**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра информационной безопасности**

отчет

**по лабораторной работе №10**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: **Создание и использование делегатов**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 3363 |  | Овсейчик Н. И., Минко Д. А.,  Гончаренко О. Д. |
| Преподаватель |  | Новакова Н. Е. |

**Цель работы**

Изучение делегатов в C#, путем написания программы, которая будет имитировать действие, требующее логирования, и записывать это самое действие в текстовый файл.

**ХОД РАБОТЫ**

**Упражнение №1.**

1. Создание проекта

В среде разработки Microsoft Visual Studio был создан новый проект с именем «OopLabs.Delegates». Проект имел тип «*Console Application*».

1. Объявление делегата

Внутри класса «Program» был объявлен делегат «Log» с сигнатурой «void Log(string message)».

1. Реализация метода «DoDomething»

Реализован метод «DoSomething», принимающий делегат типа «Log» в качестве аргумента. Этот метод имитирует выполнение операции, требующей журналирования, записывая сообщение с текущей меткой времени:

1. Реализация метода «LogToFile»

Был создан метод «LogToFile» с такой же сигнатурой, как у делегата «Log». Метод выполняет запись переданного ему сообщения в текстовый файл, расположенный в папке «C:\Users\Nef0o0r\Documents\log.txt».

1. Добавление вызова «DoSomething» методе «Main»

В методе «Main» был вызван метод «DoSomething», передающий в качестве аргумента метод «LogToFile».

1. Компиляция и запуск программы

Программа была успешно откомпилирована и запущена. После выполнения программы в текстовом файле «log.txt», расположенном по пути «C:\Users\Nef0o0r\Documents\log.txt», появилась запись с текущей меткой времени.

1. Результат

Программа выполнила задачу корректно. Лог-файл «log.txt» был создан, и в нем появилась запись (рис. 1).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1 – Запись формата в файле «log.txt»

**Упражнение №2.**

1. Добавление анонимного делегата

В метод «Main» был добавлен вызов метода «DoSomething», которому передан анонимный делегат для вывода сообщения на консоль.

После компиляции и запуска программы было проверено, что сообщение записалось в файл «log.txt» и отобразилось на экране (рис. 2).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Демонстрация проверки работоспособности кода

1. Добавление вызова с лямбда-выражением

Был добавлен еще один вызов метода «DoSomething», в котором анонимный делегат был заменен на лямбда-выражение.

После компиляции и запуска программы было проверено, что сообщение записалось в файл «log.txt» и отобразилось на экране (рис. 3).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3 – Демонстрация проверки работоспособности кода

1. Замена лямбда-выражения методом «Console.WriteLine»

Был добавлен еще один вызов метода «DoSomething», в котором в качестве параметра был передан метод «Console.WriteLine» напрямую.

После компиляции и запуска программы было проверено, что сообщение записалось в файл «log.txt» и отобразилось на экране (рис. 4).

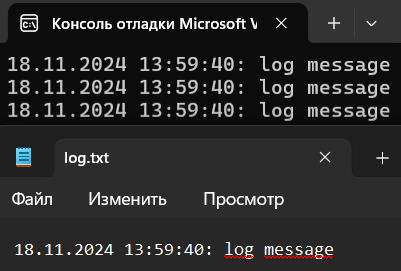


Рисунок 4 – Демонстрация проверки работоспособности кода

1. Результат

Задание выполнено в полном объеме, все функциональные возможности программы работают корректно.

**Упражнение №3.**

1. Замена типа параметра в методе «DoSomething»

Тип параметра метода «DoSomething» был изменен с пользовательского делегата «Log» на «Action<string>».

1. Удаление объявления делегата «Log»

Объявление делегата «Log» было удалено из класса «Program», так как использование общего типа «Action<string>» делает его избыточным.

1. Проверка работоспособности программы

Программа была успешно откомпилирована без ошибок. Исходный функционал остался неизменным (рис. 5).

* Метод «DoSomething» принимает «Action<string>» в качестве параметра.
* В методе «Main» различные делегаты (включая анонимные, лямбда-выражения и встроенные методы) корректно передаются в «DoSomething» и выполняют свои функции.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 5 – Демонстрация проверки работоспособности кода

ВЫВОД

По результатам выполнения работы было установлено, что использование различных подходов к реализации делегатов и их замена на тип «Action<T>» существенно упрощают код и делают его более универсальным.

В упражнении 1 реализация именованного делегата позволила управлять процессом журналирования, записывая сообщения в файл. Это продемонстрировало основы работы с делегатами и их применение для передачи методов в качестве параметров.

В упражнении 2 использование анонимных делегатов, лямбда-выражений и метода «Console.WriteLine» показало, как можно минимизировать явное объявление делегатов и повысить читаемость кода. Сообщения успешно выводились на экран и записывались в файл, демонстрируя гибкость и удобство таких подходов.

В упражнении 3 замена пользовательского делегата «Log» на общий тип «Action<string>» сделала код более компактным и универсальным. Удаление необходимости в явном объявлении сигнатур делегатов упростило процесс разработки, сохранив при этом функциональность программы.

Все задания выполнены успешно, программа корректно выполняет логирование в файл и вывод сообщений на консоль. Полученные результаты демонстрируют преимущества использования современных возможностей языка C#, включая анонимные делегаты, лямбда-выражения и типы «Action<T>» для упрощения разработки и повышения ее эффективности.

ИСХОДНЫЙ КОД

Упражнение 1:

using System;

using System.IO;

namespace OopLabs.Delegates

{

class Program

{

private delegate void Log(string message);

static void DoSomething(Log log)

{

log(DateTime.Now + ": log message");

}

static void LogToFile(string message)

{

string myDocsPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

string logFilePath = Path.Combine(myDocsPath, "log.txt");

File.AppendAllText(logFilePath, message + Environment.NewLine);

}

static void Main()

{

DoSomething(LogToFile);

Console.ReadKey();

}

}

}

Упражнение 2:

using System;

using System.IO;

namespace OopLabs.Delegates

{

class Program

{

private delegate void Log(string message);

static void DoSomething(Log log)

{

log(DateTime.Now + ": log message");

}

static void LogToFile(string message)

{

string myDocsPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

string logFilePath = Path.Combine(myDocsPath, "log.txt");

File.AppendAllText(logFilePath, message + Environment.NewLine);

}

static void Main()

{

DoSomething(LogToFile);

DoSomething(delegate (string message) { Console.WriteLine(message); });

DoSomething(message => Console.WriteLine(message));

DoSomething(Console.WriteLine);

Console.ReadKey();

}

}

}

Упражнение 3:

using System;

using System.IO;

namespace OopLabs.Delegates

{

class Program

{

static void DoSomething(Action<string> log)

{

log(DateTime.Now + ": DoSomething()");

}

static void LogToFile(string message)

{

string myDocsPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

string logFilePath = Path.Combine(myDocsPath, "log.txt");

File.AppendAllText(logFilePath, message + Environment.NewLine);

}

static void Main()

{

DoSomething(LogToFile);

DoSomething(delegate (string message) { Console.WriteLine(message); });

DoSomething(message => Console.WriteLine(message));

DoSomething(Console.WriteLine);

Console.ReadKey();

}

}

}