**Тема:**

**Финализации и деструктори**

От:

**Симеон М.**

Клас:

**11А**

1. **Какво са финализаторите ?**

* Финализаторите позволяват на разработчиците да пишат код, който ще изчисти ресурсите на класа. За разлика от конструкторите, които се извикват изрично с помощта на new операторa, финализаторите не могат да бъдат извикани изрично чрез кода. Няма еквивалент, като например оператор за изтриване. По-скоро **garbage collector-а е отговорен за извикването на финализатор на инстанция на обект.** Следователно разработчиците не могат да определят по време на компилация точно кога ще се изпълни финализаторът. Всичко, което знаят, е, че финализаторът ще работи някъде между момента, в който обектът е бил използван за последен път, и когато приложението се изключи нормално. (Финализаторите може да не се изпълнят, ако процесът бъде прекратен необичайно. Например, събития като изключване на компютъра или принудително прекратяване на процеса ще попречат на изпълнението на финализатора).
* Финализаторите не позволяват да се подават никакви параметри, така че те не могат да бъдат претоварени. Освен това финализаторите не могат да бъдат извикани изрично – т.е. само garbage collector-a може да извика финализатор. Следователно модификаторите за достъп на финализаторите са безсмислени и като такива те не се поддържат. Финализаторите в base класовете ще бъдат извикани автоматично като част от извикване за финализиране на обект.
* Тъй като **garbage collector-а обработва цялото управление на паметта**, финализаторите не са отговорни за деразпределянето на паметта. По-скоро те са отговорни за освобождаването на ресурси като връзки с бази данни и файлови манипулатори – ресурси, които изискват изрична дейност, за която garbage collector-а не знае.
* Финализаторите изпълняват на своя собствена нишка, което прави изпълнението им още по-малко детерминистично. Този неопределен характер прави необработеното изключение в рамките на финализатора (извън дебъгера) трудно за диагностициране, тъй като обстоятелствата, довели до изключението, не са ясни. От гледна точка на потребителя, необработеното изключение ще бъде хвърлено сравнително случайно и с малко внимание към всяко действие, което потребителят е извършил. Поради тази причина трябва да се погрижите да избегнете изключения в рамките на финализаторите. Вместо това трябва да използвате техники като **defensive programming**, като например проверка за нули

**Синтаксис на финализатор:**

**A screen shot of a computer

Description automatically generated**

* Не трябва да се използват празни финализатори. Когато един клас съдържа финализатор, се създава запис в Finalize Queue. Тази опашка се обработва от garbage collector-а. Когато GC обработва опашката, той извиква всеки финализатор. Ненужните финализатори, включително празни финализатори, финализатори, които извикват само финализатора на базовия клас, или финализатори, които извикват само условно излъчени методи, причиняват ненужна загуба на производителността.

1. **Какво представлява Dispose метода**
   * NET framework улеснява garbage collector-а (GC), управлява паметта и ресурсите на обектите и възстановява невалидните препратки към паметта на обектите, като извиква Finalize – недетерминистичен метод. Методът Dispose контролира живота на инстанциите на обектната памет и осигурява изричен контрол за почистване на паметта, в сравнение с изчистването на имплицитната памет на Finalize. Dispose може да бъде извикан дори когато съществуват други инстанции на обекти на паметта, докато Finalize може да бъде извикан само след като последният обект на паметта е унищожен.
2. **Правила при използването на Dispose метода**

* Използва се за неуправлявани ресурси, изискващи незабавно освобождаване след употреба.
* Ако метода Dispose не бъде извикан, трябва да се приложи методът Финализиране.
* След извикване на метода Dispose, GC. Методът SuppressFinalize трябва да бъде извикан, за да се предотврати методът Финализиране и да се избегне ненужното GC.
* Изключенията трябва да се обработват внимателно, ако методът Dispose се използва повече от веднъж. Ако ресурсите са разположени, всеки метод на инстанцията може да хвърли **ObjectDisposedException**.
* Обект с по-рано наречен метод Dispose не може да се използва повторно.
* Dispose се препоръчва само за управление на основни ресурсни обекти и Component Object Model (COM) обекти, изложени на .NET Framework.
* Изхвърлянето може да не се извиква едновременно от множество нишки, поради непредвидими резултати.
* Типовете стойности не трябва да се създават като типове за еднократна употреба или с неуправлявани членове на ресурси
* Когато използвате неуправлявани ресурси, се счита за най-добра практика да приложите командата Using на изходния код, която автоматично извиква метода Dispose на обекта след завършване на обектния код.

**A screenshot of a computer program

Description automatically generated Синтаксис на метод Dispose:**

1. **Деструктори**

* Деструкторите се използват за унищожаване на инстанции на класове, когато вече не са необходими. Деструкторът се извиква изрично от Garbage Collector-а на .NET Framework. Променлива на инстанция или обект отговаря на условията за унищожаване, когато вече не е достижим.
* Деструкторът е уникален за своя клас, т.е. не може да има повече от един разрушител в един клас.
* Деструкторът няма тип на връщане и има точно същото име като името на класа (включително същия случай).
* Той се отличава от конструктора поради символа Tilde (~), предшестващ името му.
* Деструкторът не приема никакви параметри и модификатори.
* Тя не може да бъде дефинирана в структурите. Използва се само с класове.
* Тя не може да бъде претоварена или наследена.

**Синтаксис:**

A screenshot of a computer code

Description automatically generated

1. **Разлика между финализатори, деструктори и метод Dispose**
2. **Деструктор -** Destructor ще бъде написан в клас, за да почисти паметта, използвана от инстанциите на този клас. Деструктор не може да бъде изрично извикан в C#. Той ще бъде извикан от GC процес, докато събира боклука. Те са специални методи, които съдържат код за почистване на обекта. Не можете да ги извикате изрично във вашия код, тъй като те се наричат имплицитно от GC (Garbage Collector). В C# те имат същото име като името на класа, предшествано от знака "~".
3. **Dispose** - Dispose метод Трябва да се извика изрично по всяко време, точно както всеки друг метод. Съдържа кода за изчистване на неуправлявания код, достъпен от обекта. Те са точно като всички други методи в класа и могат да бъдат наречени изрично, но те имат специална цел за почистване на обекта. В метода dispose пишем clean up код за обекта. Важно е да освободим всички неуправлявани ресурси в метода dispose като връзка с база данни, файлове и т.н.
   1. Dispose() се извиква от потребителя
   2. Същата цел като финализиране, за освобождаване на неуправлявани ресурси. Въпреки това, реализирайте това, когато пишете персонализиран клас, който ще се използва от други потребители.
   3. Заместването на Dispose () предоставя начин потребителският код да освободи неуправляваните обекти във вашия потребителски клас.
   4. Методът Dispose може да бъде извикан само от класовете, които имат IDisposable интерфейс.
4. **Финализатор -** Метод за финализиране е кодът за изчистване на паметта, използвана от класа. Метод за финализиране може да бъде извикан изрично с помощта на "objectname". Финализиране ()" синтаксис
   1. Внедрете го, когато имате неуправлявани ресурси в кода си и искате да сте сигурни, че тези ресурси са освободени, когато се случи събирането на боклука.
   2. Финализаторите трябва да освобождават само неуправлявани ресурси.
   3. Финализаторите винаги трябва да бъдат защитени, а не публични или частни, така че методът да не може да бъде извикан директно от кода на приложението и в същото време да може да направи повикване към базата. Финализиране на метод