**Занятие № 18**

**Дата выполнения работы:** 11.05.2023

# **Тема работы:** «Рефлексия»

**Ход работы**

**Задание 1**

1. Для изучения .NET Reflection API допишите класс Рефлектор,

который будет содержать методы выполняющие следующие действия:

a. выводит всё содержимое класса в текстовый файл

(принимает в качестве параметра имя класса);

b. извлекает все общедоступные публичные методы класса

(принимает в качестве параметра имя класса);

c. получает информацию о полях и свойствах класса;

d. получает все реализованные классом интерфейсы;

e. выводит по имени класса имена методов, которые содержат

заданный (пользователем) тип параметра (имя класса

передается в качестве аргумента);

f. вызывает некоторый метод класса, при этом значения для

его параметров необходимо прочитать из текстового файла

(имя класса и имя метода передаются в качестве

аргументов).

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

using System.Security.Cryptography.X509Certificates;

using System.IO;

namespace pr18

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Reflector.write(textBox1.Text);

foreach(MethodInfo method in Reflector.writemeth(textBox1.Text))

{

listBox1.Items.Add(method.Name);

}

foreach (FieldInfo method in Reflector.writefield(textBox1.Text))

{

listBox1.Items.Add(method.Name);

}

foreach (Type method in Reflector.writeinter(textBox1.Text))

{

listBox1.Items.Add(method.Name);

}

Reflector.L(textBox1.Text);

}

}

public class mem

{

public string name;

public int age;

public int count { get; set; }

public static int s;

public mem(string name, int age, int count)

{

this.name = name;

this.age = age;

this.count = count;

}

public static void now()

{

s = 0;

}

}

public class Reflector

{

public static void write(string s)

{

Type type = Type.GetType("pr18." + s, false, true);

TypeInfo typeInfo = type.GetTypeInfo();

MemberInfo[] memberInfo = typeInfo.GetMembers();

using (StreamWriter writer = new StreamWriter("text.txt"))

{

foreach (MemberInfo member in memberInfo)

{

writer.WriteLine(member.Name);

}

}

}

public static MethodInfo[] writemeth(string s)

{

Type type = Type.GetType("pr18." + s, false, true);

TypeInfo typeInfo = type.GetTypeInfo();

return typeInfo.GetMethods();

}

public static FieldInfo[] writefield(string s)

{

Type type = Type.GetType("pr18." + s, false, true);

TypeInfo typeInfo = type.GetTypeInfo();

return typeInfo.GetFields();

}

public static Type[] writeinter(string s)

{

Type type = Type.GetType("pr18." + s, false, true);

TypeInfo typeInfo = type.GetTypeInfo();

return typeInfo.GetInterfaces();

}

public static void L(string s)

{

Type type = Type.GetType("pr18." + s, false, true);

type.GetMethod("now").Invoke(null, new object[] { });

}

}

public class Computer

{

public mem mem;

public string name;

public int age;

public static int s;

public int count { get; set; }

public Computer(string name, int age, int count)

{

mem = new mem(name, age, count);

this.name = name;

this.age = age;

this.count = count;

}

public static void now()

{

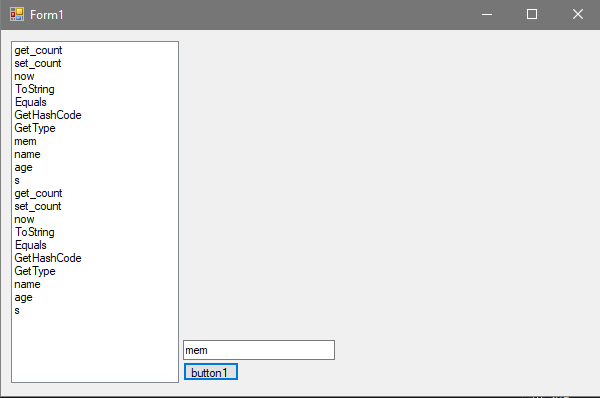
s = 0;

}

}

}

**Результат:**

****

**Вывод:** Рефлексия в C# - это мощный инструмент, который позволяет анализировать и изменять типы, их поведение и информацию во время выполнения программы. Она предоставляет доступ к метаданным типов, позволяет создавать экземпляры, вызывать методы, обращаться к свойствам и изменять значения. Рефлексия полезна, когда требуется динамическая работа с типами, однако ее использование может привести к снижению производительности. Поэтому следует использовать рефлексию осторожно и только там, где она действительно необходима.