Java Garbage Collection

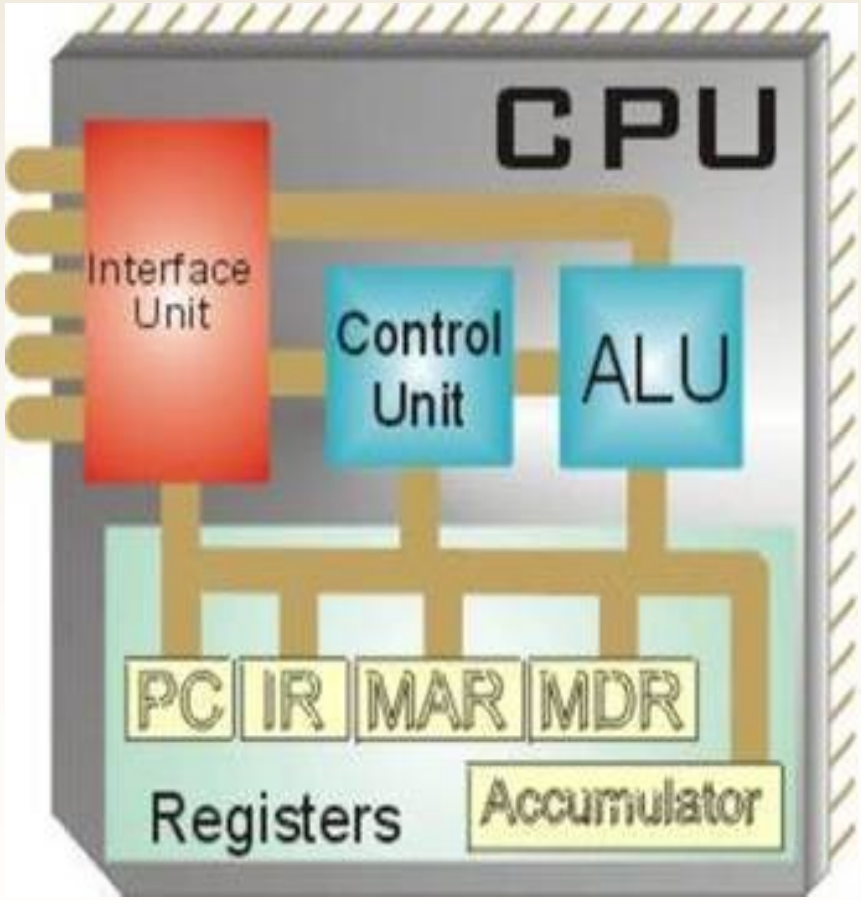
harezmi bilişim çözümleri

Veri Depolama Alanları

- Sistemdeki verinin **CPU ve RAM içinde değişik yerler**de tutulması söz konusu olabilir
- CPU'da
  - register
- RAM'de
  - stack
  - heap

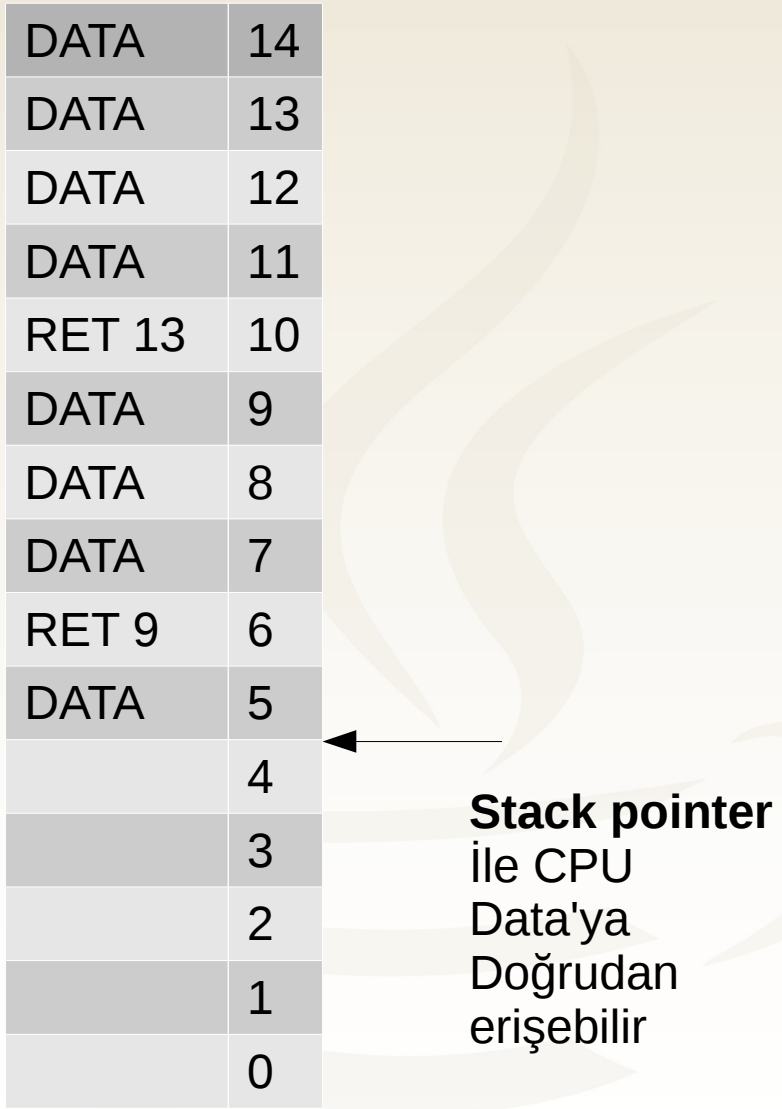
Register

- En hızlı veri depolama alanıdır
- Programcılarının register'lara erişimi **Java'da mümkün değil**dir



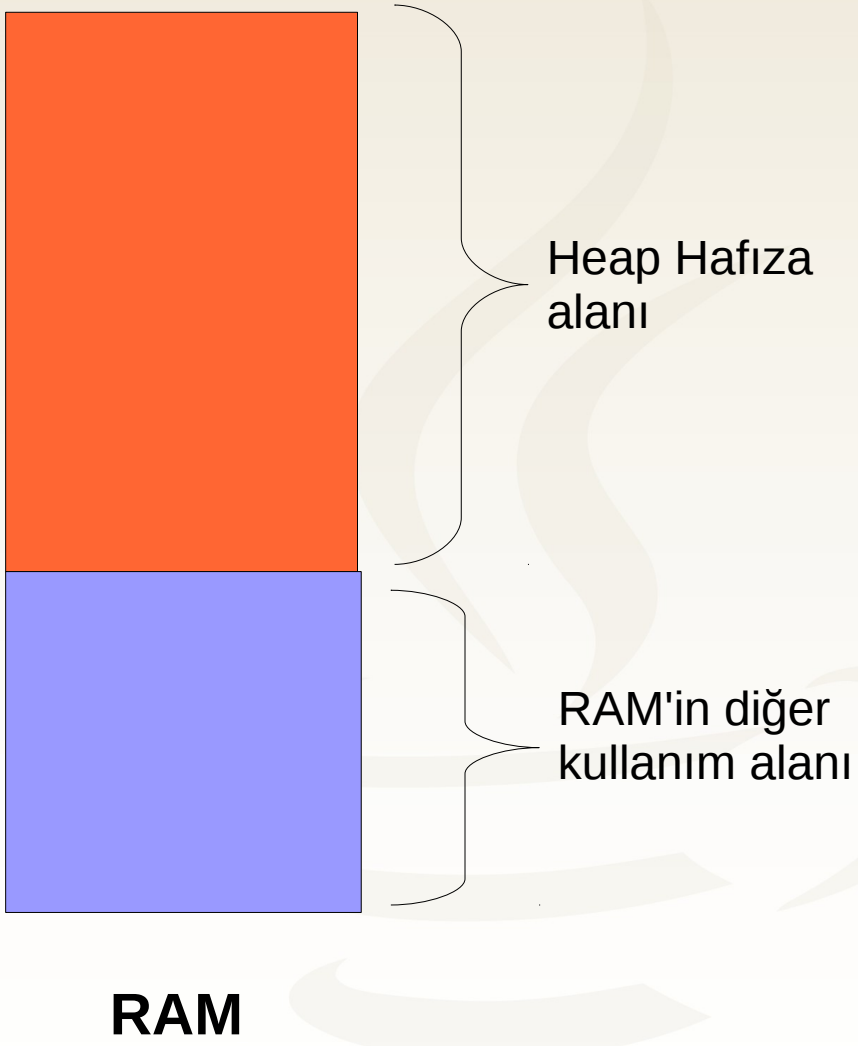
Stack

- RAM içerisindedir
- Stack pointer hafızada aşağıya ilerletilerek yeni bir değer saklanır, veya yukarı taşınarak mevcut hafızanın boşaltılabilir
- Register'lardan sonra en hızlı veri saklama alanıdır
- **Java nesneleri stack üzerinde tutulmaz**
- Stack üzerinde **sadece primitif lokal değişkenler** tutulur

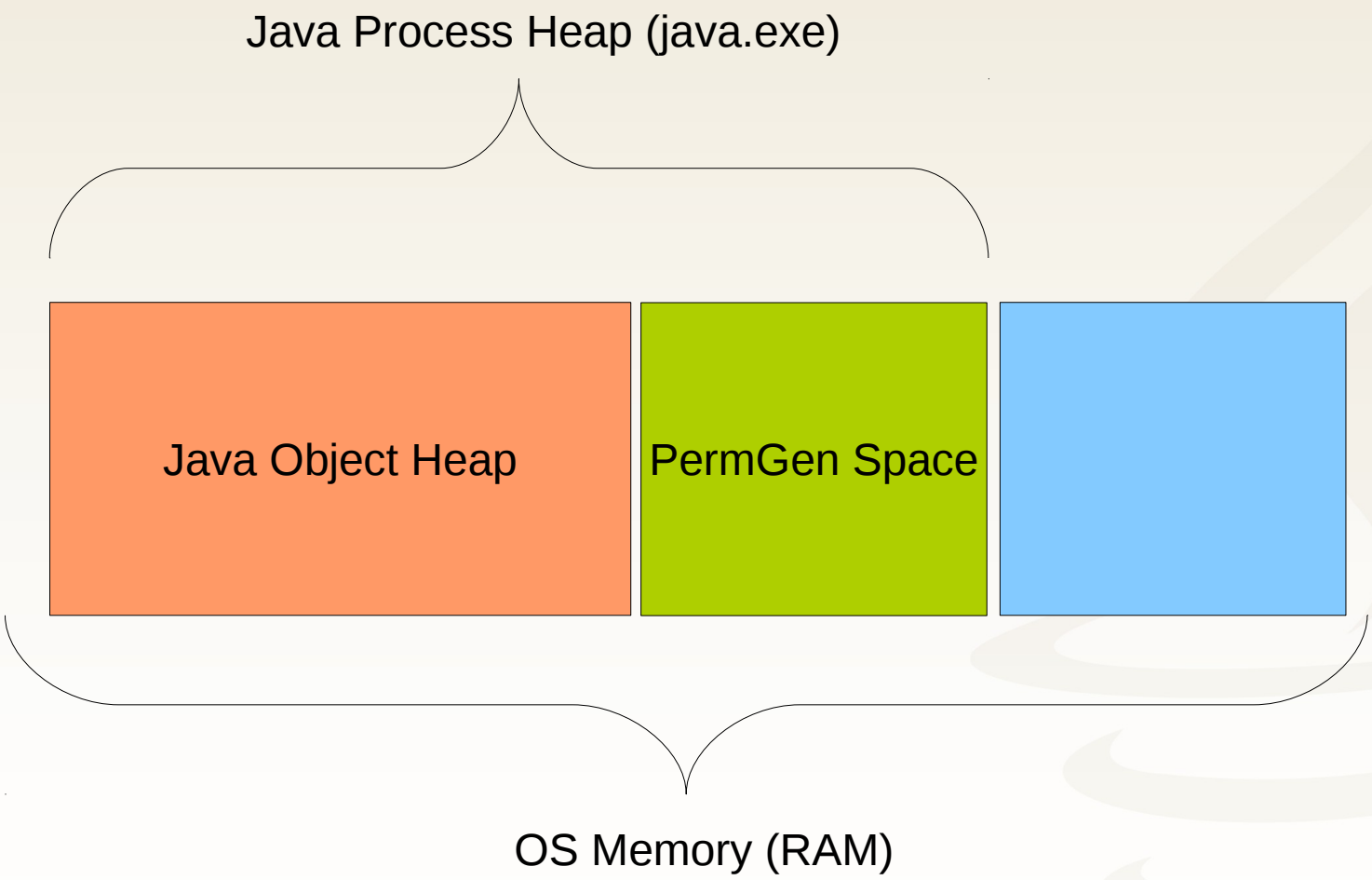


Heap

- RAM'de yer alır
- Genel amaçlı hafıza havuzudur
- Heap, stack ile karşılaştırıldığında daha yavaştır
- Bütün **Java nesneleri bu alanda tutulur**lar



İşletim Sistemide RAM Layout



PermGen Space Nedir?

- JVM ve uygulamayla ilgili diğer verinin tutulduğu alandır
  - Class, method tanımları
  - String literal değerler
  - JNI kodları
  - Thread stack'leri
  - GC ile ilgili bilgiler
  - Socket bağlantıları ile ilgili bilgiler

PermGen Space ve Java 8

- Java 8'de **perm gen space kaldırılmış** ve yerine **metaspace gelmiştir**
- **Temel fark** perm gen space'in size'ı JVM başlatılırken fix'lenir
- Metaspace size'ı ise işletim sistemindeki **mevcut hafıza alanı kadar** büyüyebilir
- -XX:MaxMetaspaceSize ile bu özellik sınırlandırılabilir

Java'da Garbage Collection İşlemi

JAVA Eğitimleri

- Java Virtual Machine, **otomatik hafıza yönetimi** yapmaktadır
- **Nesne yaratılması** programcının istediği anda gerçeklemektedir
- Ancak scope'dan çıkmış **nesnelere ait hafıza alanının serbest bırakılması** tamamen JVM'in kontrolüne bırakılmıştır
- Uygulama içerisinden **doğrudan veya dolaylı olarak erişimi olmayan nesnelere** ait hafıza alanı "**garbage collector**" tarafından toplanır

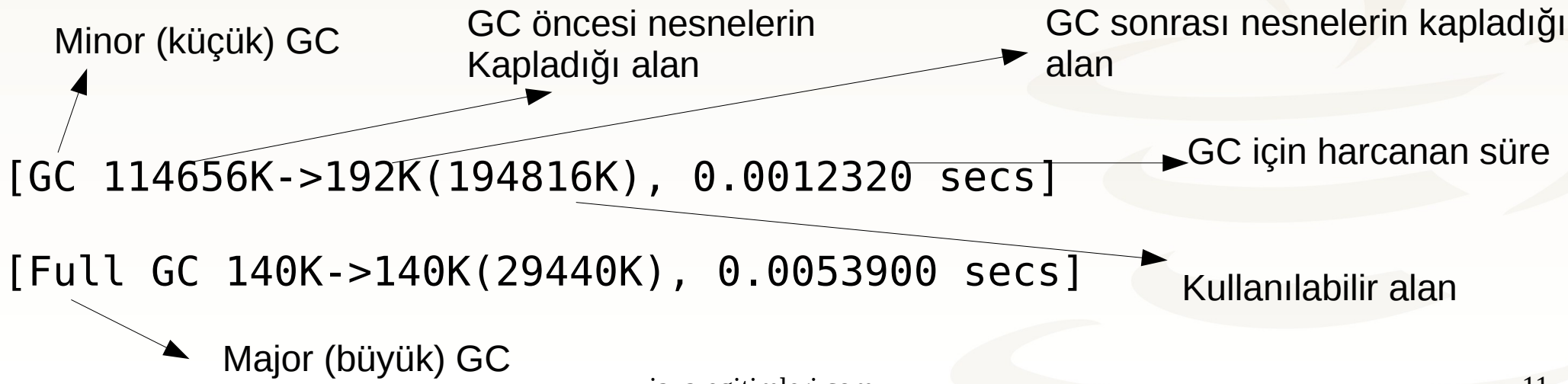
Java'da Garbage Collection İşlemi

- GC'nin uygulama çalışırken **ne zaman devreye gireceği** net değildir
- Erişilemez hale gelmiş nesnelerin **"garbage collect" edilip edilmeyecekleri** de kesin değildir
- JVM nesneleri **heap hafıza alanı**nda yaratmaktadır
- Heap alanının büyüklüğü **-Xms -Xmx** directive'leri ile belirlenebilir
- java -Xms512m -Xmx1g MyApp

Java'da Garbage Collection İşlemi

```
public class GCTest {
   public static void main(String[] args) {
      int i=0;
      while(true) {
         new String("hello"+i++);
      }
   }
}
```

java -verbose:gc ile uygulamayı çalıştırdığımız vakit aşağıdakine benzer bir log ortaya çıkar

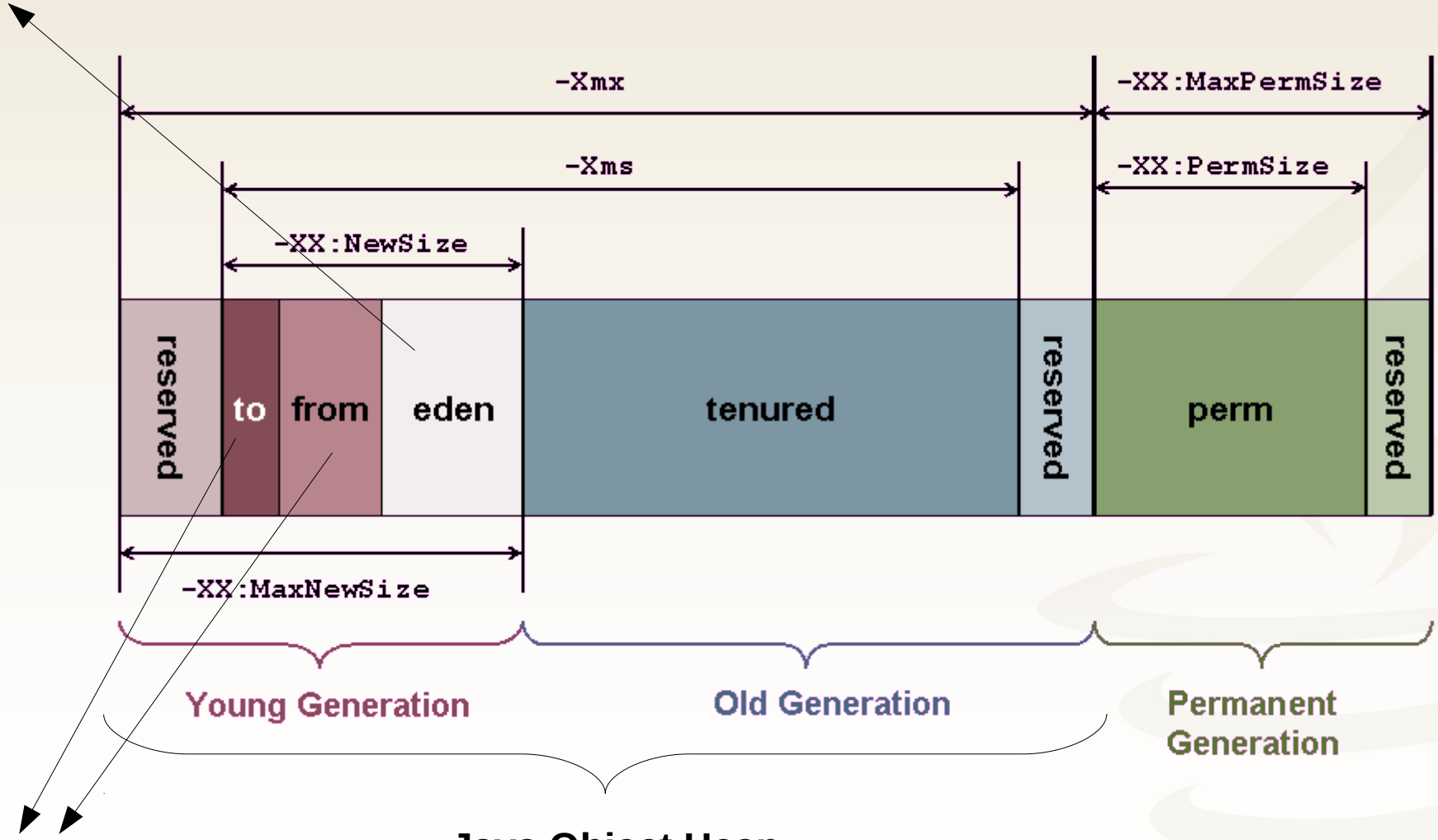


Java'da Garbage Collection İşlemi

- Minor GC sıklıkla yapılır
- Major GC old generation'da yer azalmayana değin yapılmaz
- JVM'in idle olduğu durumlarda **System.gc()** metodu ile Full (major) GC tetiklenebilir

JVM Process RAM Layout

Yeni nesneler ilk olarak **eden space**'de yaratılır



Survivor space adı verilir
Minor GC sırasında kullanılır

JVM Process RAM Büyüklüğü

- **32 bit** işletim sisteminde **max 2GB RAM** kullanılabilir
- **64 bit** işletim sisteminde böyle bir **sorun yoktur**
- Eğer 2GB RAM olan bir sistemde JVM'in -Xms1800m -Xmx1800m ile başlatılması doğru değildir
- Çünkü diğer process'ler için neredeyse hiç yer kalmayacaktır

Java Object Heap Kullanımı

- Yaratılan **nesnelerin %80-%90'ı çok kısa ömürlü**dür
- "**Young Generation**" için uygun büyüklük **heap alanının %33'ü**dür
- Bütün **yeni nesneler** young generation'ın **eden space'inde yaratılır**lar
- Eğer nesneler **1-2 minor GC'de aktif kalırlarsa old generation'a taşınır**lar

Heap Dump

- JVM heap'in belirli bir anda elde edilen **snapshot görüntü**südür, binary hprof formatındadır
- İçerisinde o an heap'de yer alan **bütün nesnelerin state'leri** yer alır
- Farkı yöntemlerle elde edilebilir
  - JDK_HOME/bin/jcmd ile java processlerinin ID'leri tespit edilebilir
  - JDK_HOME/bin/jmap -dump:live,file=out.hprof <PID>
  - JDK_HOME/bin/jcmd <PID> GC.heap_dump out.hprof

OutOfMemoryError Hatası

- JVM **nesneler için yer temin edemediği vakit** ortaya çıkar
- **-XX:+HeapDumpOnOutOfMemoryError** opsiyonu ile OOME olduğu vakit **heap dump** alınabilir
- Örneğin -Xmx256m bir heap **256m .hprof dosyası** oluşturacaktır
- Dump sırasında JVM yeniden başlatılmamalıdır

OutOfMemoryError Hatası

- OOME kurtulmak için -Xmx ile max heap size'ın artırılması **genellikle çözüm olur**
- Diğer durumlarda daha **ayrıntılı monitoring** yapılması gerekebilir

PermGen Space ve OutOfMemoryError Hatası

- Perm gen space'de tutulan Class vb yapılar **garbage collect edilmez**ler
- Web uygulamalarında reload işlemlerinden dolayı bu **permgen space'in tükenmesi** ve OutOfMemoryError hatası sıkça rastlanır
- Çoğu zaman **max perm size**'ın artırılması sorunu çözer
- Java 8 ile birlikte bu durum ortadan kalkmıştır

Thread Stack Size ve OutOfMemoryError Hatası

- Bazen OOME mesajı “unable to create new native thread” şeklinde olabilir
- JVM'de her thread için ayrı bir **“thread stack” memory alanı** ayrılır
- -Xss ile kontrol edilir, default değeri OS/JVM'e göre değişir
- Bu tür bir OOME hatasının çözümü için -Xmx veya -Xss değerleri azaltılabilir
- Diğer yandan çok düşük -Xss değeri ise **StackOverflowError** hatasına yol açabilir

Finalize Metodu

- Nesnenin **hafıza alanı garbage collector tarafından geri alınırken çalıştırılan bir metot**tur
- Nesnelerin hepsinin GC yapılacağı garanti olmadığı için **finalize metodunun da hiç çağrılmama ihtimali var**dır
- Bu metot içerisinde **memory release etme işlemlerinin dışında bir cleanup yapılmamalı**dır!
- Örneğin açılan **dosyaların finalize metodunda kapatılması doğru değil**dir!

Finalize Metodu

```
protected void finalize() throws Throwable {
    try {
        //cleanup islemleri yapilir
    } finally {
        super.finalize();
    }
}
```

Üst sınıfa ait nesnenin
Cleanup işlemlerinin de
Gerçekleşmesi sağlanmalıdır

Native metotlarla ilgili memory
Clean up işlemleri finalize için
En uygun işlemlerdir

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri