**Memory Management in Java**

Memory management is very important in understanding the working of Garbage Collection (G.C). Java has an automatic system of clearing unused objects, but learning about garbage collection helps to find the memory leaks easily. Understanding memory management concepts help to write applications effectively in an optimized way which never crash with out of memory.

**JVM Memory Model :**

JVM memory model is divided into two categories at a broader level. They are:

i) Young Generation

ii) Old Generation

**Memory Management -Young Generation:**

All the newly created objects will settle in the young generation sector of the heap. Once the younger generation is filled garbage collection is performed. This is called minor G.C

Young Generation sector is divided into three parts :

i) Eden

ii) Survivor Memory 0

iii) Survivor Memory 1

Eden mostly comprises the newly created objects, Once Eden gets filled Minor garbage collection takes place. All the survived objects will be moved to one of the survived slots, the objects present in one survival slot will moved into another slot, so that one survival slot would be empty all the time. Even after many cycles if there are any objects persisted in young generation, they are moved into the old generation of heap. This is done by setting up a threshold value for the age of objects present in the heap.

**Memory Management - Old Generation:**

The objects which are survived after many rounds of minor G.C are present in the old generation. If the old generation of the heap gets filled, Garbage collection takes place and this process is called Major G.C. The process of Major G.C usually takes longer time.

**Stop the World event :**

Usually garbage collections are stop the world events. Application becomes unresponsive as all the application threads are stopped during garbage collection. Minor G.C's do not take much time, hence application is not affected by this. But when it comes to Major G.C, it takes longer time as it checks for live objects everywhere. Major G.C should be minimized as it makes application unresponsive for longer time. This results in Timeout errors. However duration of garbage collection depends upon the strategy it makes during the process.

**Java Permanent Generation:**

Also called as "perm gen", This is not a part of java heap memory. It consists of meta data of classes which are required by JVM to describe methods and classes in application. It is populated by JVM at run time based on the classes used by JVM.

**Java Memory Model - Method area:**

This method space is the part of Perm gen which is used to store class structure and code for methods and constructors.

**Java Memory Model - Memory Pool:**

This area is created to store immutable objects by JVM. Memory pool can be a part of Heap or Perm Gen depending upon JVM memory manager.

**Java Memory Model - Java Stack Memory:**

Java Stack Memory is used for execution of a thread. Stack refers to the value types and short lived objects in the heap.

**Java Garbage Collection:**

This is a process to clear unused objects from the memory and free space is allocated to objects which are created in further processing. Garbage collector is the program which runs in the background all the time. The program looks for unused objects and allocates space for other objects.

**Steps involving in garbage collection:**

i) **Marking** : This step involves identifying the unused objects and process them.

ii) **Normal Deletion :** Garbage collector removes the unused objects and reclaim for free space.

iii) **Deletion with Compacting** : For better performance all the survived objects will be moved together.

**Garbage Collection Types :**

There are five types of garbage collections that we use in our applications, They are:

i) Serial GC

ii) Parallel GC

iii) Parallel Old GC

iv) Concurrent Mark Sweep Collector

v)G1 GC

**Serial GC :**

This is useful in client machines such as our simple stand alone applications and machines with smaller CPU.

**Parallel GC :**

Parallel GC is also called as throughput collector because it uses multiple CPU's to increase GC performance.

**Parallel Old GC :**

This is same as Parallel GC except that it uses multiple threads for both Young Generation and Old Generation garbage collection.

**Concurrent Mark Sweep Collector :**

CMS Collector is also referred as concurrent low pause collector. It does the garbage collection for Old generation. CMS collector tries to minimize the pauses due to garbage collection by doing most of the garbage collection work concurrently with the application threads.

**G1 Garbage Collector:**

This is available from Java7, Its goal is to replace the CMS collector. G1 collector is a parallel, concurrent and incrementally compacting low pause garbage collector. This removes the concept of Young and Old generations. The heap is divided into equal sized regions. When garbage collector is invoked it first collects the region with lesser live data.