Garbage collection

Garbage Collection is a process to identify and delete the objects from Heap memory which are not in use. GC frees the space after removing unreferenced objects.

Garbage= un -referencable object.

Automatic Gargabe Collection --- to avoid memory, leaks/holes

JVM creates 2 system thrds --- main thrd(to exec main() sequentially) -- foreground thrd G.C --- daemon thrd ---background thrd --- JVM activates it periodically(only if required) --- GC releases the memory occupied by un-referenced objects allocated on the heap(the objects whose no. of ref=0)

How to request for GC ? API of System class

public static void gc() eg : System.gc();//it's simply a REQUEST to JVM , for running GC htread.

Object class API protected void finalize() throws Throwable Automatically called by the garbage collector on an object before garbage collection of the object takes place.

Whenever Heap is under pressure (i.e there is no space in heap to create more objects) , JVM activates the **GC thread**. BUT it refers to Heap only. The classes which are loaded in method area are unloaded : when JVM terminates.

----------1st half over----------

Releasing of non- Java resources.(eg - closing of DB connection, closing file handles,closing socket connections) is NOT done automatically by GC

Triggers for marking the object for GC(candidate for GC)

1. Nullifying all valid refs. eg : Box b1=new Box(1,2,3); Box b2=b1; b1=b2=null;//Box obj is marked for GC
2. re-assigning the reference to another object eg : Box b1=new Box(10,20,30); b1=new Box(2,3,4);
3. Object created within a method & its ref NOT returned to the caller.
4. Island of isolation

More Details

Garbage Collection is a process to identify and delete the objects from Heap memory which are not in use. GC frees the space after removing unreferenced objects.

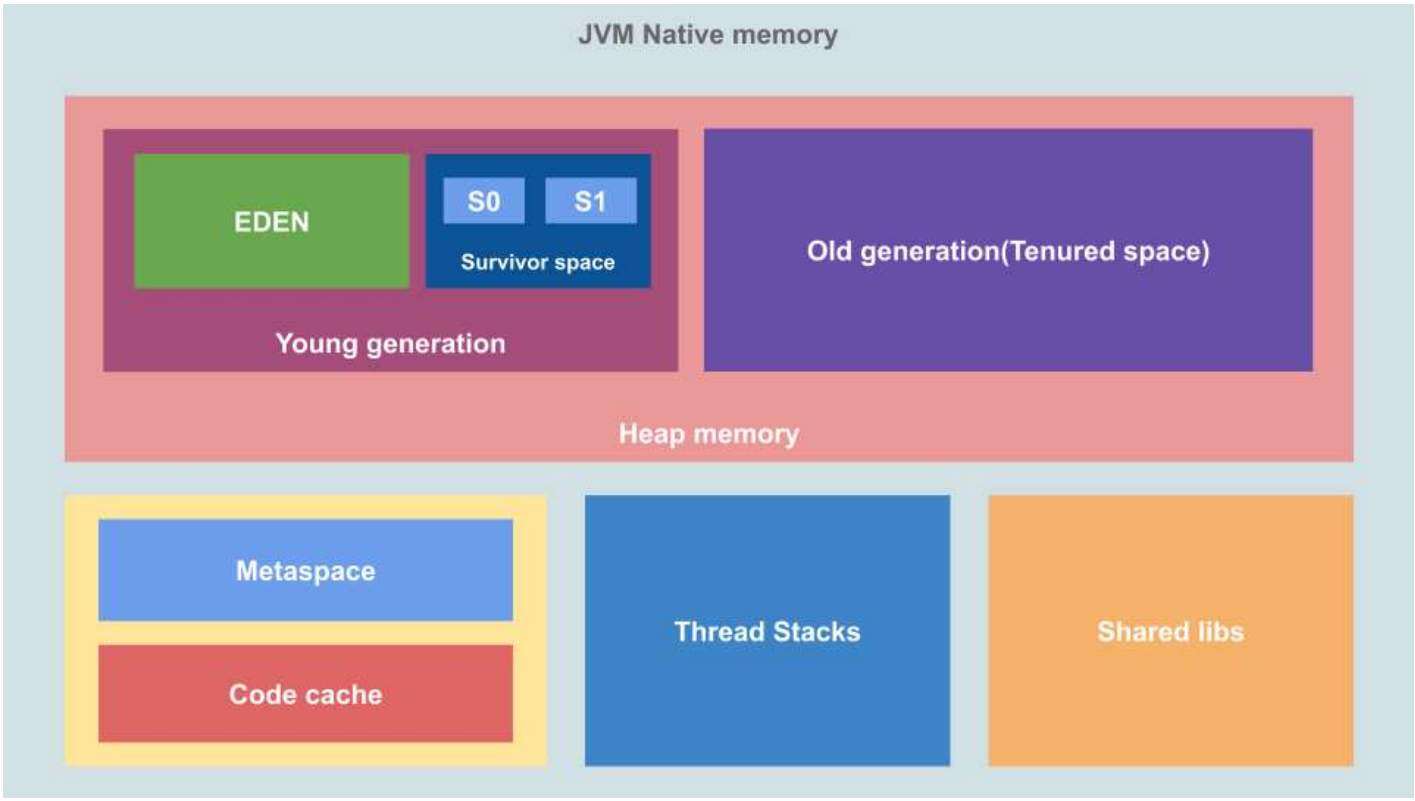
The event in which Garbage Collectors are doing their job is called **"Stop the world"** event which means all of your application threads are put on hold until the garbage is collected.

How Garbage Collector works

The basic process of Hotspot JVM Garbage collector completes in two phases:

1. **Marking** This phase is called marking phase in which GC identifies which objects are in use or which are not. All objects are scanned in the marking phase to make this determination.
2. **Deletion** In Deletion phase, the marked object is deleted and the memory is released. Deletion of the unreferenced objects can be done in two ways:

2.1 **Normal Deletion: I**n this phase, all unused objects will be removed and memory allocator has pointers to free space where a new object can be allocated. OR 2.2 **Deletion and Compaction:** As you see in normal deletion there are free blocks between referenced objects. To further improve performance, in addition to deleting unreferenced objects, remaining referenced object will be compacted(re aligned).



Why Heap divided into Generations ? It is a time consuming process to scan all of the objects from a whole heap and further mark and compact them. The list of the object grows gradually which leads to longer garbage collection time as more and more objects are allocated with time.

In General Applications most of the objects are short-lived. Fewer and fewer objects remain allocated over time.

That's why to enhance the performance of the JVM, Heap is broken up into smaller parts called generations and JVM performs GC in these generations when the memory is about to fill up.

Generational Process of Garbage Collection

1. New objects are allocated in Eden Space of Young Generation. Both Survivor Spaces are empty in starting.
2. A minor garbage collection will trigger once the Eden space fills up. Referenced objects are moved to the S0 survivor space and Eden Space will be cleared and all unreferenced objects will be deleted.
3. It will happen again to Eden space when next time GC will be triggered. But, in this case, all referenced objects are moved to S1 survivor space. In addition, objects from the last minor GC on the S0 survivor space have their age incremented and get moved to S1. Now both Eden and

S0 will be cleared, and this process will repeat every time when GC is triggered. On every GC triggered, survivor spaces will be switched and object's age will be incremented.

4. Once the objects reach a certain age threshold, they are promoted from young generation to old generation. So, this is how objects promotion takes place.
5. The major GC will be triggered once the old generation completely fills up.

Available Garbage collectors in Hotspot JVM

1. Serial Garbage Collector: Serial GC designed for the single-threaded environments. It uses just a single thread to collect garbage. It is best suited for simple command-line programs. Though it can be used on multiprocessors for applications with small data sets.

2. Parallel Garbage Collector: Unlike Serial GC it uses multiple threads for garbage collection. It is a default collector of JVM and it is also called the Throughput garbage collector.

3. CMS(concurrent mark & sweep) Garbage Collector: CMS uses multiple threads at the same time to scan the heap memory and mark in the available for eviction and then sweep the marked instances.

4. G1 Garbage Collector: G1 Garbage collector is also called the Garbage First. It is available since Java 7 and its long-term goal is to replace the CMS collector. The G1 collector is a parallel, concurrent, and incrementally compacting low-pause garbage collector.