|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ                              Информатика и системы управления

КАФЕДРА                  Системы обработки информации и управления

**Лабораторная работа №6**

**По курсу «Базовые компоненты интернет-технологий»**

Подготовила:

Студентка группы ИУ5-35Б.

Очеретная С.В.

18.11.2020

Проверил:

Преподаватель кафедры ИУ5  
Гапанюк Ю.Е.

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.

Текст программы

**Делегаты**

using System;

namespace LABA6.\_1

{

class Program

{

delegate string Person(bool ill, int age); //(2)

static string russian(bool ill, int age) //(3)

{

string regime;

if (age > 65) regime = "Соблюдайте особый режим!";

else regime = "Лучше сидите дома";

if (ill) return "Наставление: " + regime;

else return "Можно не беспокоиться";

}

static string american(bool ill, int age) //(3)

{

string regime;

if (age > 65) regime = "Observe a special regime!";

else regime = "Better stay at home";

if (ill) return "Manual: " + regime;

else return "Don't worry";

}

static void Coronavirus\_test(bool ill, int age, Person p) //(4)

{

string result = p(ill, age);

Console.Write("TEST Message: {0} \n", result);

}

static void Coronavirus\_test2(bool ill, int age, Func<bool,int,string> p) //(5)

{

string result = p(ill, age);

Console.Write("TEST Message: {0} \n", result);

}

public void Coronavirus\_test3(bool ill, int age, Action<bool, int, string> p) //(5.1)

{

string result = "";

p(ill, age, result);

Console.Write("TEST Message: {0} \n", result);

}

static void Main(string[] args)

{

Console.WriteLine("COVID TEST\n");

Coronavirus\_test(true, 20, american); //с параметром-делегатом(4)

Person dead = (bool ill, int age) => { return "Life is sad"; }; //лямбда-выражение (4)

Coronavirus\_test(false, 50, dead);

Coronavirus\_test2(true, 66, russian); //обобщенный делегат (5)

Coronavirus\_test2(true, 66, russian);

}

}

}

**Рефлексия**

using System;

using System.Reflection;

using System.Text.RegularExpressions;

using System.Threading.Tasks.Dataflow;

namespace LABA6.\_2

{

public class Student : IComparable //(2)

{

public Student() { } //конструкторы

public Student(string name) { }

public Student(string name, int mark) { }

[MyAttribute(Description = "Назначение аттрибута свойству оценка")]

public int mark //свойства

{

get { return mark; }

set { mark = value; }

}

[MyAttribute(Description = "Назначение аттрибута свойству имя")]

public string name

{

get { return name; }

set { name = value; }

}

public string group

{

get { return group; }

set { group = value; }

}

public string Statistics (string name, int mark) //методы

{

Console.WriteLine("Statictics: ");

if (mark <= 2) return name + " отчислен";

else return name + " оценка: " + mark;

}

public int increaseMark (int mark)

{

if (mark < 5) return mark + 1;

else return mark;

}

public int CompareTo(object obj) { return 0; }

public static bool IsAttributeInProperty(Type attributeType, PropertyInfo info, out object attribute) // info - информация о проверяемом свойстве

{

bool result = false;

attribute = null;

var isAttribute = info.GetCustomAttributes(attributeType, false); // поиск аттрибутов с заданным типом, возвращается коллекция , в атрибут - 1 элемент коллекции

if (isAttribute.Length > 0)

{

result = true;

attribute = isAttribute[0];

}

return result;

}

}

[AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inherited = false)] /\*..Targets - для каких элементов классов используется,

..Multiple - применение нескольких атрибуттов, Inherited - наследование производными классами\*/

public class MyAttribute : Attribute //(4)

{

public MyAttribute() { }

public string Description { get; set; }

public MyAttribute(string description){ Description = description; }

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Ocheretnaya Svetlana IU5-35B\n");

Student D = new Student();

Type typeD = D.GetType(); //(3)

Console.WriteLine("Тип класса: " + typeD.FullName + "\nПространство:" + typeD.Namespace + "\nСборка:" + typeD.AssemblyQualifiedName);

Console.WriteLine("Конструкторы: ");

foreach(var t in typeD.GetConstructors()) { Console.WriteLine(t); }

Console.WriteLine("\nСвойства: ");

foreach (var t in typeD.GetProperties()) { Console.WriteLine(t); }

Console.WriteLine("\nМетоды: ");

foreach (var t in typeD.GetMethods()) { Console.WriteLine(t); }

Type info = typeof(Student); //(5)

Console.WriteLine("\nСвойства, которым назначен атрибут: ");

foreach (var property in info.GetProperties())

{

object attr;

if (Student.IsAttributeInProperty(typeof(MyAttribute), property, out attr)) {

MyAttribute myattr = attr as MyAttribute;

Console.WriteLine(property.Name + ": " + myattr.Description);

}

}

Student st = (Student)info.InvokeMember(null, BindingFlags.CreateInstance, null, null, new object[] { });

Console.WriteLine(info.InvokeMember("Statistics", BindingFlags.InvokeMethod, null, st, new object[] { "Mike", 4 })); //вызов метода с помощью рефлексии(6)

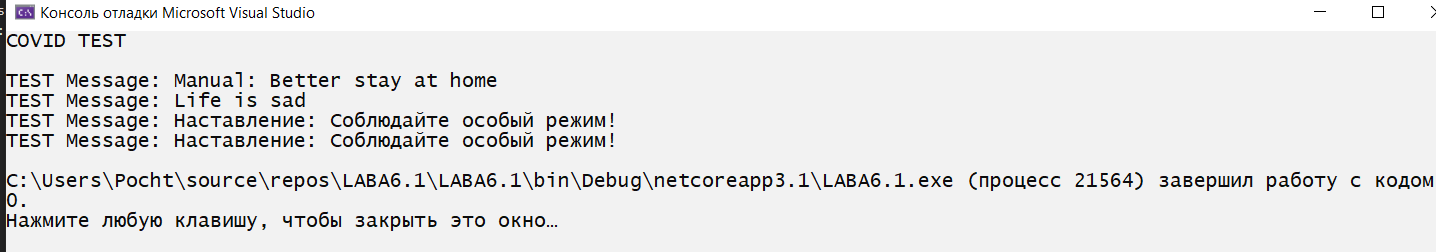
}

}

}

Результаты

Делегаты:



Рефлексия:

