Java 8 still widely used

- Course runs on Java11

Things changed in Java7

- G1 garbage collector introduced

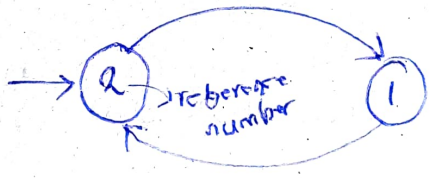
Things changed in Java9

- CMS garbage collector deprecated
- Finalizers deprecated
- Cleaner introduced

Forms of Garbage collection:-

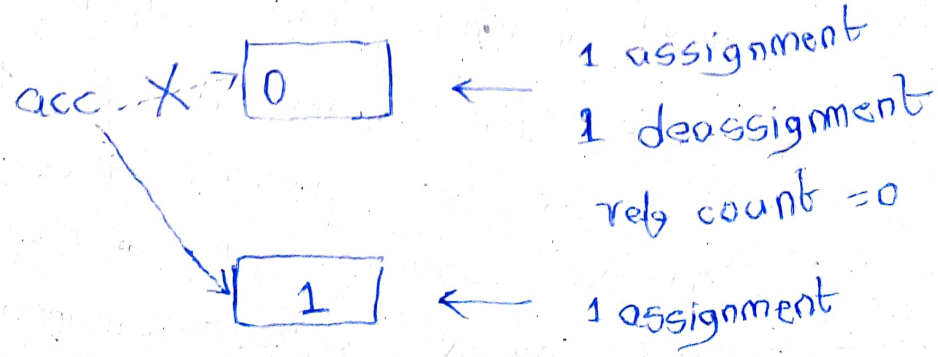
→ one type → Do Nothing
→ Reference Counting
→ Mark and Sweep
→ Copying
→ Generational
→ Incremental

Reference Counting :-



Account acc = new Account();

acc = new Account();



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

→ copying Form space to Tospace

Generational Collectors :-

Maintain different generations for memory

- Long living objects 'promoted' to different generation
- For a given definition of 'long'

How GC works in the Oracle JVM :-

Has a 'young generation' and an 'Old gen"

Most initial objects allocated in 'Eden space'

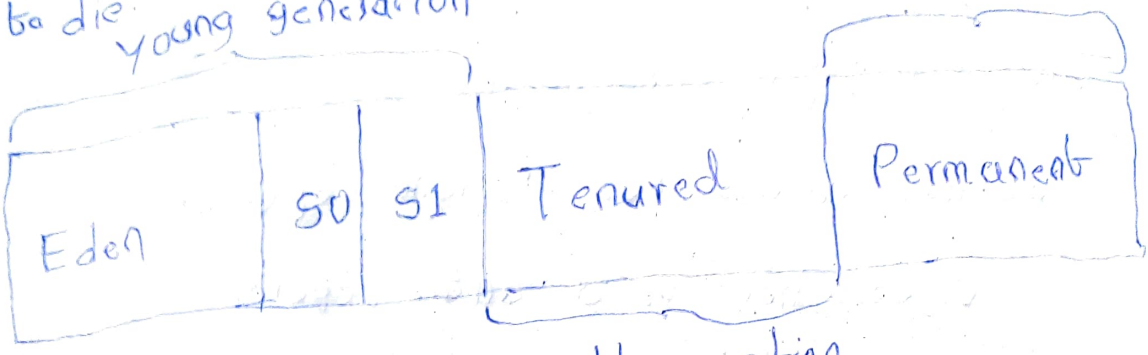
Young gtion also has two 'Survivor' spaces

→ Objects that survive a GC get moved to the survivor space

→ Only one survivor space in use at a time

→ Objects copied between Survivor spaces

Old generation is where long lived objects go to die.



: Memory layers

→ java provides no of garbage collectors it is difficult to pick a collector.

Garbage collection tools :-

→ profile the application under as close to production load as possible

Test under different garbage collectors

Garbage collection tools :-

MXBeans :-

There is a java MX Bean for the Garbage collector

- Name of collectors
- Number of collections
- Time of collections

JStat :-

Command line tool provided with the JVM

→ This tool is useful to monitor the no of garbage collection happening and how quickly

They are happening.

Java Reference Classes:-

Java has always had 'strong' references

→ Object not GC'd until references are released.

Other types of references are availble

- 'special' class in java.lang.ref package
- Soft, Weak and Phantom references

Reference Rules:-

Strong → Soft → Weak → Phantom
ref > ref > ref > ref

Object not GC if there is a strong refernce

→ But can be GC'd if there is a soft, weak or phantom refernce

→ Soft referenced objects will be collected if there is memory pressure

→ Weak will be collected immediatly

→ phantom refereces different to the other two

Usage of Reference Types

Weak Reference
- Associate meta data with another type
- Use WeakHashMap

Soft Reference
- Can be used for caching

PhantomReference
- Interaction with the garbage collector

→ Soft Reference Caching
Hold a softReference to an object
- as well as a strong reference
When strong reference is cleared soft is still availble so you can retrive the object

problem
Not always a great mechanism
→ No control over cache

WeakHashMap:-
Like a HashMap
Key is a Weak reference to an object
- Store a weak reference to an object as a key
- Value is the object's 'meta data'