = age;

this.Name = name;

this.Money = money;

}

}

public class Cloth

{

public string model { get; private set; }

public double price { get; private set; }

public string brand { get; private set; }

public int size { get; private set; }

public int yearofcollection { get; private set; }

public static uint count { get; private set; }

public Cloth(string model, double price, int size, string brand, int yearofcollection)

{

this.model = model;

this.price = price; ;

this.size = size;

this.brand = brand;

this.yearofcollection = yearofcollection;

count = 20;

}

static bool validateCustomer(Person customer)

{

Type t = customer.GetType();

var attributes = t.GetCustomAttributes(typeof(AgeValidationAttribute), false);

foreach (AgeValidationAttribute attribute in attributes)

{

return customer.Age >= attribute.Age;

}

return true;

}

public bool Buy(Person customer)

{

if (count <= 0)

return false;

if ((validateCustomer(customer) && customer.Money >= this.price)&& (validateCustomer(customer) && customer.Size == this.size))

{

customer.Money -= this.price;

count--;

return true;

}

return false;

}

public void Use()

{

Console.WriteLine("Oh, I bought a new year costume!");

}

public void Display()

{

Console.WriteLine($"Model: {this.model} | Price: {this.price} \n" +

$" Size: {this.size} | Brand: {this.brand} \n" +

$" Year of Collection: {this.yearofcollection}");

}

}

static void Main(string[] args)

{

try

{

Type classCloth = typeof(Cloth);

Console.WriteLine("Fields:");

foreach (FieldInfo field in classCloth.GetFields())

{

Console.WriteLine($"{field.FieldType} {field.Name}");

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Properties:");

foreach (PropertyInfo prop in classCloth.GetProperties())

{

Console.WriteLine($"{prop.PropertyType} {prop.Name}");

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Methods:");

foreach (MethodInfo method in classCloth.GetMethods())

{

string modificator = string.Empty;

if (method.IsStatic)

modificator += " static";

if (method.IsVirtual)

modificator += " virtual";

Console.Write($"{modificator} {method.ReturnType.Name} {method.Name} (");

var parameters = method.GetParameters();

for (int i = 0; i < parameters.Length; i++)

{

Console.Write($"{parameters[i].ParameterType.Name} {parameters[i].Name}");

if (i + 1 < parameters.Length) Console.Write(", ");

}

Console.WriteLine(")");

}

}

catch (Exception ex)

{

throw ex;

}

Cloth dress = new Cloth("Dress", 3000, 36, "Zara", 2020);

Person margo = new Person("Margo", 36, 15000, 18);

Cloth pants = new Cloth("Pants", 2000, 44, "Zara Man", 2019);

Person alex = new Person("Alexey", 46, 3000, 32);

Cloth pants2 = new Cloth("Pants", 2999, 46, "Zara Man", 2019);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nMargo buying dress: " + dress.Buy(margo));

Console.WriteLine("Alexey buying Pants 1: " + pants.Buy(alex));

Console.WriteLine("Alexey buying Pants 2: " + pants2.Buy(alex));

}

}

}

Результаты работы программы:

Lab6\_1

Изображение выглядит как текст, снимок экрана, монитор, экран

Описание создано с очень высокой степенью достоверности

Изображение выглядит как текст

Описание создано с очень высокой степенью достоверности