Garbage Collector in java **

Unit - 1

- GC is a feat. of java wh. is used to release unused m/m.

How is m/m allocated & released for data members / variables

- PDT r static data members. Their m/m is allocated at compile time.
- obj, str, arr are ~~ref~~ ~~var~~ dynamic data members. m/m is allocated at run time.

- For static data members m/m allocation is done autoy by OS.
- The static members r also destroyed autoy by OS

- However for dy. data members, m/m allocation needs to be done by user using new operator
- Similarly for dy. data members, m/m is released by the GC when they r no longer needed.

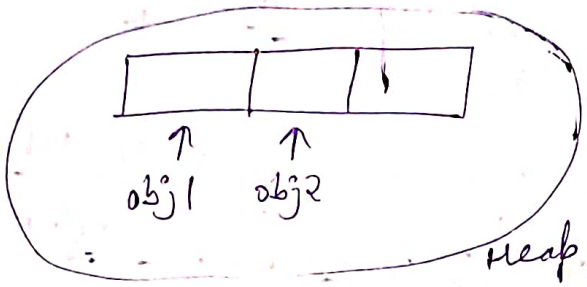
For static var :-
Allocation } done by OS
m/m released }

For ~~data members~~ dy. var :-
Allocation → done using new opr.
m/m released → autoy by GC when no longer needed.

Q) How does GC works?

- GC releases the m/m of obj when they r no longer needed.
- GC runs periodically & checks if there is any ~~obj~~ m/m w/ no ref. If such m/m is found then GC releases the m/m

eg

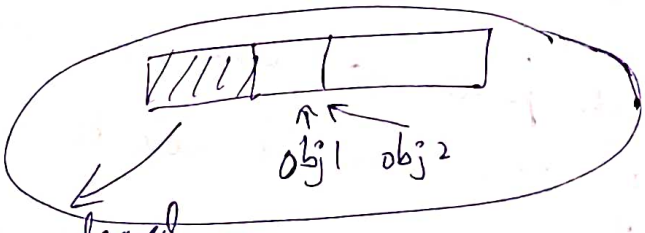


- Suppose 2 obj r created obj1 & obj2
- obj1 & obj2 r ref var. They point to m/m location

- Suppose we run the statement

  obj1 = obj2;

Now both ref pt. to the same m/m loc.



- The I^st m/m loc has no ~~obj~~ ref. pointing to it.

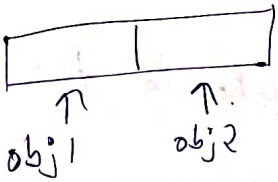
- GC runs periodically & unpredictably.
- When GC runs it finds no ref to the 1st m/m loc. So GC releases that m/m.

Feat. of GC

- It runs periodically
- It is unpredictable when it will run.
- It releases the m/m when no ref. to the m/m is found.
- It releases/destroys dy. objects, strings, arr.
- It helps to automate & improve m/m management

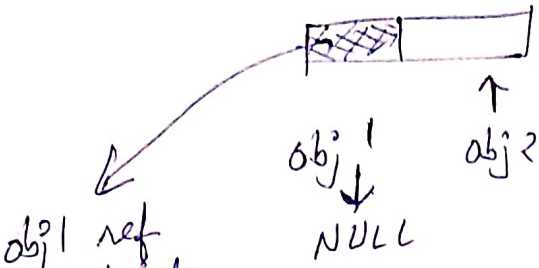
Another eg. of GC

- m/m can also be released by nullification of a ref var.



- obj1 & obj2 both point to some m/m loc.
- If we nullify obj1, then it will be released by GC.

```
obj1 = NULL;
```



// obj 1 now points to NULL

obj 1 ref var points to NULL, so this m/m will be released.