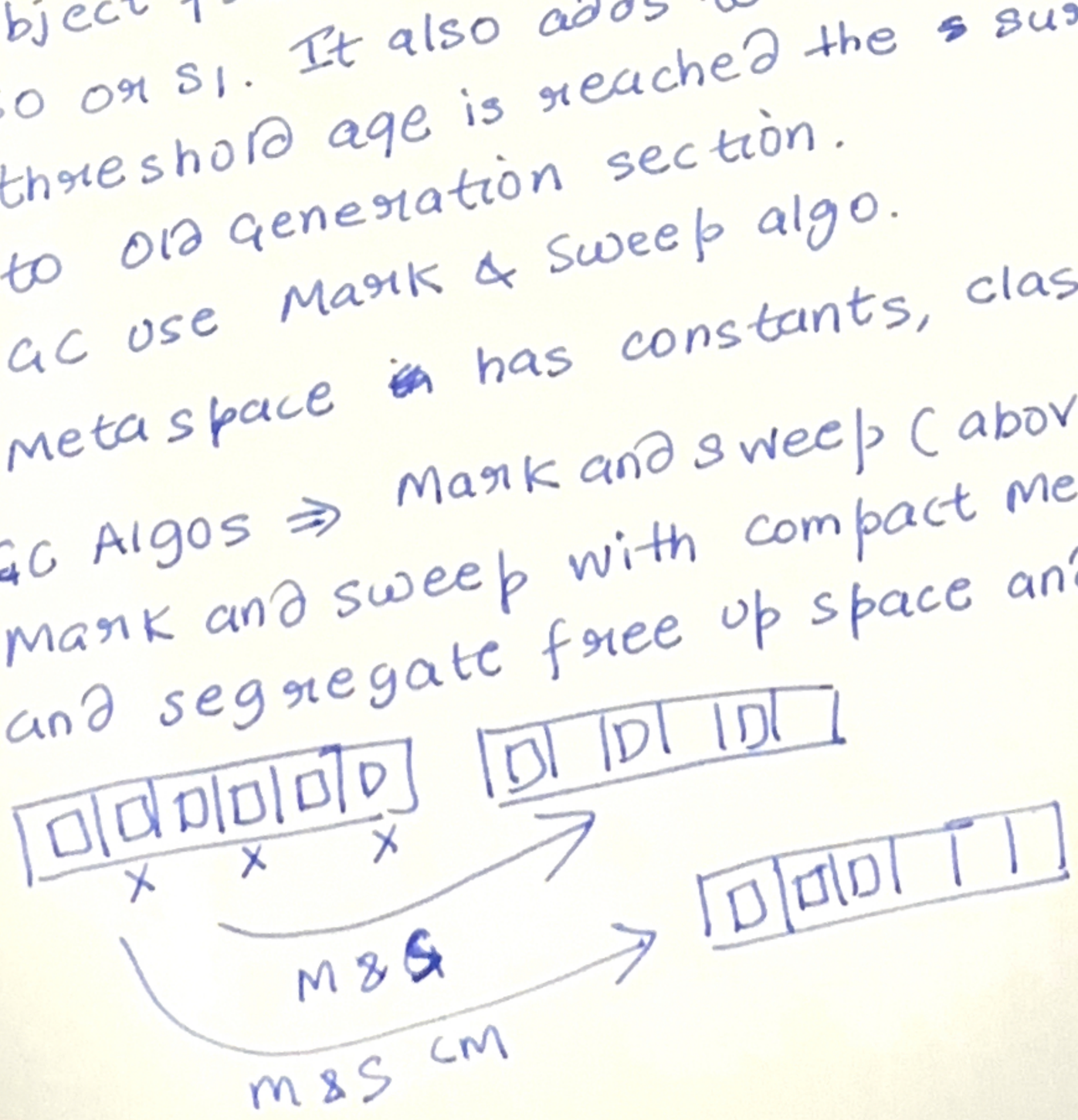
* Each thread has its own stack memory.
* once last bracket is reached all the references are deleted but heap memory is not deleted. That when garbage collector comes into play.
* Strong reference ⇒ Object obj = new Object().
  GC will not delete this object from until the reference is deleted.
* Weak Reference ⇒ WeakReference <Object> WR = new WeakReference <Object> (new Object()
  Will survive only till garbage collector doesn't runs.
  Once GC runs, this will get deleted from the heap.
* Soft Reference ⇒ kind of weak Reference, but GC will only delete it if it is very urgent.
* Heap ⟶ Young generation → Eden, S0, S1. + metaspace.
  ⟶ Old Generation

Eden stores new objects. Onee GC runs it deletes object from Eden, S0/S1 and places them in either S0 or S1. It also adds to their ages. Once some threshold age is reached the survivors are promoted to Old generation section.

GC use Mark & Sweep algo.

metaspace has constants, class variables (static)

* GC Algos ⇒ Mark and sweep (above)
  Mark and sweep with compact memory. ⇒ Mark object and segregate free up space and occupied space



* GC versions :-

- •) Serial GC ⇒ only one thred runs to delete object
- •) Parallet GC ⇒ mutiple threads runs to delete objs.
- •) Concurrent GC ⇒ GC runs without pausing application threads.
- •) G1 GC ⇒ Concurrent GC + compact memory.

*) When GC runs application threads are paused.