Two major memories created and managed by JVM at runtime in the computers RAM.

1. Stack memory
2. Heap memory

Stack memory:

* is used to store method’s temporary variables.
* Used to store primitives
* Used to store reference of heap objects
  + Strong reference
    - GC cannot free this memory until the object is unreferred.
  + Week reference
    - GC can free this memory when it needs.
  + Soft reference
    - GC can free this memory, when it is very urgent need of memory space.
* Each thread has its own stack memory
* All the threads share a same heap memory
* When stack memory is full, java throws java.lang.StackOverflowError.

Heap memory:

* When we create a new object, the object is stored in the heap and its reference is stored in stack memory of the method where it is created.
* Heap is divided into two parts
  + Young generation
    - Eden
    - S0
    - S1
  + Old generation
* When we create object, it is created in Eden
* When garbage collector runs, it removes the unreferenced objects, and moves the referenced objects to S0 with age 1, this process is minor GC, this happens very periodically.
* When GC runs next time, it removes the dereferenced objects in Eden, s0 and moves the remaining objects to s1 and increases their age to 2.
* And the next time, based on the promotion age, it moves to older generation area.
* When Heap memory is full, java throws java.lang.OutofMemoryError.

Meta space:

* Is not part of heap memory
* Used to store class variables (static variables), constants.

String constant pool:

* Is a special memory under the heap area, to store the String objects without duplicates.
* When two string objects are created with same value, then those two refrence will point to same object in the string pool, for efficiency.

Garbage collector:

* When scope of a method is over, all the references in that method memory are deleted from the stack area.
* Then what about the objects in the heap area?
* When the references are cut from the stack area to heap area, the left over objects become eligible for garbage collection.
* We can call the System.gc() method or thr GC will run on its own to clear all the left over objects and free up memory space.
* Calling System.gc() is not guaranteed to run the garbage collector at that point of time.