МГТУ им. Баумана  
Факультет «Информатика и системы управления»  
Кафедра «Автоматизированные системы обработки информации и управления»  
Дисциплина «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе №6

Выполнил:  
Студент группы ИУ5-33Б  
Андреев А.А.   
Преподаватель:  
Гапанюк Ю.

1. **Описание задания**
   1. Разработать программу, использующую делегаты.
      1. (В качестве примера можно использовать проект «Delegates»).
      2. Программа должна быть разработана в виде консольного приложения на языке C#.
      3. Определите делегат, принимающий несколько параметров различных типов и возвращающий значение произвольного типа.
      4. Напишите метод, соответствующий данному делегату.
      5. Напишите метод, принимающий разработанный Вами делегат, в качестве одного из входным параметров. Осуществите вызов метода, передавая в качестве параметра-делегата:
      6. метод, разработанный в пункте 3;
      7. лямбда-выражение.
      8. Повторите пункт 4, используя вместо разработанного Вами делегата, обобщенный делегат Func< > или Action< >, соответствующий сигнатуре разработанного Вами делегата.
   2. Разработать программу, реализующую работу с рефлексией**.**
      1. (В качестве примера можно использовать проект «Reflection»).
      2. Программа должна быть разработана в виде консольного приложения на языке C#.
      3. Создайте класс, содержащий конструкторы, свойства, методы.
      4. С использованием рефлексии выведите информацию о конструкторах, свойствах, методах.
      5. Создайте класс атрибута (унаследован от класса System.Attribute).
      6. Назначьте атрибут некоторым свойствам классам. Выведите только те свойства, которым назначен атрибут.
      7. Вызовите один из методов класса с использованием рефлексии.
2. **Текст программы**

**(Файл 1)**

**Элемент 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("корень1 1: " + root(-b + discr(b, a \* c), a));

Console.WriteLine("корень1 2: " + root(-b - discr(b, a \* c), a));

}

else

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

Console.WriteLine("корень1 : " + root(-b, a));

else

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

Console.WriteLine("нет корней");

}

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("корень1 1: " + root(-b + discr(b, a \* c), a));

Console.WriteLine("корень1 2: " + root(-b - discr(b, a \* c), a));

}

else

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

Console.WriteLine("корень1 : " + root(-b, a));

else

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

Console.WriteLine("нет корней");

}

static void Main(string[] args)

{

int a, b, c;

Console.WriteLine("введите 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("нахождение корней при помощи делегатов и лямбда-выражения:");

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

Console.WriteLine();

Console.WriteLine("поиск коренй при помощи Func:");

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

Console.WriteLine();

Console.WriteLine("нажмите чтобы продолжить");

Console.Read();

}

}

}

**(файл 2)**

**Элемент 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("\nинформация типа");

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

Console.WriteLine("Квалифицированная сборка " + 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("\nData (public):");

foreach (var x in t.GetFields())

{

Console.WriteLine(x);

}

Console.WriteLine("\nсвойства с артибутами:");

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("\nметод с использованием рефлексии");

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

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

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

Console.WriteLine("нынешнее местоположение " + Result);

Console.WriteLine("\nнажмите кнопку чтобы продолжить");

Console.Read();

}

}

}

**Элемент 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; }

}

}

**Элемент 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("передние когти")]

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("хвост")]

public String Tail

{

get

{

return tail;

}

set

{

tail = value;

}

}

}

}

1. **Экранные формы с примерами выполнения программы**



