



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Финализация. Деструктори. Ръчно управление на паметта. Ефективно управление на паметта

Freachable & IDisposable

Съдържание

- Какво е Финализацията?
- Деструктори
- Особености на финализацията
- Ръчно управление на паметта
- Взаимодействие със системата за почистване на паметта (СПП)
- Ефективно управление на паметта
- Техниката "пулинг на ресурси"

Какво е финализация?

- Тя е указание към CLR
 - “преди този обект да бъде унищожен, трябва да се изпълни ето този код”.
- класът трябва да имплементира метод `Finalize()`.
- Когато garbage collector установи, че даден обект вече не се използва от приложението, той проверява дали обектът дефинира `Finalize()` метод и ако е така:
 1. При първото преминаване се установява че обектът подлежи на унищожение и се изпълнява финализаторът,
 2. а при второто се освобождава и заетата от обекта памет.
- `Finalize()` не може да се извиква явно.
- Най-малко **две** преминавания на garbage collector са необходими за да се унищожи обект, дефиниращ `Finalize()` метод. поради преминаването на обекта в по-горно поколение.

Деструктори

- в C# не може Finalize() да се имплементира явно. Вместо това се използват деструктори, които имат следния специален синтаксис:

```
~MyClass()  
{  
    // Тук е кода за почистването  
}
```

Запомнете: общото между деструкторите в C# и тези в C++ се изчерпва със синтаксиса.

- Унищожаването на обектите не е детерминистично и програмистът няма възможност да определи кога и в какъв ред се изпълняват финализаторите. При някои специални обстоятелства дори няма гаранция, че те изобщо ще се изпълнят

Финализация – пример

В кода показан по-долу, дефинираме клас, който капсулира някакъв Windows ресурс (манипулатор към който се съхранява в член-променливата [ResourceHandle](#)):

```
using System;
// Wrapper around Windows resource
class ResourceWrapper
{
    private IntPtr mResourceHandle =
        IntPtr.Zero;
    public ResourceWrapper()
    {
        // Заделяне на ресурс тук
    }
}
```

```
~ResourceWrapper()
{
    if (mResourceHandle != IntPtr.Zero)
    {
        // Освобождаване на ресурс тука
        //... mResourceHandle = IntPtr.Zero;
    }
}
```

Опашката Freachable

- Създаването на обект, поддържащ финализация изисква **поставянето на указател във Finalization list** и отнема малко повече време
- **Опашката Freachable** съдържа указатели към всички обекти, чиито **Finalize()** методи вече могат да се извикат. Всеки обект в опашката е **достъпен от приложението и не е отпадък**. Т.е. Опашката Freachable се счита за **част от корените** на приложението

Особности на финализацията

- Финализацията е неефективна
- Проблеми с нишките
- Проблеми с реда на изпълнение на финализаторите
- Не разчитайте само на финализацията за да освобождавате ресурси.
Имплементирайте `IDisposable` и използвайте `Finalize()` методите съвместно с него.

Ръчно управление на ресурсите с IDisposable

Интерфейсът **IDisposable** се препоръчва от Microsoft в тези случаи, в които искате да гарантирате моментално освобождаване на ресурсите (вече знаете, че използването на **Finalize()** не го гарантира)

```
public interface IDisposable
{
    void Dispose();
}
```



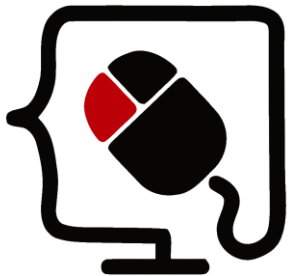
Ето как изглежда
този интерфейс:

Взаимодействие със системата за почистване на паметта

- Помогнете на GC като не ѝ помагате
- Почистването на поколение 0 се случва достатъчно често и е сравнително “евтино”
- Дефинирайте финализатор само ако класът ви обгръща неуправляван ресурс!
- Не създавайте обекти без да е необходимо
- Не създавайте обекти с излишни полета
- Не инициализирайте полетата в конструкторите
- Не проектирайте излишно дълбоки йерархии
- Заделете цялата памет, нужна за създаването на структура от данни, наведнъж
- Избягвайте използването на отражение на типовете (reflection), Използвайте възможно най-малко финализатори

Техниката "пулинг на ресурси"

- **Пулът** обикновено представлява списък от обекти, които се създават предварително (например при инициализация на приложението), а после се "раздават" при поискване.
- Клиентите взимат обекти от пула, използват ги известно време и след като вече не им трябва, не ги унищожават, а ги връщат обратно в пула.



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН) и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).