

Харківський національний університет радіоелектроніки

Практичні роботи №1-2

З дисципліни «Робота з даними на платформі .Net»

Виконали студенти групи (ПЗПІ-19-)-2

Селевич Олексій Вадимович

Логвінов Олександр Володимирович

Перевірила старший викладач кафедри  
ПІ

Олійник О.В.

2021

# Практична робота №1- РОЗРОБКА БІБЛІОТЕКИ КЛАСІВ ІЗ СУВОРИМ ІМ'ЯМ І ЇЇ РОЗМІЩЕННЯ В GAC

**Мета роботи:** ознайомитися з поняттями збірки , збірки з суворим ім'ям, набути практичні навички створення зборок і їх розміщення в Global Assembly Cache (GAC, глобальний кеш збірок).

**Хід роботи:**

Для виконання практичних робіт була обрана тема “Бюро знахідок”, та програмний код, створений й цій збірці, реалізує базові функції, котрі необхідні для коректної роботи бюро знахідок.

## 1)Демонстрація роботи програми:

Збірка Lost\_And\_Found\_Lib була підключена до проекту Test\_PZ1. Після запуску програми (Test\_PZ1.Program.cs) вам відкривається головне меню бюро знахідок, воно виглядає наступним чином:

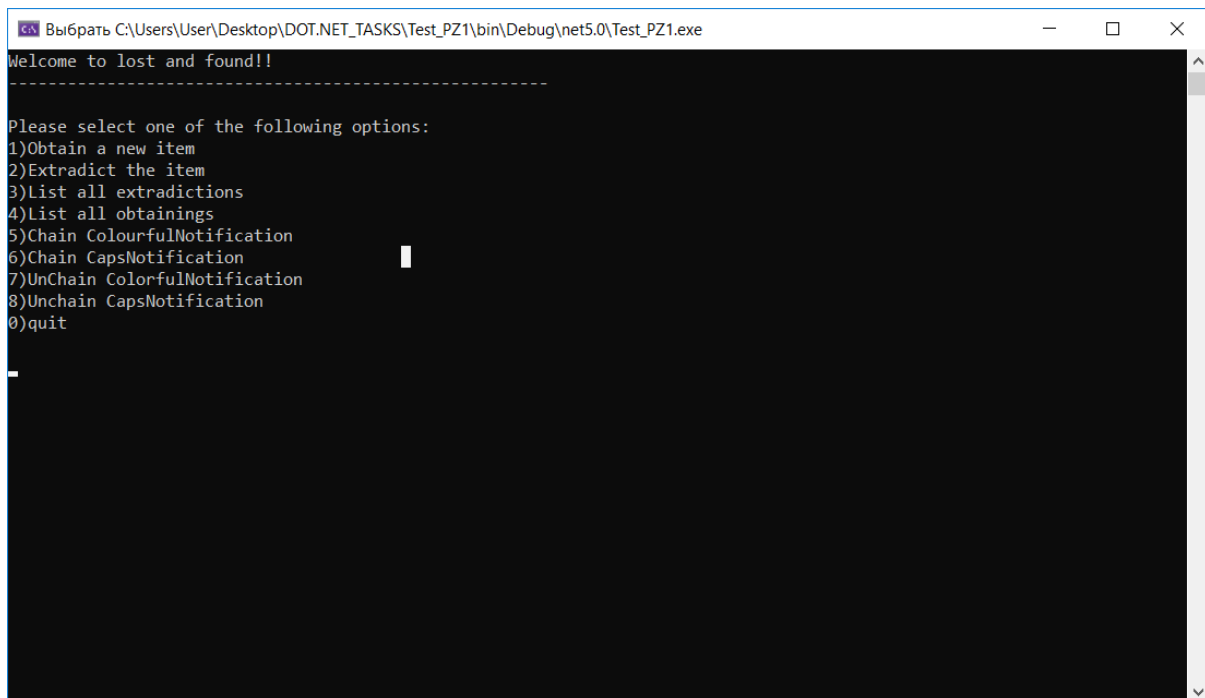


Рисунок 1- вигляд головного меню програми

У ньому ви можете додати новий предмет до сховища, видати предмет його власнику отримати список всіх знахідок вивести список усіх предметів, виданих власникам. Для демонстрації виклику ланцюжка делегатів у меню зроблені пункти 5-8. Пункти 5-6 надають можливість додати до ланцюжка виклик кольорового сповіщення, або сповіщення написанного загальними літерами. Навпаки, пункти 7-8 дозволяють видалити виклик одного з методів, котрі були додані до ланцюжка викликів (кожен виклик пунктів 5-8 додає чи віднімає лише один метод до ланцюжка викликів).

```
New iPhone 28 was brought to the lost and found at 19:02:03.0469490 by Oleksandr3 Logvinov3 and was taken to storage by Anton Mazuritskiy
The static method was invoked

Please select one of the following options:
1)Obtain a new item
2)Extradict the item
3)List all extradictions
4)List all obtainings
5)Chain ColourfulNotification
6)Chain CapsNotification
7)UnChain ColorfulNotification
8)Unchain CapsNotification
0)quit
```

Рисунок 2 - додавання нового предмету до бюро знахідок

```
The iPhone 25 was given to owner : Oleksii3 Selevych3 at 19:02:04.6525200 by Anton Mazuritskiy
The static method was invoked

Please select one of the following options:
1)Obtain a new item
2)Extradict the item
3)List all extradictions
4)List all obtainings
5)Chain ColourfulNotification
6)Chain CapsNotification
7)UnChain ColorfulNotification
8)Unchain CapsNotification
0)quit
```

Рисунок 3 - отримання предмета його власником

Things extradicted:

| Time             | Owner              | Finding   | Worker            |
|------------------|--------------------|-----------|-------------------|
| 19:04:51.3859446 | Oleksii2 Selevych2 | iPhone 28 | Artem1 Vredniy1   |
| 19:04:52.0976430 | Oleksii2 Selevych2 | iPhone 28 | Artem1 Vredniy1   |
| 19:04:52.9348986 | Oleksii Selevych   | iPhone 29 | Artem Vredniy     |
| 19:04:53.2109897 | Oleksii Selevych   | iPhone 26 | Anton Mazuritskiy |

Рисунок 4 - отримання списку отримань предметів їх власниками

| Things obtained: |                      |           |                     |
|------------------|----------------------|-----------|---------------------|
| Time             | Finder               | Finding   | Worker              |
| 19:04:52.3699585 | Oleksandr1 Logvinov1 | IPhone 24 | Anton1 Mazuritskiy1 |
| 19:04:52.5232237 | Oleksandr2 Logvinov2 | IPhone 29 | Anton1 Mazuritskiy1 |
| 19:04:52.6760561 | Oleksandr1 Logvinov1 | IPhone 25 | Artem Vredniy       |

Рисунок 5 - отримання списку всіх знаходжень

## 2)Переглянути інформацію метаданих:

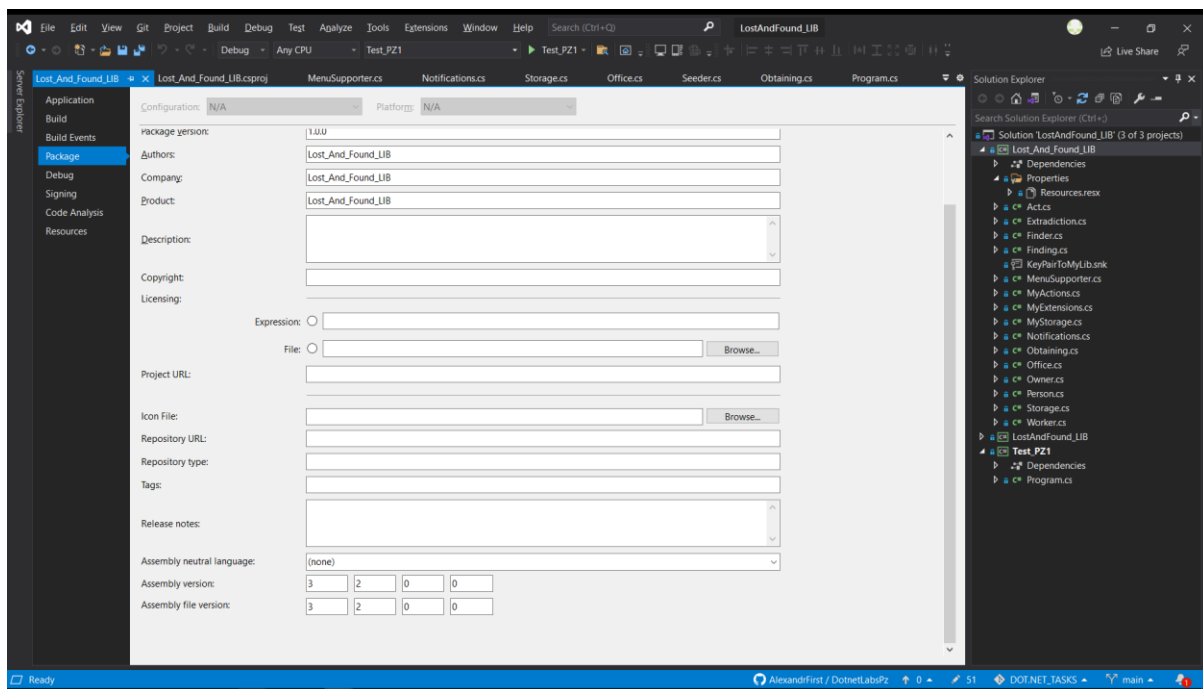


Рисунок 6 - отримані метадані збірки

Для отримання метаданих збірки у проєкті на платформі .Net Core потрібно натиснути на праву кнопки мишки, а потім натиснути на вкладку “Properties” та у вікні що відкриється обрати підпункт “Package”.

## 3)Перевірити та змінити номер версії збірки

Перевірити номер збірки можна завжди у вікні де була отримана інформація про метадані. Змінити номер збірки можна за допомогою Visual Studio та використовуючи програмний код. Ми змінили номер збірки за допомогою Visual Studio:

Assembly neutral language: (none)

Assembly version: 3 2 0 0

Assembly file version: 3 2 0 0

Рисунок 7 - попередній номер збірки

Assembly version: 4 1 0 0

Assembly file version: 3 2 0 0

Рисунок 8 - новий номер збірки

Середя розробки .NET Core автоматично не додає класс AssemblyInfo.cs до проекту, але за бажанням його можна додати та змінити номер версії завдяки використання атрибуту AssemblyVersion.

#### 4) Підписати збірку суворим ім'ям

Для підписання збірки суворим ім'ям ми використовуємо утиліту командного рядка sn.exe. Виконання наступної команди створює пару приватного та публічного ключів, які потрібні для процесу підписання збірки:

**sn.exe -k c:\KeyPairToMyLib.snk**

після виконання команди на жорсткому диску c, створюється файл з ключами. Після цього заходимо у Visual Studio та заходимо у властивості проекту, потім заходимо у підпункт “Signing” та обираємо пару ключів для підписання нашої збірки:

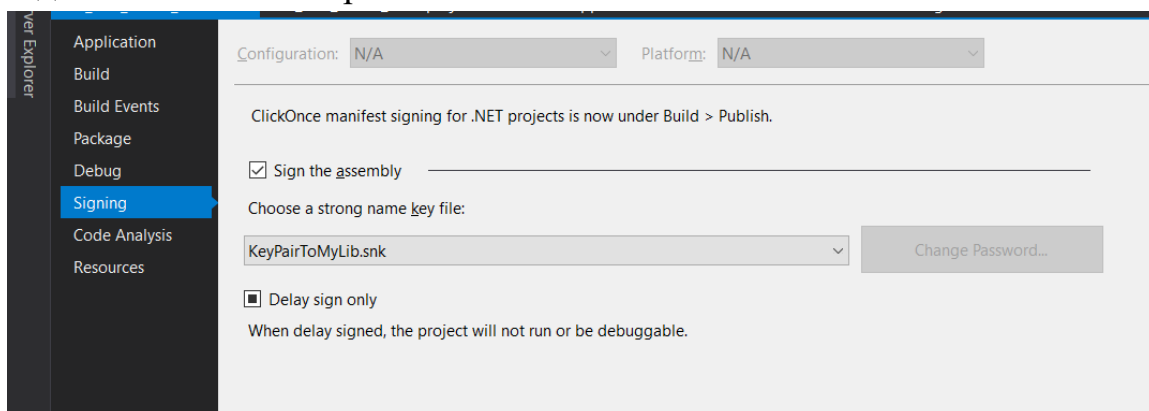


Рисунок 9 - підписання збірки за допомогою Visual Studio

Але якщо ви використовуєте .Net Framework або додали AssemblyInfo.cs до вашого проекту ви можете підписати збірку за допомогою AssemblyKeyFile атрибуту.

## 5) Розмістити збірку в GAC

Щоб розмістити збірку у global assembly cache необхідно використати утиліту командного рядку gacutil.exe. Перейдемо у директорію з файлом збірки та виконаємо наступну команду у командному рядку:

**gacutil.exe -i Lost\_And\_Found\_Lib.dll**

Після її успішного виконання збірки буде додано до GAC. Побачити це ми можемо у директорії c:\Windows\Microsoft.NET\assembly\GAC\_MSIL, усі збірки, котрі використовують версії .NET вище 3.5 будуть розміщені у цій папці.

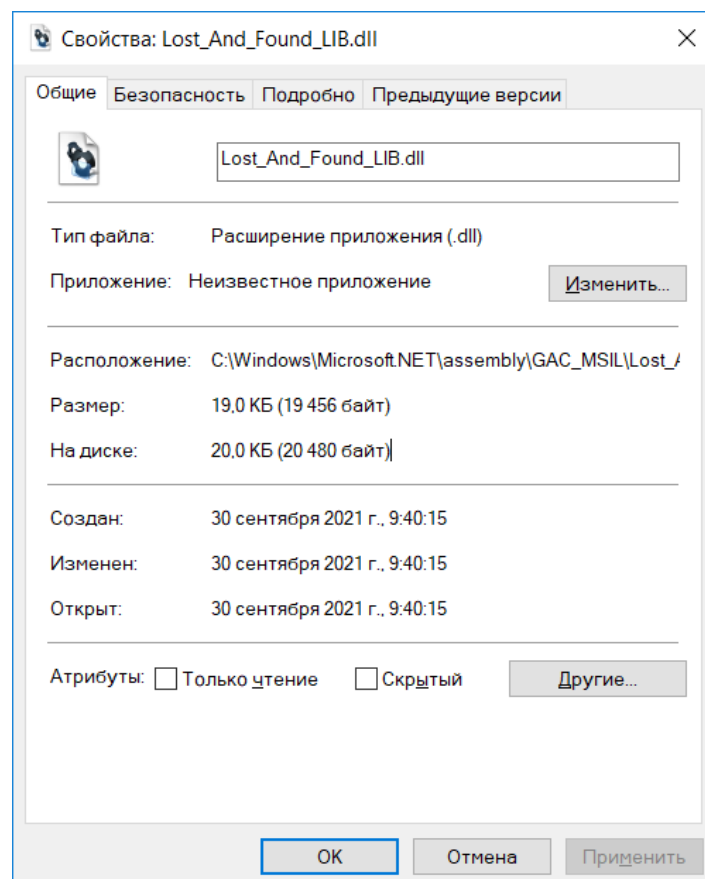


Рисунок 10 - збірка, котра була розміщена у global assembly cache

## 6)Перевірити та змінити номер версії збірки

|                        |   |   |   |   |
|------------------------|---|---|---|---|
| Assembly version:      | 5 | 1 | 0 | 0 |
| Assembly file version: | 5 | 1 | 0 | 0 |

Рисунок 11- змінений номер версії збірки

## 7)Повторно розмістити збірку в GAC та перевірити кількість збірок з однаковим ім'ям

```
C:\Windows\System32>cd C:\Users\User\Desktop\DOT.NET_TASKS\Lost_And_Found_LIB\bin\Debug\net5.0
C:\Users\User\Desktop\DOT.NET_TASKS\Lost_And_Found_LIB\bin\Debug\net5.0>gacutil.exe -i Lost_And_Found_LIB.dll
Microsoft (R) .NET Global Assembly Cache Utility. Version 4.0.30319.0
Copyright (c) Microsoft Corporation. All rights reserved.
Assembly successfully added to the cache
```

Рисунок 12 - повторне додання збірки до GAC

буфер обмена

упорядочить

Создать

Открыть

↑ > Этот компьютер > Локальный диск (C:) > Windows > Microsoft.NET > assembly > GAC\_MSIL > Lost\_And\_Found\_LIB >

bug


^

Имени

Дата изменения


Тип

Размера

 v4.0\_3.2.0.0\_eb7f3365002a3011

10.10.2021 20:09

Папка с файлами

 v4.0\_5.1.0.0\_eb7f3365002a3011

10.10.2021 20:15

Папка с файлами

Рисунок 13 - вміст директорії з різними версіями збірки

**Висновок:** ознайомилися з поняттями збірки , збірки з суворим ім'ям, набули практичні навички створення збірок і їх розміщення в Global Assembly Cache (GAC, глобальний кеш збірок).

# Практична робота №2 - СТВОРЕННЯ ДЕЛЕГАТІВ І ПОДІЙ У ВЛАСНИХ КЛАСАХ

**Мета:** ознайомитися з поняттями делегатів, набути практичні навички створення та роботи з делегатами.

**Хід роботи:**

## 1)Створити делегат:

У файлі Storage.cs був створений наступний делегат:

```
public delegate void Notify(string message);
```

## 2) Продемонструвати коваріативність та / або контрваріативність посилальних типів при прив'язці методу до делегату:

У проєкті ми маємо базовий абстрактний тип даних Person.cs. Від нього були успадковані наступні типи: Finder, Owner, Worker.

Код Person.cs:

```
public abstract class Person
{
    private int id;
    private string name;
    private string surname;
    private DateTime birthday;
    public int Id
    {
        get => id;
        set => id = value;
    }
    public DateTime Birthday
    {
        get => birthday;
        set => birthday = value;
    }
    public string Name
    {
        get => name;
        set => name = value;
    }
}
```



```

    public string Surname
    {
        get => surname;
        set => surname = value;
    }
    public Person(string Name, string Surname, DateTime Birthday,
int Id)
    {
        this.Name = Name;
        this.Surname = Surname;
        this.Birthday = Birthday;
        this.Id = Id;
    }
}

```

### **Код Owner.cs:**

```

public class Owner : Person
{
    public Owner(string Name, string Surname, DateTime Birthday,
int Id) : base(Name, Surname, Birthday, Id)
    {

    }
}

```

### **Код Worker.cs:**

```

public class Worker : Person
{
    public Worker(string Name, string Surname, DateTime Birthday,
int Id)
        : base(Name, Surname, Birthday, Id) { }
}

```

### **Код Finder.cs:**

```

    public class Finder : Person
    {
        public Finder(string Name, string Surname, DateTime Birthday,
int Id) : base(Name, Surname, Birthday, Id)
        {
        }
    }
}

```

**З метою демонстрації контрваріантності делегатів ми створили наступний делегат, котрий приймає об'єкт типу Person, але ми передаємо в нього**

об'єкти похідних типів від Person. Він повертає ім'я того об'єкту котрий був переданий у нього.

```
public delegate string GetPersonFullName(Person person);
```

Для прикладу продемонструємо виклик делегату під час виконання методу Extradict класу Storage.cs:

```
NotifyEvent?.Invoke($"The {finding.Name} was given to owner :  
{GetFullName?.Invoke(owner)} at {actTime.TimeOfDay}" +  
    $" by {GetFullName(worker)}");
```

### **3) Продемонструвати використання делегатів для виклику статичних і екземплярних методів:**

Щоб продемонструвати вам роботу статичних та екземплярних методів при викликанні статичних методів була перевірка властивості у делегата. Перевірялось значення Target та якщо воно є рівним null, то здійснюється виклик статичного методу, а якщо метод є екземплярним, то Target буде мати посилання на об'єкт метод якщо буде використовуватися.

Метод GetPersonFullName(Person p), дійсно є статичним та розташований у статичному класі MyActions.cs.

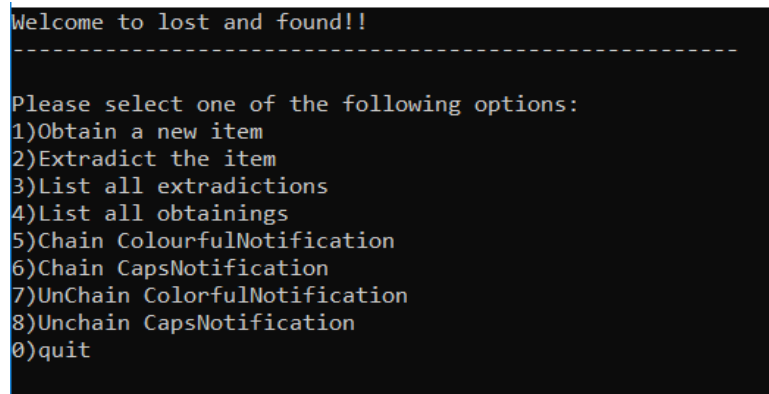
Код методу GetPersonFullName(Person p):

```
public static string GetPersonFullName(Person person) =>  
    person.Name + " " + person.Surname;
```

Наступним чином здійснюється перевірка на те є статичним метод, чи ні:

```
if (GetFullName.Target == null)  
    Console.WriteLine("The static method was invoked");
```

### **4)Створити ланцюжок делегатів. Додати , видалити ...**



```
Welcome to lost and found!!  
-----  
  
Please select one of the following options:  
1)Obtain a new item  
2)Extradict the item  
3)List all extraditions  
4)List all obtainings  
5)Chain ColourfulNotification  
6)Chain CapsNotification  
7)UnChain ColorfulNotification  
8)Unchain CapsNotification  
0)quit
```

## Рисунок 1 - меню програми

У меню додатку підпункти 5-8 відповідають за додання та видалення методів з ланцюжку делегатів. За це відповідають наступні методи:

```
public static void AddColourfulMessageNotification<T>(T param) where
T : Office
{
    Notifications notifications =
Notifications.AddColourfulNotification();
    param.Storage.NotifyEvent +=
notifications.ColourfulMessage;
    Console.WriteLine("You have successfully chained a
ColorfulMessage method");
}
public static void AddCapsMessageNotification<T>(T param)
where T : Office
{
    Notifications notifications =
Notifications.AddCapsNotification();
    param.Storage.NotifyEvent += notifications.CapsMesssage;
    Console.WriteLine("You have successfully chained a
CapsMessage method");
}
public static void RemoveColourfulMessageNotification<T>(T
param) where T : Office
{
    Notifications notifications =
Notifications.RemoveColourfulNotification();
    if (notifications == null)
        return;
    param.Storage.NotifyEvent -=
notifications.ColourfulMessage;
    Console.WriteLine("You have successfully unchained a
ColorfulMessage method");
}
public static void RemoveCapsMessageNotification<T>(T param)
where T : Office
{
    Notifications notifications =
Notifications.RemoveCapsNotification();
    if (notifications == null)
        return;
    param.Storage.NotifyEvent -= notifications.CapsMesssage;
    Console.WriteLine("You have successfully unchained a
CapsMessage method");
}
```

У нас є подія, котра відповідає за надсилання повідомлень користувачу про дії, котрі виникають у нашому додатку, тому завдяки підпунктам меню 5-6 ми можемо або додати метод до ланцюжку виклику, або прибрати його звідси.

Наприклад додамо два кольорових повідомлення та одне з загальними літерами.

```
New iPhone 22 was brought to the lost and found at 22:24:18.9824719 by Oleksandr3 Logvinov3 and was taken to storage by Anton1 Mazuritskiy1
New iPhone 22 was brought to the lost and found at 22:24:18.9824719 by Oleksandr3 Logvinov3 and was taken to storage by Anton1 Mazuritskiy1
New iPhone 22 was brought to the lost and found at 22:24:18.9824719 by Oleksandr3 Logvinov3 and was taken to storage by Anton1 Mazuritskiy1
NEW IPHONE 22 WAS BROUGHT TO THE LOST AND FOUND AT 22:24:18.9824719 BY OLEKSANDR3 LOGVINOV3 AND WAS TAKEN TO STORAGE BY ANTON1 MAZURITSKIY1
The static method was invoked

Please select one of the following options:
1)Obtain a new item
2)Extradict the item
3)List all extradictions
4)List all obtainings
5)Chain ColourfulNotification
6)Chain CapsNotification
7)UnChain ColorfulNotification
8)Unchain CapsNotification
0)quit
```

Рисунок 2 - приклад додання методів до ланцюжку

Зараз видалимо одне кольорове повідомлення та додамо ще одне з загальними літерами.

```
The iPhone 23 was given to owner : Oleksii1 Selevych1 at 22:25:50.6540550 by Anton Mazuritskiy
The iPhone 23 was given to owner : Oleksii1 Selevych1 at 22:25:50.6540550 by Anton Mazuritskiy
THE IPHONE 23 WAS GIVEN TO OWNER : OLEKSII1 SELEVYCH1 AT 22:25:50.6540550 BY ANTON MAZURITSKIY
THE IPHONE 23 WAS GIVEN TO OWNER : OLEKSII1 SELEVYCH1 AT 22:25:50.6540550 BY ANTON MAZURITSKIY
The static method was invoked

Please select one of the following options:
1)Obtain a new item
2)Extradict the item
3)List all extradictions
4)List all obtainings
5)Chain ColourfulNotification
6)Chain CapsNotification
7)UnChain ColorfulNotification
8)Unchain CapsNotification
0)quit
```

Рисунок 3 - інший приклад додання методів до ланцюжка

## 5)Продемонструвати використання узагальнених делегатів і власних

У бібліотеці використовується наступний узагальнений делегат:  
**public delegate void MenuAction<T>(T param) where T : Office;**

Він забезпечує коректну роботу головного меню додатку, та потім завдяки ньому у програмі здійснюється виклик одного з наступних методів:

```

public static void ObtainAction<T>(T param) where T : Office
{
    param.Storage.Obtain(DateTime.Now,

    param.MyStorage.Workers.GetRandom(),

    param.MyStorage.Findings.GetRandom(),

    param.MyStorage.Finders.GetRandom());
}
public static void ExtradictAction<T>(T param) where T: Office
{
    param.Storage.Extradict(DateTime.Now,

    param.MyStorage.Workers.GetRandom(),

    param.MyStorage.Findings.GetRandom(),

    param.MyStorage.Owners.GetRandom());
}
public static string GetPersonFullName(Person person) =>
    person.Name + " " + person.Surname;
public static void DisplayAllObtainings<T>(T param) where T :
Office
{
    Console.WriteLine("Things obtained: ");
    Console.WriteLine("          Time          |          Finder
| Finding      | Worker      ");
    Console.WriteLine("-----
-----");
    foreach(var obtainedThing in param.Storage.Obtainings)
    {

        Console.WriteLine("{0}|{1}|{2}|{3}",obtainedThing.ActTime.Time
OfDay.ToString().FitWithLength(),

        GetPersonFullName(obtainedThing.Finder).FitWithLength(),

        obtainedThing.Finding.Name.FitWithLength(),

        GetPersonFullName(obtainedThing.Worker).FitWithLength());
    }
}
public static void DisplayAllExtradictions<T>(T param) where
T: Office
{
    Console.WriteLine("Things extradicted: ");
    Console.WriteLine("          Time          |          Owner
| Finding      | Worker      ");
    Console.WriteLine("-----
-----");
    foreach (var obtainedThing in
param.Storage.Extradictions)
    {

```

```

        Console.WriteLine("{0}|{1}|{2}|{3}",
obtainedThing.ActTime.TimeOfDay.ToString().FitWithLength(),

        GetPersonFullName(obtainedThing.Owner).FitWithLength(),

        obtainedThing.Finding.Name.FitWithLength(),

        GetPersonFullName(obtainedThing.Worker).FitWithLength());
    }
}

public static void AddColourfulMessageNotification<T>(T
param)where T :Office
{
    Notifications notifications =
Notifications.AddColourfulNotification();
    param.Storage.NotifyEvent +=
notifications.ColourfulMessage;
    Console.WriteLine("You have successfully chained a
ColorfulMessage method");
}

public static void AddCapsMessageNotification<T>(T param)
where T : Office
{
    Notifications notifications =
Notifications.AddCapsNotification();
    param.Storage.NotifyEvent += notifications.CapsMessage;
    Console.WriteLine("You have successfully chained a
CapsMessage method");
}

public static void RemoveColourfulMessageNotification<T>(T
param) where T : Office
{
    Notifications notifications =
Notifications.RemoveColourfulNotification();
    if (notifications == null)
        return;
    param.Storage.NotifyEvent -=
notifications.ColourfulMessage;
    Console.WriteLine("You have successfully unchained a
ColorfulMessage method");
}

public static void RemoveCapsMessageNotification<T>(T param)
where T : Office
{
    Notifications notifications =
Notifications.RemoveCapsNotification();
    if (notifications == null)
        return;
    param.Storage.NotifyEvent -= notifications.CapsMessage;
    Console.WriteLine("You have successfully unchained a
CapsMessage method");
}
}
}

```

## 6)Продемонструвати роботи з подіями

Як вже було сказано раніше, усі повідомлення для користувача відправляються за допомогою виникнення наступної події :

**public event Notify NotifyEvent;**

**storage.NotifyEvent += notifications.TextMessage;**

За допомогою другої лінії коду ми автоматично додаємо звичайне текстове повідомлення у ланцюжок виклику делегатів події та таким чином забезпечуємо, що користувач гарантовано отримує повідомлення. Результат роботи програми був детально продемонстрований у пункті 4.

**Висновки:** ознайомилися з поняттями делегатів, набули практичні навички створення та роботи з делегатами.