**Garbage Collection**

**Plan:**

* GC – nin necə işlədiyini öyrənmək;
* GC – nin hansı hallarda işə düşdüyünü anlamaq;
* GC – nin optimallaşdırılması və performans təsirlərini bilmək;
* Dispose, Finalizer və IDisposable arasındakı fərqləri bilmək;
* GC ilə bağlı müsahibə sualları

**Garbage Collector**

**GC (Garbage Collector)**, **C# və .NET**-də **istifadə olunmayan obyektlərin** avtomatik olaraq **RAM-dan silinməsi üçün nəzərdə tutulmuş mexanizmdir**. Əgər GC olmasaydı, **C++** kimi **manual memory management** etməli olardıq (malloc/free, new/delete).

**Əsas məqsədi:**

* Lazımsız obyektləri silib RAM – ı boşaltmaq;
* Memory Leak – ların qarşısını almaq;
* Manual memory management yükünü developerin üzərindən götürmək.

**Memory Leak –** yaddaş sızması, proqramın istifadə etdiyi amma geriqaytarmadığı hallar nəticəsində zamanla yaddaşın dolmasıdır. Bu istifadəsiz obyektlərin yaddaşda qalmasından yaranır və sistemi yavaşladır.

**Nə vaxt Memory Leak ola bilər?**

* **Obyektlər səhvən referansda qalarsa.**  
  Məsələn, **Dispose** metodunu çağırmadıqda və ya **GC** tərəfindən silinməyən obyektlər varsa.
* **Unmanaged resursları düzgün sərbəst buraxmasaq.**  
  Məsələn, faylları, əlaqələri və ya əməliyyat sisteminin digər resurslarını sərbəst buraxmaq vacibdir.
* **Yanlış metodlar istifadə edildikdə.**  
  Məsələn, **GC.Collect()** metodu düzgün yerə qoyulmadıqda, **manual olaraq** yaddaşın düzgün idarə olunmaması.

**Memory Leak-i necə qarşısını almaq olar?**

1. **Dispose() və IDisposable istifadə et.**  
   Həm **managed**, həm də **unmanaged** resursları sərbəst buraxmaq üçün müvafiq olaraq Dispose() metodunu çağırmalısınız.
2. **Obyektləri düzgün idarə et.**  
   Obyektlər **işiniz bitdikdə** onları **null** etmək və ya Dispose() çağırmaq lazımdır. **using** blokları ilə də bu işi asanlaşdıra bilərik.
3. **Finalizer və GC.SuppressFinalize() istifadə et.**  
   Obyektinizin **finalizer** metodunu çağıraraq, istifadə edilməyən obyektlərin **garbage collector tərəfindən təmizlənməsini təmin edirsiniz.**

**Garbage Collector necə işləyir?**

**GC əsas 3 mərhələdən ibarətdir:**

1. **Mark – Hansı obyektlərin hələ də istifadə edildiyini müəyyən edir;**
2. **S**weep – İstifadə edilməyən obyektləri silir;
3. Compact – Fraqmentasiyanı lazım gələrsə, aradan qaldırır.

### ****GC-də Compact nədir?****

**Problem:** GC **obyektləri siləndə** yaddaşda **boşluqlar (fragmentlər)** yaranır. Yəni,

* **Məsələn:** Əgər 4 obyekt vardısa və GC onların 2-ni sildisə, indi yaddaşda boş yerlər var.
* Amma yeni obyektlər eyni ölçüdə olmaya bilər. Bu isə **yaddaşın parçalanmasına (fragmentasiyaya)** səbəb olur.

**Həll:**

* **Compact mərhələsi** silinmiş obyektlərin **boşluqlarını** yenidən düzür.
* Yəni **mövcud obyektləri sıxlaşdırır**, fraqmentasiyanı aradan qaldırır.

**Nəticə:**

* Yaddaş **daha səmərəli** istifadə olunur.
* Yeni obyektlər **böyük boşluqlara ehtiyac olmadan** rahat yerləşdirilir.
* Performance artır.

.NET generational model istifadə edir:

|  |  |
| --- | --- |
| **Generation** | **Xüsusiyyəti** |
| **Gen 0** | **Yeni obyektlər buraya düşür. Tez – tez silinir.** |
| **Gen 1** | **Gen 0 – dan sağ çıxan obyektlər buraya düşür.** |
| **Gen 2** | **Uzun ömürlü obyektlər (singleton, static) buraya düşür, ən gec təmizlənir.** |

**Gen 0 -> Gen 1 -> Gen 2 istiqamətində obyektlər “yaşadıqca” irəliləyir və GC daha onlara az – az toxunur.**

**Dispose, Finalizer və IDisposable**

GC avtomatik işlədə də, bəzi resursları manual azad etməliyik, məsən,

* File
* Stream
* Database Connection
* Socket
* Və s.

Bunun üçün əsas 3 metod var:

1. Dispose – Idisposable interfeysini implement edib, using blokunda avtomatik çağırılan metoddur.
2. Finalizer(ClassName) – Destruktor funksiyası kimidir, amma GC çağırır.
3. GC.SuppressFinalize() – Finalizer çağırılmasının qarşısını alır.

|  |  |  |
| --- | --- | --- |
| **Metod** | **Nə edir?** | **Nə vaxt çağırılır?** |
| **Dispose()** | Resursları manual sərbəst buraxır | Developer özü çağırır (using və ya Dispose()) |
| **Finalizer(ClassName)** | GC tərəfindən avtomatik çağırılır | GC obyektin ölü olduğunu anlayanda |
| **GC.SuppressFinalize()** | Finalizer çağırılmasını deaktiv edir | Dispose() içində çağırılır |

**Məşhur interview sualları:**

1. **GC nədir və niyə lazımdır?**
2. **Hansı obyektlər Garbage Collector tərəfindən silinmir?**
3. **Dispose və Finalizer arasındakı fərq nədir?**
4. **IDisposable interfeysinin məqsədi nədir?**
5. **Garbage Collector necə işləyir və neçə mərhələsi var?**
6. **Hansı halda GC.Collect() çağırmaq düzgün olar?**

1. Garbage Collector lazımsız və uzun müddət işləyənməyən obyektlərin silinməsi və RAM-ın boşaldılması üçün hazırlanmış mexanizmdir. Onun lazımlılığı yaddaşda artıq obyektlərin qalmaması, memory-leak yaranmaması və memory management manual olaraq developerin üzərinə yüklənməməsi üçündür.

2. Tez - tez istifadə edilən, uzunömürlü (static, singleton və s.) olduğu müəyyən edilmiş fayllar. Bir də silinmənin qarşısı alınmış obyektlər.

3. Dispose manual olaraq developer tərəfindən çağırılır, interfeys ilə tətbiq edilir, resursları azad buraxır. Finalizer isə GC tərəfindən çağırılır, çağırılıb çağırılmasına hər zaman əminlik olmur, əgər gc useless obyekt kimi tanıyarsa, o zaman çağırır.

4. IDisposable interfeysinin məqsədi garbage collectorun işini yerinə yetirmək üçün tətbiq edilməkdə hazır olan interfeysdir. Manual olaraq çağırılsa da, əvvəlcədən təyin olunduğu üçün nisbətən rahatlıq yaradır. Məlum olduğu kimi interfeys pattern ilə işləmək daha avtomatlışdırılmış və elastik kodlaşdılmadır.

5. Garbage Collector 3 mərhələ işləyir. İlk olaraq Mark silinməməsi gərəkən və hələ də istifadə edilməkdə olan obyektləri təyin edir. Daha sonra Sweep - useless obyektləri silir. Son olaraq əgər lazım gələrsə Compact fraqmentasiyaları azaldır. (Compactı daha detallı izah et, zəhmət olmasa. Beynimdə qaranlıq qalan məqamları var). .NET isə bunun üçün generation istifadə edir. Gen 0 - yeni yaranan obyektlər buraya düşür, tez tez silinir. Gen 1 - Gen 0 - dan sağ çıxanlar buraya düşür. Gen 2 isə uzunpmürlü obyektləri özündı saxlayır, az silinir. Gen 0 - Gen 2 istiqamətində obyektlər irəlilədikcı gc onlara daha az toxunur.

6. Son olaraq, GC.Collect() ən çox stream, file, database connection, socket və s. istifadəsi zamanı çağırmaq lazmdlr.