

# 10 Points about Java heap memory

When I started java programming I didn't know what is java heap or heap space in Java, I was even not aware of where does object in Java gets created, it's when I started doing professional programming I came across error [java.lang.OutOfMemoryError in Tomcat](#) then I realized What is Heap in Java or Java Heap Space. Its happens with most of programmer because learning language is easy but learning basics is difficult since there is no formal process which can teach you every basics of programming its experience and work which reveals the secret of programming. For Java developer knowledge of Heap in Java, setting size of java heap space, dealing with Java heap space `OutOfMemoryError`, analyzing heap dumps is very important. This Java Heap tutorial is for my beginner brothers who are new in programming and learning it. It makes too much difference if you know the basics and underlying, until you know that object is created in heap, you won't be able to think why `OutOfMemoryError` occurs in Heap. I am trying to provide as much information about Heap in Java as I know but would like you guys to contribute and share your knowledge about Heap in Java to benefit all. By the way if you are confused between heap and stack, which is where your local variables get created, then, you can also check [difference between heap and stack memory in Java](#).

## What is Heap space in Java?

When a Java program started Java Virtual Machine gets some memory from Operating System. Java Virtual Machine or JVM uses this memory for all its need and part of this memory is call java heap memory. Heap in Java generally located at bottom of address space and move upwards. whenever we create object using new operator or by any another means object is allocated memory from Heap and When object dies or garbage collected ,memory goes back to Heap space in Java, to learn more about garbage collection see [how garbage collection works in Java](#).

## How to increase heap size in Java

Default size of Heap space in Java is 128MB on most of 32 bit Sun's [JVM](#) but its highly varies from JVM to JVM e.g. default maximum and start heap size for the 32-bit Solaris Operating System (SPARC Platform Edition) is `-Xms=3670K` and `-Xmx=64M` and Default values of heap size parameters on 64-bit systems have been increased up by approximately 30%. Also if you are using throughput garbage collector in Java 1.5 default maximum heap size of JVM would be Physical Memory/4 and default initial heap size would be Physical Memory/16. Another way to find default heap size of JVM is to start an application with default heap parameters and monitor in using JConsole which is available on JDK 1.5 onwards, on VMSummary tab you will be able to see maximum heap size.

By the way you can increase size of java heap space based on your application need and I always recommend this to avoid using default JVM heap values. if your application is large and lots of object created you can change size of heap space by using JVM options `-Xms` and `-Xmx`. `Xms` denotes starting size of Heap while `-Xmx` denotes maximum size of Heap in Java. There is another parameter called `-Xmn` which denotes Size of new generation of Java Heap Space. Only thing is you can not change the size of Heap in Java dynamically, you can only provide Java Heap Size parameter while starting JVM. I have shared some more useful JVM options related to Java Heap space and Garbage collection on my post [10 JVM options Java programmer must know](#), you may

find useful.

### **Update:**

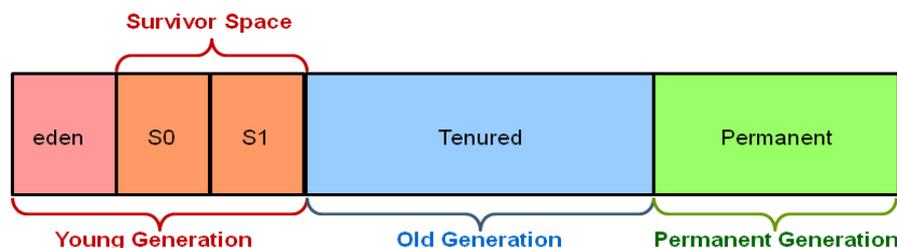
Regarding default heap size in Java, from Java 6 update 18 there are significant changes in how JVM calculates default heap size in 32 and 64 bit machine and on client and server JVM mode:

- 1) Initial heap space and maximum heap space is larger for improved performance.
- 2) Default maximum heap space is 1/2 of physical memory of size upto 192 bytes and 1/4th of physical memory for size upto 1Gig. So for 1Gig machine maximum heap size is 256MB  
2.maximum heap size will not be used until program creates enough object to fill initial heap space which will be much lesser but at-least 8 MB or 1/64th part of Physical memory upto 1GB.
- 3) for Server Java virtual machine default maximum heap space is 1G for 4GB of physical memory on a 32 bit JVM. for 64 bit JVM its 32G for a physical memory of 128GB. To learn more about [how much memory you can set in 32-bit and 64-bit JVM](#) in various operating system e.g. Windows 8, Linux, or Solaris, see here.

## **Java Heap and Garbage Collection**

As we know objects are created inside heap memory and Garbage collection is a process which removes dead objects from Java Heap space and returns memory back to Heap in Java. For the sake of Garbage collection Heap is divided into three main regions named as New Generation, Old or Tenured Generation and Perm space. New Generation of Java Heap is part of Java Heap memory where newly created object are stored, During the course of application many objects created and died but those remain live they got moved to Old or Tenured Generation by Java Garbage collector thread on [Major or full garbage collection](#). Perm space of Java Heap is where JVM stores Meta data about classes and methods, String pool and Class level details. You can see How Garbage collection works in Java for more information on Heap in Java and Garbage collection.

### **Hotspot Heap Structure**



## **OutOfMemoryError in Java Heap**

When JVM starts JVM heap space is equal to the initial size of Heap specified by -Xms parameter, as application progress more objects get created and heap space is expanded to accommodate new objects. JVM also run garbage collector periodically to reclaim memory back from dead objects. JVM expands Heap in Java some where near to Maximum Heap Size specified by -Xmx and if there is no more memory left for creating new object in java heap , JVM throws `java.lang.OutOfMemoryError` and your application dies. Before throwing [OutOfMemoryError No Space in Java Heap](#), JVM tries to run garbage collector to free any available space but even after that not much space available on Heap in Java it results into `OutOfMemoryError`. To resolve this error you need to understand your application object profile i.e. what kind of object you are creating, which objects are taking how much memory etc. you can use profiler or heap analyzer to troubleshoot `OutOfMemoryError` in Java. "`java.lang.OutOfMemoryError: Java heap space`" error messages denotes that Java heap does not have sufficient space and cannot be expanded further while "`java.lang.OutOfMemoryError: PermGen space`" error message comes when the permanent generation of Java Heap is full, the application will [fail to load a class](#) or to allocate an interned string.

## **Java Heap dump**

Java Heap dump is a snapshot of Java Heap Memory at a particular time. This is very useful to analyze or troubleshoot any memory leak in Java or any `java.lang.OutOfMemoryError`. There are tools available inside JDK which helps you to take heap dump and there are heap analyzer available tool which helps you to analyze java heap dump. You can use "`jmap`" command to get java heap dump, this will create heap dump file and then you can use "`jhat - Java Heap Analysis Tool`" to analyze those heap dumps.

## **How to increase Java heap space on Maven and ANT**

Many times we need to increase heap size of Maven or ANT because once number of classes increases build tool requires more memory to process and build and often throw `OutOfMemoryError` which we can avoid by changing or increase heap memory of JVM. For details see my post [How to increase java heap memory for Ant or Maven](#)

## **10 Points about Java Heap Space**

1. Java Heap Memory is part of memory allocated to JVM by Operating System.
2. Whenever we create objects they are created inside Heap in Java.
3. Java Heap space is divided into three regions or generation for sake of garbage collection called New Generation, Old or tenured Generation or Perm Space. Permanent generation is garbage collected during full gc in hotspot JVM.
4. You can increase or change size of Java Heap space by using JVM command line option -Xms,

-Xmx and -Xmn. don't forget to add word "M" or "G" after specifying size to indicate Mega or Gig. for example you can set java heap size to 258MB by executing following command `java -Xmx256m HelloWorld`.

5. You can use either `JConsole` or `Runtime.maxMemory()`, `Runtime.totalMemory()`, `Runtime.freeMemory()` to query about Heap size programmatic in Java. See my post [How to find memory usage in Java program](#) for more details.

6. You can use command "jmap" to take Heap dump in Java and "jhat" to analyze that heap dump.

7. Java Heap space is different than Stack which is used to store call hierarchy and local variables.

8. Java Garbage collector is responsible for reclaiming memory from dead object and returning to Java Heap space.

9. Don't panic when you get `java.lang.OutOfMemoryError`, sometimes its just matter of increasing heap size but if it's recurrent then look for [memory leak in Java](#).

10. Use Profiler and Heap dump Analyzer tool to understand Java Heap space and how much memory is allocated to each object.

Read more: <http://javarevisited.blogspot.com/2011/05/java-heap-space-memory-size-jvm.html#ixzz3UjdP2Oh4>