**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнил: |  |  |  |  |
| студент группы ИУ5-33Б |  |  |  |  | |
|  |  |  |  |  |
| Алёшин Александр |  |  |  |  |

Москва, 2020 г.

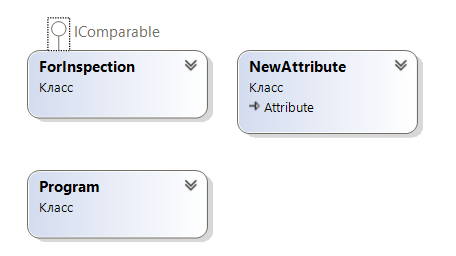
Условие лабораторной работы:

**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:

NewAttribute.cs

using System;

namespace lab.\_6\_2

{

/// <summary>

/// Класс атрибута

/// </summary>

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

public class NewAttribute : Attribute

{

public NewAttribute() { }

public NewAttribute(string DescriptionParam)

{

Description = DescriptionParam;

}

public string Description { get; set; }

}

}

ForInspection.cs

using System;

namespace lab.\_6\_2

{

/// <summary>

/// Класс для исследования с помощью рефлексии

/// </summary>

public class ForInspection : IComparable

{

public ForInspection() { }

public ForInspection(int i) { }

public ForInspection(string str) { }

public int Plus(int x, int y) { return x + y; }

public int Minus(int x, int y) { return x - y; }

[NewAttribute("Описание для property1")]

public string property1

{

get { return \_property1; }

set { \_property1 = value; }

}

private string \_property1;

public int property2 { get; set; }

[NewAttribute(Description = "Описание для property3")]

public double property3 { get; private set; }

public int field1;

public float field2;

/// <summary>

/// Реализация интерфейса IComparable

/// </summary>

public int CompareTo(object obj)

{

return 0;

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace lab.\_6\_2

{

class Program

{

/// <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;

}

/// <summary>

/// Получение информации о текущей сборке

/// </summary>

static void AssemblyInfo()

{

Console.WriteLine("Вывод информации о сборке:");

Assembly i = Assembly.GetExecutingAssembly();

Console.WriteLine("Полное имя:" + i.FullName);

Console.WriteLine("Исполняемый файл:" + i.Location);

}

/// <summary>

/// Получение информации о типе

/// </summary>

static void TypeInfo()

{

ForInspection obj = new ForInspection();

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("\nForInspection реализует IComparable -> " +

t.GetInterfaces().Contains(typeof(IComparable))

);

}

/// <summary>

/// Пример использования метода InvokeMember

/// </summary>

static void InvokeMemberInfo()

{

Type t = typeof(ForInspection);

Console.WriteLine("\nВызов метода:");

//Создание объекта

//ForInspection fi = new ForInspection();

//Можно создать объект через рефлексию

ForInspection fi = (ForInspection)t.InvokeMember(null, BindingFlags.CreateInstance, null, null, new object[] { });

//Параметры вызова метода

object[] parameters = new object[] { 3, 2 };

//Вызов метода

object Result = t.InvokeMember("Plus", BindingFlags.InvokeMethod, null, fi, parameters);

Console.WriteLine("Plus(3,2)={0}", Result);

}

/// <summary>

/// Работа с атрибутами

/// </summary>

static void AttributeInfo()

{

Type t = typeof(ForInspection);

Console.WriteLine("\nСвойства, помеченные атрибутом:");

foreach (var x in t.GetProperties())

{

object attrObj;

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

{

NewAttribute attr = attrObj as NewAttribute;

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

}

}

}

static void Main(string[] args)

{

AssemblyInfo();

TypeInfo();

InvokeMemberInfo();

AttributeInfo();

Console.ReadLine();

}

}

}

Пример работы:

