21/07/2020:-

Garbage Collection:-

Forms of Garbage Collection

- Do Nothing
- Reference counting
- mark and Sweep
- Copying
- Generational
- Incremental

Reference Counting:- Onus on client to call methods when allocating/free mem

Mark and Sweep:-

- → mark phase that identifies the objects that are still in use
- → Sweep phase to remove unused objects
- → Compact phase to Compact the memory

Copying:- uses different spaces to manage memory

Generations:- Maintain different generations for memory

- → promoted

Basic Idea

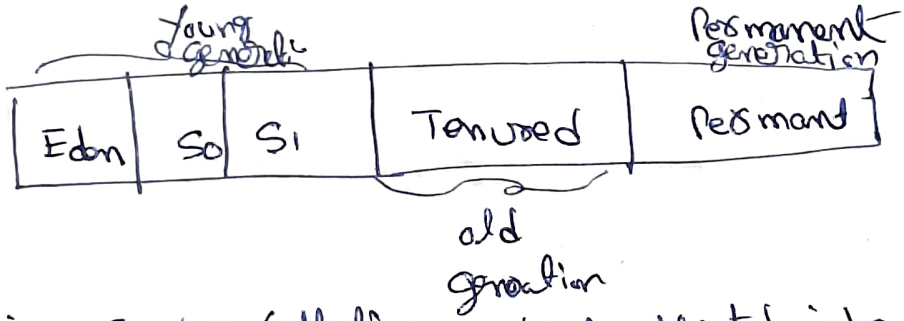
-> It has a 'young generation' and an 'old generation'

-> most initial objects allocated in 'Eden Space'

-> Young generation has two Survivor spaces

- -> Objects that survive a GC get moved to the Survivor space
- -> only one Survivor space in use at a time
- -> object copied between Survivor space

-> old generation is where long lived objects go to die



Minor Garbage Collection:- object allocated into Eden Space

when Young generation is full minor will happen

- -> GC runs objects are copied to 'newer' Survivor space
- -> objects from older Survivor space also copied to 'newer' Survivor space
- -> Survivor spaces are swapped

when older generation is full -> happen

Major Garbage Collection:-

-> Triggered when the tenured space is full

-> Collects old and Young generations

-> Although this is really a full GC

Copying to old generation:- JVM will eventually promote to old generation

-> After a certain number of garbage collects

-> If Survivor space is full

-> if JVM has been told to always create objects in old space

Allocating to old space :- we can allocate directly to old space No JVM option is required.

Allocation :-
→ Thread Local Allocation Buffers (TLABs) used
→ Each thread gets its own buffer in the Eden space
→ No Locking required

Write Barriers :-

Card table :- Each write to a reference to a young object goes through a write barrier(), This barrier updates a card table entry. one entry per 512 bytes of memory.

Different Garbage Collectors :-

i) Serial generational Collectors (use serial GC)
+ use Parallel (GC)
+ use parallel old GC

ii) + use Conc Mark Sweep GC
- use Par New GC
+ use Conc Mark Sweep GC
+ use par new GC
+ use G1 GC

Serial Collector :- → Single threaded
→ mark and Sweep
→ Small applications running on the client

Parallel Collectors :- → Multiple threads for minor Collection
→ Single thread for major Collection
→ Same process as serial, use on servers

Parallel old collector:- multiple threads for minor and major collections
Preferred over ParallelGC

Concurrent Mark and Sweep:

only collects oldSpace
no longer bump the pointer allocation
Cause heap fragmentation
Designed to be lower latency.

Phases:-
→ initial mark
→ Concurrent mark
→ Remark
→ Concurrent Sweep
→ Resetting.

G1 Collector:- ~~Off~~) new in java 6 (officially supported in java 7)
→ Is a Compacting Collector
→ planned as a replacement for CMS (Default 'server' collector in java 11
→ meant for Servers applications (multiprocessor machine with large memories)
→ Breaks heap into regions (Still has Concept of Eden Survivor and Tenured spaces)

Garbage collection Tools:-

→ MX Beans (For Garbage collector) → name of collector, collections
→ Time of collections

→ JStat → Command line tool Provided with the JVM

Java References:

Usage of Reference types:- → weak Reference (WeakHashMap) meta data with another type
→ SoftReference Can be used for caching
→ Phantom Reference → Interaction with the garbage collector.