Московский Государственный Университет им. Н.Э. Баумана

Отчет по лабораторной работе №6

по курсу БКИТ

# Выполнил:

## Болотин Андрей

ИУ5-32

Громов Вадим

ИУ5-31

# Условие задачи

**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 входных 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 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.

Код (часть 1)

## Файл: Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Lab6\_1

{

class Program

{

public delegate float roots(float p1, int p2);

static roots discr=(float p1, int p2) =>

{

return p1\*p1-4\*p2;

};

static Func<float,int,float> discr1 = (float p1, int p2) =>

{

return p1 \* p1 - 4 \* p2;

};

static float Findroot(float p1, int p2)

{

return p1 / (2 \* p2);

}

static void equation(int a, int b, int c, roots discr, roots root)

{

if (discr(b, a \* c) > 0)

{

Console.WriteLine("Root1 1: " + root(-b + discr(b, a \* c), a));

Console.WriteLine("Root1 2: " + root(-b - discr(b, a \* c), a));

}

else

if (discr(b, a \* c) == 0)

Console.WriteLine("Root1 : " + root(-b, a));

else

if (discr(b, a \* c) < 0)

Console.WriteLine("No roots");

}

static void equation1(int a, int b, int c, Func<float,int,float> discr, Func<float, int, float> root)

{

if (discr(b, a \* c) > 0)

{

Console.WriteLine("Root1 1: " + root(-b + discr(b, a \* c), a));

Console.WriteLine("Root1 2: " + root(-b - discr(b, a \* c), a));

}

else

if (discr(b, a \* c) == 0)

Console.WriteLine("Root1 : " + root(-b, a));

else

if (discr(b, a \* c) < 0)

Console.WriteLine("No roots");

}

static void Main(string[] args)

{

int a, b, c;

Console.WriteLine("Please put in a, b, c");

Console.Write("a: ");

a = Int32.Parse(Console.ReadLine());

Console.Write("b: ");

b = Int32.Parse(Console.ReadLine());

Console.Write("c: ");

c = Int32.Parse(Console.ReadLine());

Console.WriteLine();

Console.WriteLine("Finding roots using delegates and lambda expression:");

equation(a, b, c, discr, Findroot);

Console.WriteLine();

Console.WriteLine("Finding roots using Func:");

equation1(a, b, c, discr1, Findroot);

Console.WriteLine();

Console.WriteLine("Press a key to continue");

Console.Read();

}

}

}

Код (часть 2)

## Файл: Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Reflection;

namespace lab6\_2

{

class Program

{

public static bool GetPropertyAttribute(PropertyInfo checkType, Type attributeType, out object attribute)

{

bool Result = false;

attribute = null;

var isAttribute = checkType.GetCustomAttributes(attributeType, false);

if (isAttribute.Length > 0)

{

Result = true;

attribute = isAttribute[0];

}

return Result;

}

static void Main(string[] args)

{

Cat mycat = new Cat();

Type t = mycat.GetType();

Console.WriteLine("\nInformation of type");

Console.WriteLine("Namespace " + t.Namespace);

Console.WriteLine("Assembly Qualified " + t.AssemblyQualifiedName);

Console.WriteLine("\nConstructors");

foreach (var x in t.GetConstructors())

{

Console.WriteLine(x);

}

Console.WriteLine("\nMethods:");

foreach (var x in t.GetMethods())

{

Console.WriteLine(x);

}

Console.WriteLine("\nProperties:");

foreach (var x in t.GetProperties())

{

Console.WriteLine(x);

}

Console.WriteLine("\nData (public):");

foreach (var x in t.GetFields())

{

Console.WriteLine(x);

}

Console.WriteLine("\nProperties with attributes:");

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);

}

}

Console.WriteLine("\nMethod by using reflection");

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

object[] parameters = new object[] {2,5};

object Result = t.InvokeMember("Cat\_moves", BindingFlags.InvokeMethod, null, myCat, parameters);

Console.WriteLine("Current place " + Result);

Console.WriteLine("\nPress a key to continue");

Console.Read();

}

}

}

## Файл: Cat.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab6\_2

{

class Cat

{

String front\_laps;

String back\_laps;

String ears;

String tail;

public int years;

public Cat()

{

}

public Cat(string fl, string bl, string ear, string ctail , int year)

{

front\_laps = fl;

back\_laps = bl;

ears = ear;

years = year;

tail = ctail;

}

public int Cat\_moves(int cur\_pos, int steps)

{

int pos = cur\_pos + steps;

return pos;

}

[NewAttribute("Front claws")]

public String Front\_Laps

{

get

{

return front\_laps;

}

set

{

front\_laps = value;

}

}

public String Back\_Laps

{

get

{

return back\_laps;

}

set

{

back\_laps = value;

}

}

public String Ears

{

get

{

return ears;

}

set

{

ears = value;

}

}

[NewAttribute("Tail of a cat")]

public String Tail

{

get

{

return tail;

}

set

{

tail = value;

}

}

}

}

## Файл: NewAttribute.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace lab6\_2

{

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

public class NewAttribute : Attribute

{

public NewAttribute() { }

public NewAttribute(string DescriptionParam)

{

Description = DescriptionParam;

}

public string Description { get; set; }

}

}

# Примеры работающей программы



