**JAVA MEMORY MANAGEMENT**

Java memory is divided into 2 parts

1.Heap Memory 2. Stack Memory

Heap Memory :

Heap Memory again divided into 2 parts

1. Young Generation
2. Old Generation

Young Generation again divided into 3 parts

1. Eden Memory
2. Survivor Memory (S0)
3. Survivor Memory (S1)

Once we created objects, it will store first in Young Generation Eden Memory, if Eden Memory filled up then Minor Garbage Collector will perform action, that it shifts object from Eden to S0, If S0 also filled then to S1.

Internally so many garbage collectors will call, in between JVM will send the important objects to survivor Memory and remaining objects which are blank to gc()- which is minor

Old Generation:

When young generation is completely filled, then new objects start creating memory in old generation. If old generation also filled, then its time to call gc(), which is major one.

It takes long time to perform action.

Eg: In flipkart holiday season at a time many customers hit the same, then they may get timeoutexception because of this

It is always good to call minor gc() than major gc(), because of this so many objects may destroyed.

\*\*We have another memory location i.e is **Memory Pool** inside heap memory.

Memory pool is used to store immutable objects, String class, String pools

They are defined at runtime by JVM memory managers.

There is a another part called “Permanent Generation” which is separate from Heap area

* It will store all meta data of classes, methods
* It is not inside part of Heap Memory, it is separate
* It will be generated at the time of runtime

There is a another **Method Area** inside Permanent Generation which it stores class structure, Static variables, constants.

Inside Method Area, there is a **‘Run Time Constant Pool’** in whichStatic variables, constants are stored.

Finally, Heap memory stores

1. All objects of classes
2. JRE Classes
3. GC runs on heap memory only to free some memory by destroying some objects which don’t have any references.

Stack Memory:

🡪Stack Memory is used to execute threads

🡪Stores method specific values

* Stores local variables (int i=0)
* Stores Object references

i.e it stores object references here, which refer its object in heap memory

Test t1 = new Test();

t1 -> store in stack

Object new Test() 🡪 store in heap memory

In stack, the memory is filled on the basis of LIFO – Last In First Out

\*\*In stack memory, there is a method block, which gives space to particular method when the time of execution. If there are m1,m2,m3 methods.

The method block is assigned to m1 first and after completing its execution, it will free up and give space to m2 for execution….

* Stack memory is less size compared to heap memory

Heap memory stores only objects

Stack memory stores local variables, constant variables, methods, static variables, method specific variables

