МГТУ имени Н.Э. Баумана
Факультет «Информатика и системы управления»

Базовые компоненты интернет технологий. Отчёт по лабораторной работе № 6.

Белкина Екатерина
Группа ИУ5-31Б

24 декабря 2018

Условие задания:

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 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.

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;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Win32;

using Lab5;

namespace Homework
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private List<string> _words;
        public MainWindow()
        {
            InitializeComponent();
        }

        private void OpenFile_Click(object sender, RoutedEventArgs e)
        {
            var dialog = new OpenFileDialog
            {
                Filter = "Text Files(*.txt) | *.txt"
            };

            var timer = new Stopwatch();

            if (dialog.ShowDialog() == true)
            {
                timer.Start();
                var file = File.ReadAllText(dialog.FileName);
                _words = file.Trim('.').Trim(',').Split(' ').Distinct().ToList();
                timer.Stop();

                openTimer.Content = "Время чтения: " + timer.ElapsedMilliseconds + " мс";
            }

        }
```

```
private void Search_Click(object sender, RoutedEventArgs e)
{
    listBox.Items.Clear();

    var expectedWord = findWord.Text.Trim(' ');

    if (expectedWord == "")
    {
        listBox.Items.Add("Введите слово для поиска");
        return;
    }

    if (maxDist.Text.Trim(' ') == "")
    {
        listBox.Items.Add("Введите максимальное расстояние");
        return;
    }

    if (countOfThreads.Text.Trim(' ') == "")
    {
        listBox.Items.Add("Введите количество потоков");
        return;
    }

    var dist = int.Parse(maxDist.Text);
    var tasksCount = int.Parse(countOfThreads.Text);
    var timer = new Stopwatch();
    var tasks = new Task<List<string>>[tasksCount];

    var listOfWords = new List<List<string>>();
    for (var i = 0; i < tasksCount; i++)
    {
        listOfWords.Add(new List<string>());
    }

    for (var i = 0; i < _words.Count; i += tasksCount)
    {
        for (var j = 0; j < tasksCount && i + j < _words.Count; j++)
        {
            listOfWords[j].Add(_words[i + j]);
```

```
    }
}

timer.Start();

for (var i = 0; i < tasksCount; i++)
{
    var list = listOfWords[i];
    tasks[i] = Task.Run(() =>
    {
        var findList = new List<string>();
        foreach (var word in list)
        {
            if (Lab5.Lab5.Dist(word, expectedWord) <= dist)
            {
                findList.Add(word);
            }
        }

        return findList;
    });
}


Task.WaitAll(tasks);

foreach (var task in tasks)
{
    foreach (var word in task.Result)
    {
        listBox.Items.Add(word);
    }
}

timer.Stop();

            Console.WriteLine("Count of tasks: " + tasksCount + " has found " +
listBox.Items.Count + " words");

if (listBox.Items.Count == 0)
```

```
        {
            listBox.Items.Add("Нет совпадений");
        }

        searchTimer.Content = "Время поиска: " + timer.ElapsedMilliseconds + " мс";
    }
  }
}
```

РЕФЛЕКСИЯ

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
  class Program
  {
    static void Main(string[] args)
    {
      Type t = typeof(Class1);
      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);
        }

        //Properties with attributes
        Console.WriteLine("\nСвойства(С атрибутом):");
        foreach (var x in t.GetProperties())
        {
           if (Attribute.IsDefined(x, typeof(NewAttribute)))
           {
              Console.WriteLine(x);
           }
        }

        Console.WriteLine("\nВызов метода Plus:");
            Console.WriteLine(t.GetMethod("Plus").Invoke(new Class1(), new object[] { 10, 20
    }));

        Console.Read();
      }
   }
}
```

Class2.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{

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

```
    public class NewAttribute : Attribute
    {
        public NewAttribute() { }
        public NewAttribute(string description)
        {
            Description = description;
        }
        public string Description { get; set; }
    }
}
```

Class1.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Class1
    {
        //Constructors
        public Class1() { }
        public Class1(int p) { }
        public Class1(string s) { }

        //Methods
        public int Plus(int x, int y) { return x + y; }
        public int Minus(int x, int y) { return x - y; }

        //Properties
        [NewAttribute("Описание для property1")]
        public string Property1 { get; set; }
        public int Property2 { get; set; }
        [NewAttribute(Description = "Описание для property3")]
        public double Property3 { get; private set; }

        //Fields
        public int p1;
```

```
        public float p2;

    }
}
```

Примеры выполнения программы

Делегаты:

```
34,3
-14,3
34,3
-14,3
```

Рефлексия:

```
Void set_Property1(System.String)
Int32 get_Property2()
Void set_Property2(Int32)
Double get_Property3()
System.String ToString()
Boolean Equals(System.Object)
Int32 GetHashCode()
System.Type GetType()

Свойства:
System.String Property1
Int32 Property2
Double Property3

Поля данных (public):
Int32 p1
Single p2

Свойства(С атрибутом):
System.String Property1
Double Property3

Вызов метода Plus:
30
```