Garbage Collection Teknolojisi Ve Java'da Uygulanması

- AD SOYAD: ZİKRULLAH CELEP
- BİLGİSAYAR MÜHENDİSLİĞİ 3. SINIF
- OKUL NO: 21360859066

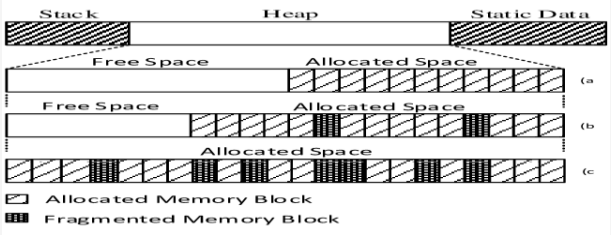
- TARİH: 18.04.2024

İçindekiler:

1. Bellek Yönetimi Sorunları
2. Garbage Collection Nedir
3. Garbage Collection Gelişimi
4. Garbage Collection Çalışma Mantığı
5. Kullanılan Çöp Toplama Algoritmaları ve Yaklaşımları
6. Javada Bellek Yapısı
7. Java’da Garbage Collection Uygulanması
   - Minor Garbage Collection
   - Major Garbage Collection
8. Javada Farklı Garbage Collection Türleri
9. Bu Teknolojinin Yazılım Güvenliğine Olan Etkileri ve Dezavantajları

Bellek Yönetimi Sorunları

- **Bellek Sızıntıları :**
  - Programın kullanmadığı ama hala bellekte yer işgal eden nesnelerin oluşturduğu durumdur.
  - Bellek sızıntıları zamanla birikebilir ve programın performansının düşmesine ve hatta sistemin kilitlenmesine yol açabilir.
- **Bellek Parçalanması :**



  - Bellek alanının küçük kullanılamaz parçalara bölünmesi durumudur.
  - Bellek parçalanması sık sık bellek tahsisi ve serbest bırakma işlemlerinin yapılmasıyla ortaya çıkabilir.

Garbage Collection Nedir ?

- Normalde programcılar programın kullandığı belleğin yönetiminden sorumludurlar.
- Kullanılan belleğin el ile temizlenmesi unutulursa bu bellek sorunlarına yol açabilir.
- Garbage collection yüksek seviyeli dillerde otomatik bellek yönetimini sağlar.
- GC programın kendisi tarafından kullanılan bellek alanını takip eder ve artık kullanılmayan nesneleri otomatik olarak siler.
- Bu bellek sızıntılarını ve diğer bellek hatalarını önler.

Garbage Collection Gelişimi

- **1959:** John Backus, Lisp programlama dili için ilk Garbage Collection algoritmasını geliştirir.
- **1970'ler:** Ada programlama dili, Garbage Collection'ı bir standart olarak benimser.
- **1980'ler:** Generational Garbage Collection gibi daha gelişmiş algoritmalar geliştirilir.
- **1990'lar:** Java ve C# gibi programlama dilleri, Garbage Collection'ı bellek yönetimi tekniği olarak benimser.
- **2000'ler:** Garbage Collection, web tarayıcıları ve sanal makineler gibi karmaşık uygulamalarda yaygın olarak kullanılır.

Garbage Collection Çalışma Mantığı

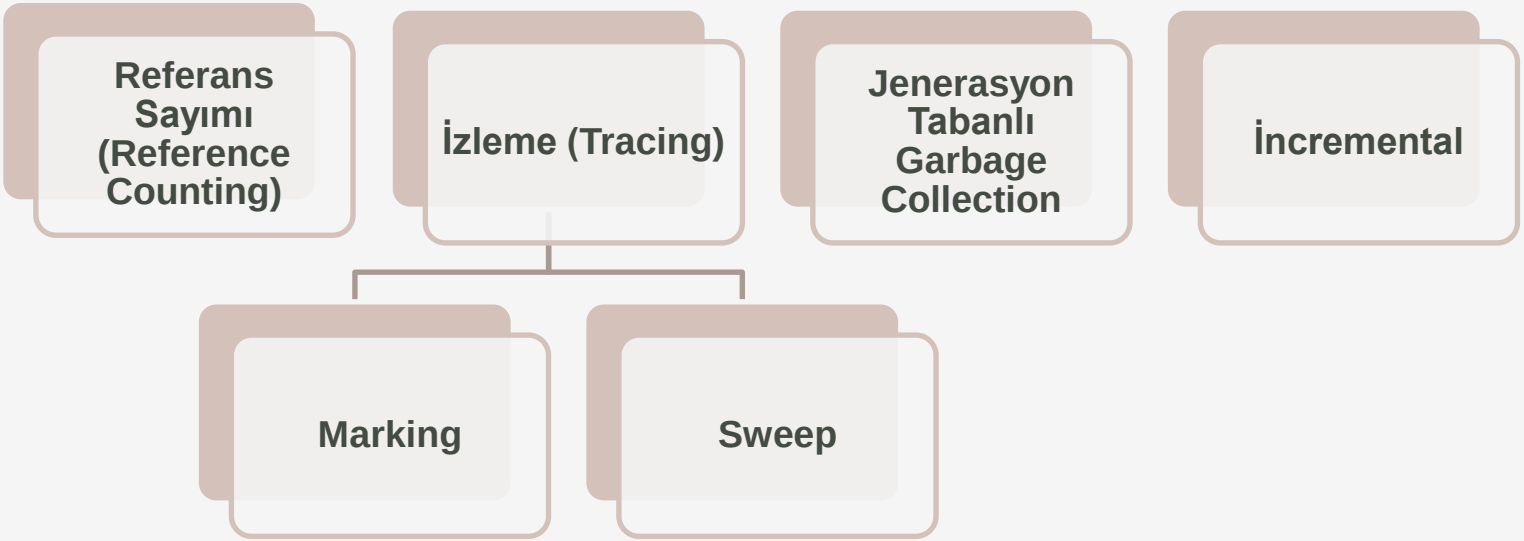
Garbage Collection 3 Ana Adımda Çalışır:

- **Bellek Tahsisi :** Program yeni bir nesne oluşturduğunda GC bu nesne için bellekten yer ayırır.

- **Referans Takibi :** Garbage collector, programın hangi nesneleri kullandığını takip eder. Bu takip, programın nesnelere olan referanslarını izleyerek yapılır.

- **Silme :** Kullanılmayan yani artık referansı bulunmayan nesnelerin bellekten silinmesi.

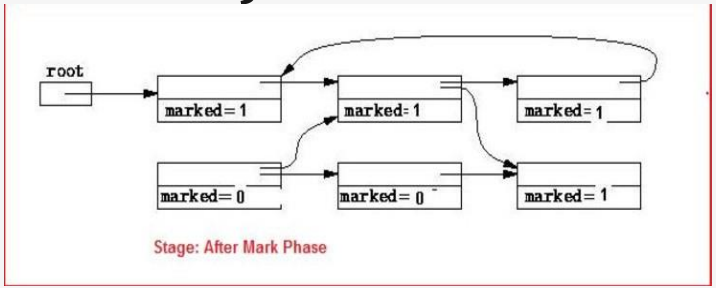
Kullanılan Algoritmalar Ve Yaklaşımlar



Referans Sayma(Reference Counting)

- Nesne üzerindeki referansları tutan bir sayaç değeri vardır.Nesne oluşturulduğunda referans sayısı 0 olarak ayarlanır.
- Bu değer nesneye yeni bir referans eklendiğinde veya nesneye işaret eden bir referans silindiğinde güncellenir.
- Eğer sayaç değeri 0 olursa, nesne artık kullanılmıyor anlamına gelir ve nesne bellekten silinir.
- Eğer birbirine referans veren 2 veya daha fazla nesne varsa sayaç asla sıfır olmaz. (Referans Döngüsü)

İzleme(Tracing) Yaklaşımı



2 Adımdan Oluşur: İşaretleme ve Silme

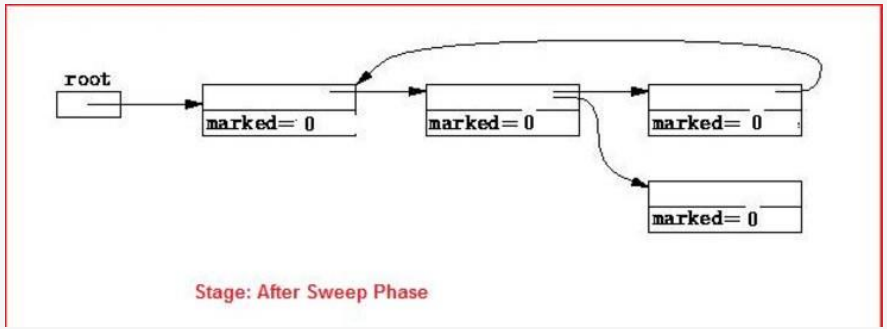
İşaretleme(Marking):

- Garbage collector , programın köklerinden başlayarak tüm erişilebilir nesneleri işaretler (mark).
- Kökler, genellikle global değişkenler, stack'teki değişkenler ve CPU kayıtları gibi yerlerdir.
- İşaretleme işlemi genellikle derinlik arama (depth-first search) algoritmasına benzer şekilde yapılır.
- İşlem, başlangıç noktalarından erişilebilen tüm nesneleri işaretlemeye kadar devam eder.

İzleme(Tracing) Yaklaşımı(Devam)

Silme (Sweep):

- Bu aşamada, tüm bellek alanı taranarak işaretlenmemiş nesneler tespit edilir ve bellekten kaldırılır.
- Bazı Garbage Collector çeşitlerinde, kullanılan bellek blokları arasındaki boşluklar da silinebilir. Bu belleği daha düzenli yapar.
- Bellek temizlendikten sonra, önceki yapılan işaretlemeler kaldırılır.
- Bu yöntem döngüler gibi karmaşık referans yapılarını da kontrol edebilir.
- Programın çalışması sırasında ek bir yük oluşturmaz.

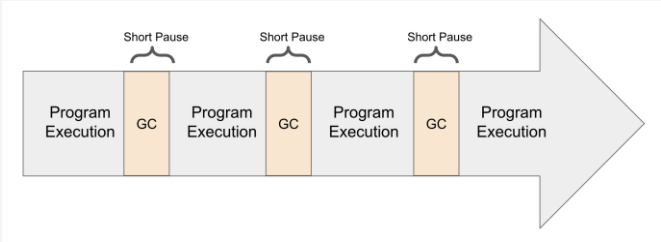


Jenerasyonel Yaklaşım

- Bu yöntemde nesneler, oluşturuldukları zamana göre farklı nesillere (generation) ayrılır.
- Yeni oluşturulan nesneler genellikle kısa ömürlüdür ve genç nesne grubuna atanır.
- Daha uzun süre kullanılan nesneler daha yaşlı gruplara alınır.
- Genç nesneleri toplamak daha sık ve hızlı bir şekilde gerçekleşir.
- Yaşlı nesneleri toplama işlemi daha nadir ve daha yavaş gerçekleşir.
- Çünkü yaşlı nesneler daha uzun süre kullanılır ve daha az değişir.

İncremental Garbage Collection

- Genellikle programın çalışmasını durdurmayacak küçük adımlarla çalışır.
- Bellek alanını belirli bölgelere ayırarak çöp toplama işlemini yapar. Böylece tüm bellek alanı tek seferde taranmamış olur.
- Threadler kullanılarak çöp toplama işlemi normal program çalışmasına entegre edilir.
- Programın çalışması sırasında gerçekleşen küçük iş parçacıkları sayesinde daha az takılma ve performans düşüşü yaşanır.
- Kullanım Alanları : Oyun Motorları, Gömülü Sistemler,Gerçek Zamanlı Sistemler



Java'da Bellek Yapısı

Heap Bellek

- Java programlarında oluşturulan diziler ve nesneler heap bellekte saklanır. Bu nesneler dinamik olarak oluşturulur ve heap belleğin içinde yönetilir.
- Javada nesneler arası ilişkiler referanslar ile sağlanır. Nesne referansları da heap bellekte saklanır.
- Heap bellek tüm threadler arasında ortak olarak paylaşılır.
- Garbage Collector artık kullanılmayan nesneleri heap bellekten siler.

Java'da Bellek Yapısı

Stack Bellek

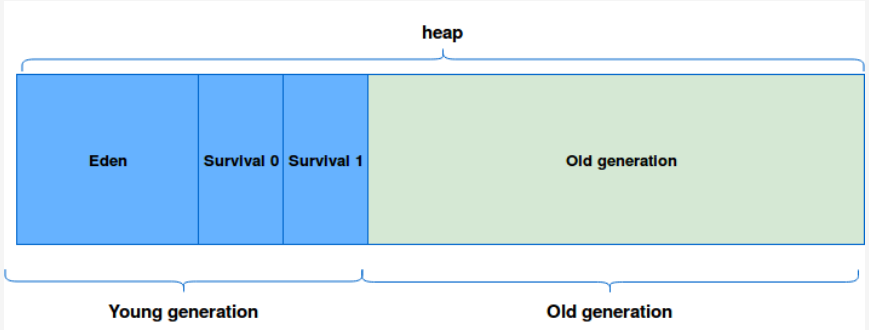
- Stack, yerel değişkenler, method çağrıları ve diğer program yürütme bilgilerinin saklandığı bellek alanıdır.
- Statik bir bellek alanıdır. Program başında oluşturulur ve program sonuna kadar aynı kalır.
- Metod çağrıları için Stack Frameler oluşturulur. Stack Frame ilgili metodun çalışması için gerekli olan bilgileri tutar.
- Her thread için ayrı bir stack bellek alanı oluşturulur.Bu sayede her thread kendi metod çağrılarını ve yerel değişkenlerini saklayabilir.

Java'da Garbage Collection

- Yeni oluşturulan nesneler genellikle kısa zamanda kullanılır ve çöp olur.Bu yüzden heap bellek farklı yaş kategorilerine ayrılmıştır.

- Yeni oluşturulan tüm nesneler ilk olarak Eden alanına taşınır.Bu alan nispeten küçüktür ve nesnelerin çoğu burda kısa bir süre toplanır.

- Eden bellek alanı dolduktan sonra Minor Garbage Collection işlemi tetiklenir.



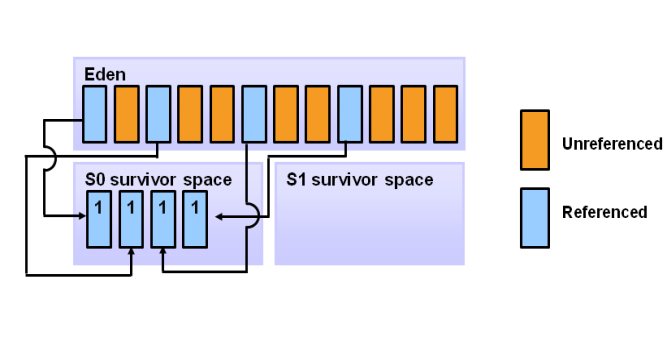
Java'da Garbage Collection

Minor Garbage Collection:

- Eden alanı dolduktan sonra gerçekleşir.Hayatta kalan nesneler işaretlenerek S0 alanına taşınır.İşaretsiz olanlar silinir.

- S0 alanı da dolduktan sonra hayatta kalan nesneler S1 alanına taşınır. Yani bir nesne 2 garbage collection işleminden sağ çıkarsa S1 alanına taşınır.

- S1 alanı çöp olma ihtimali daha az olan nesnelerin tutulduğu bir alan oluğu için daha az kontrol edilir.



Java'da Garbage Collection

Old Generation Bellek Alanına Taşıma 2 Şekilde Gerçekleşir :

1. **Yaş Temelli Taşıma:**

- S1 alanındaki nesneler belirli bir süre(varsayılan 15 dk.) tutulduktan sonra Old Generation alanına taşınır.

- Bu süre JVM komutlarıyla değiştirilebilir.

2. **S1 Alanının Dolması Durumu:**

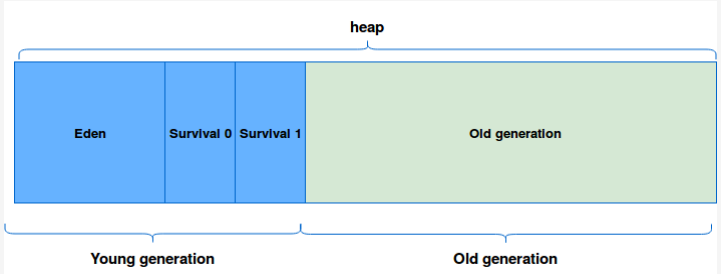
- S1 alanı dolduğunda Full-Garbage Collection işlemi tetiklenir.

- Hayatta kalan nesneler Old Generation alanına taşınır.

Java'da Garbage Collection

Major Garbage Collection:

- Old Generation uzun süre yaşayan ve nadiren GC tarafından toplanan nesnelerin toplandığı alandır.
- Old Generation alanı dolduğunda veya JVM için bellek yetersizliği durumu oluştuğunda gerçekleşir.
- Bütün bellek alanlarını kapsayan bir temizleme işlemidir.
- Minor Garbage Collection işlemine göre daha az sıklıkla ve daha yavaş gerçekleşir.
- Büyük miktarda bellek alanını temizlemeye çalıştığı için program performansını etkileyebilir.



Farklı Garbage Collection Türleri

Java da farklı garbage collection türleri, bellek yönetimini optimize etmek ve uygulama performansını arttırmak için JVM tarafından sunulur.

Serial Garbage Collector:

- Genellikle single thread ortamlar için uygundur.
- Basit ve düşük maliyetlidir.
- Çok işlemcili ortamlarda performans sorunlarına neden olabilir.
- Büyük heap boyutları için uygun değildir.

Java'da Garbage Collection

Parallel Garbage Collector:

- Genellikle çok işlemcili ortamlar için uygundur.
- Çoklu işlemci kullanarak toplama işlemini hızlandırır.
- Verimli ve iyi performans sunar.

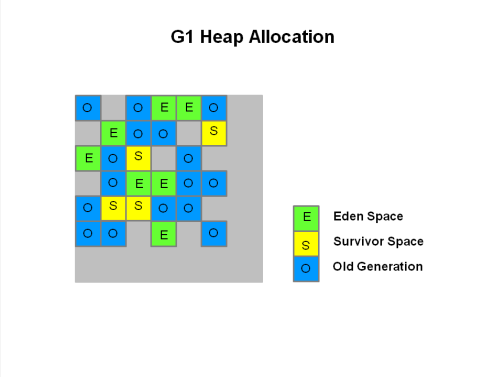
Concurrent Garbage Collector:

- Kısa duraklamalar ve yüksek tepki süreleri gereken uygulamalar için kullanılır.
- CPU kullanımı genellikle daha düşüktür.
- Fragmentasyon problemleri yaşanabilir.

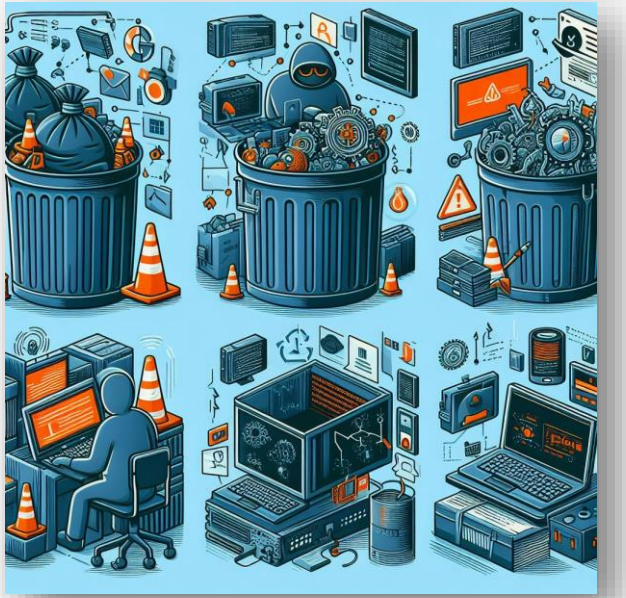
Java'da Garbage Collection

G1 Garbage Collector:

- Büyük heap boyutları ve düşük gecikme süreleri için tasarlanmıştır.
- Heap belleği farklı bölgelere ayırır ve ayrı ayrı toplar.
- Küçük ölçekli uygulamalarda diğer GC tiplerine göre daha fazla kaynak tüketebilir.
- Uygulama thread'leri çalışırken çöp toplama işleminin bir kısmını arka planda gerçekleştirebilir. Bu sayede programın akıcılığını korumaya çalışır.



Yazılım Güvenliğine Olan Etkileri



- **Olumlu Etkileri:**
- Bellek sızıntılarını önler.
- Bellek yönetimini kolaylaştırır.
- Program ömrünü uzatır.
- **Olumsuz Etkileri:**
- Performans sorunlarına yol açabilir.
- Gargabe Collection karmaşık algoritmalar kullanır. Bu programcıların kodların nasıl davranacağını anlamasını, dolayısıyla hata ayıklamayı zorlaştırabilir.

KAYNAKLAR:

https://tugrulbayrak.medium.com/jvm-garbage-collector-nedir-96e76b6f6239

https://www.bing.com/chat?q=Microsoft%20Copilot&qs=ds&form=HPCODX

https://gemini.google.com/app/b0a6ef03a5906d7a

https://www.baeldung.com/java-stack-heap

https://medium.com/bili%C5%9Fim-hareketi/javada-bellek-y%C3%B6netimi-c99d73657577

https://www.youtube.com/watch?v=Mlbyft_MFYM&pp=ygUXamF2YSBnYXJiYWdlIGNvbGxlY3Rpb24%3D

Sorularınız ?

Beni Dinlediğiniz İçin Teşekkürler