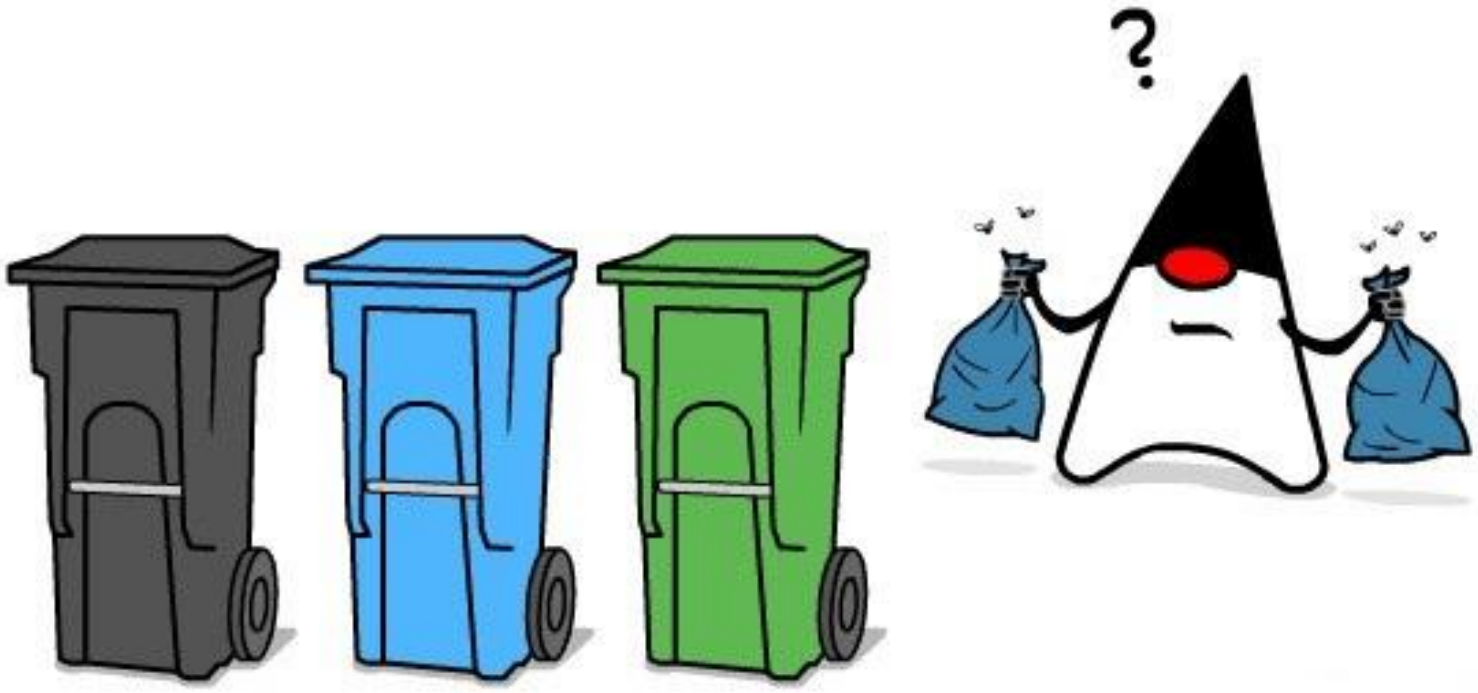
Dot Net Framework

Garbage Collector



Garbage

Garbage Collector

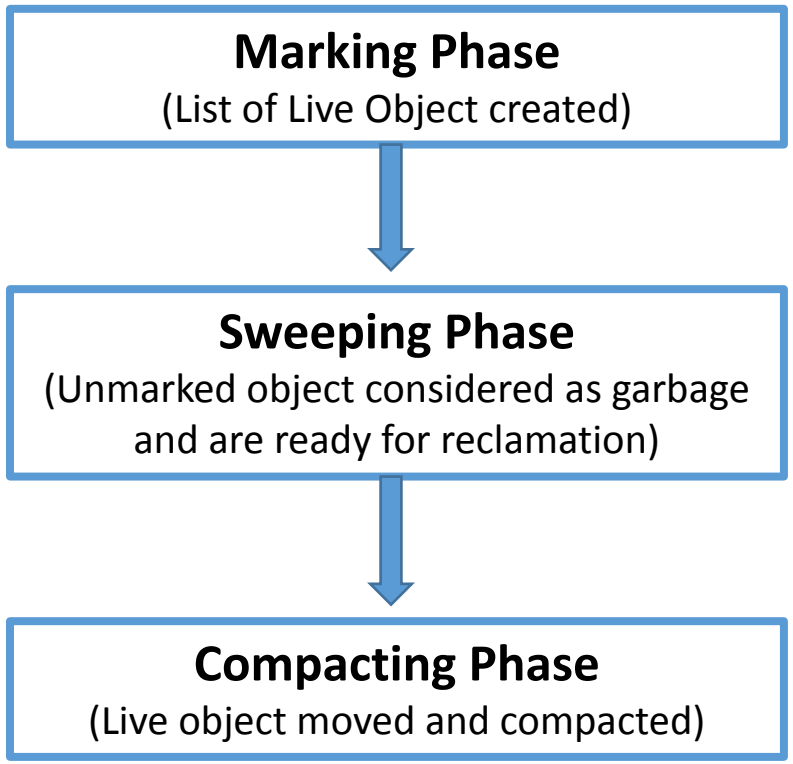

Int a=10;
a
10

Garbage Collector

The garbage collector provides the following benefits:

- Frees developers from having to manually release memory.
- Allocates objects on the managed heap efficiently.
- Reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations. Managed objects automatically get clean content to start with, so their constructors don't have to initialize every data field.
- Provides memory safety by making sure that an object cannot use the content of another object.

Phases in Garbage Collection



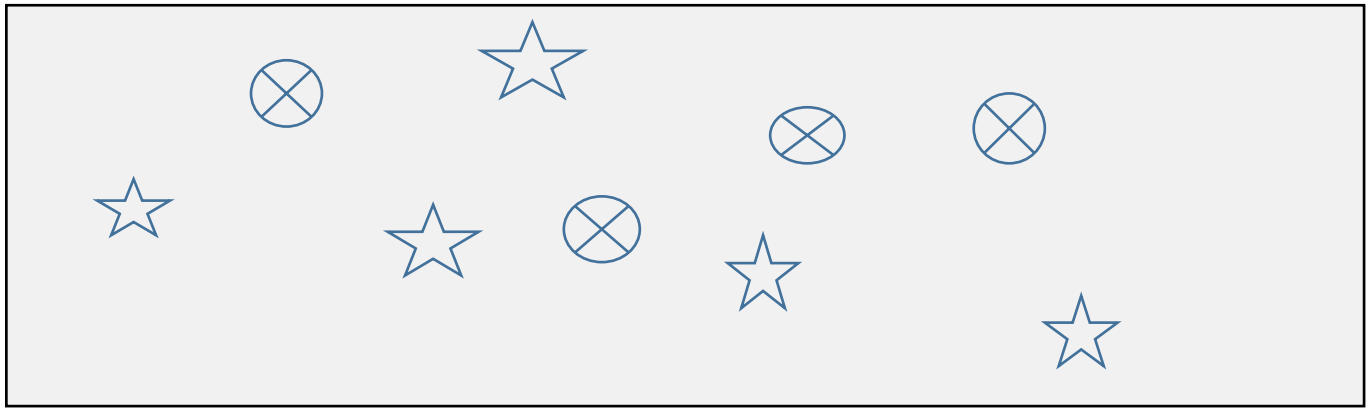
Phase -1

Phase -2

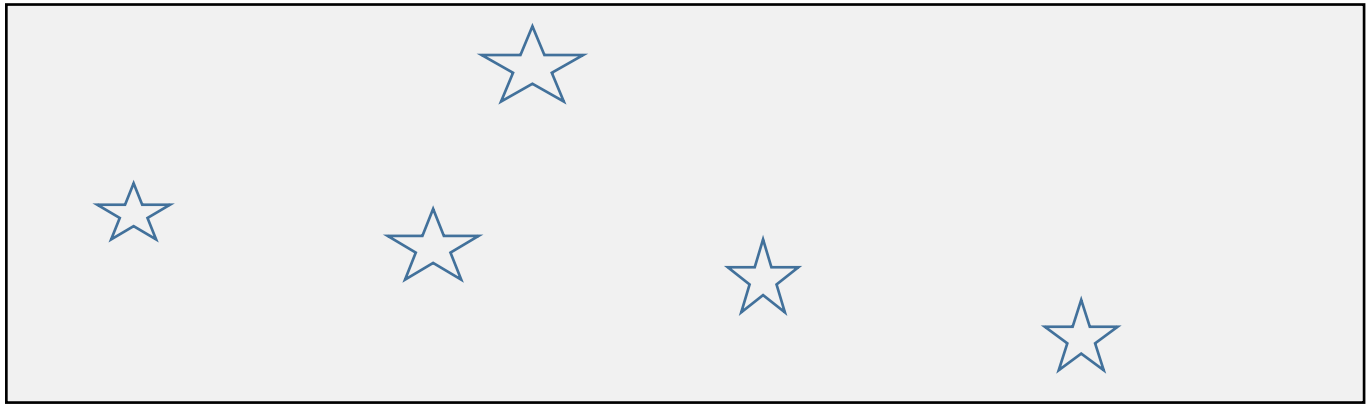
Phase -3

Wedding Lunch Program

Lunch Ground



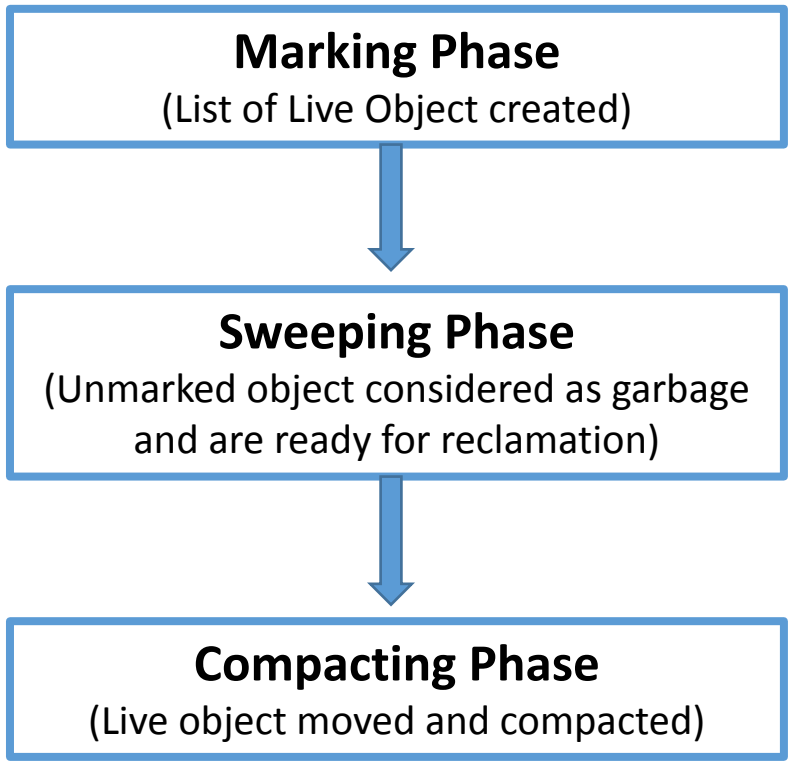
Phase -1



Phase -2

Phase -3

Phases in Garbage Collection



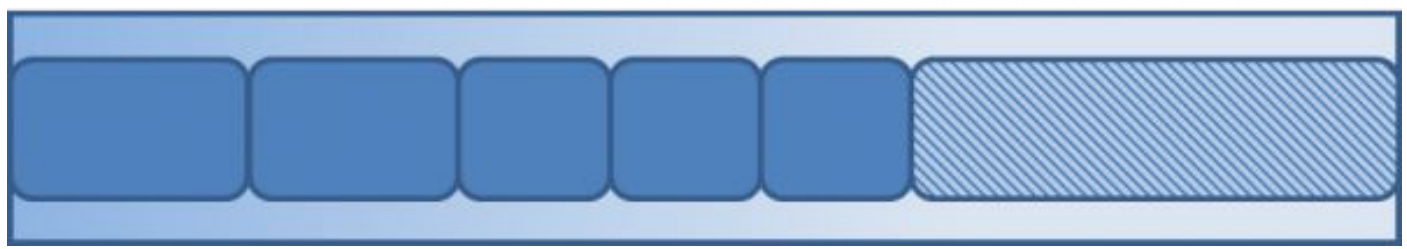
Phase -1

Phase -2

Phase -3

Fragmented Heap

Heap after Compacting



Phases in Garbage Collection

- **Marking Phase:** During a garbage collection cycle, the GC starts by marking all reachable objects. It traverses the object graph, starting from the roots (global variables, references on the stack, etc.), marking each reachable object as "in-use."

- **Sweeping Phase**: Once the marking phase is complete, the GC sweeps through the managed heap, identifying all the objects that were not marked during the marking phase. These unmarked objects are considered garbage and are ready for reclamation.

- **Compact Phase**: In the final phase, the garbage collector reclaims the memory occupied by the garbage objects. It compacts the live objects, moving them together to eliminate memory fragmentation, and updates the heap's free space.

Garbage Collector

Heap Generations in Garbage Collection

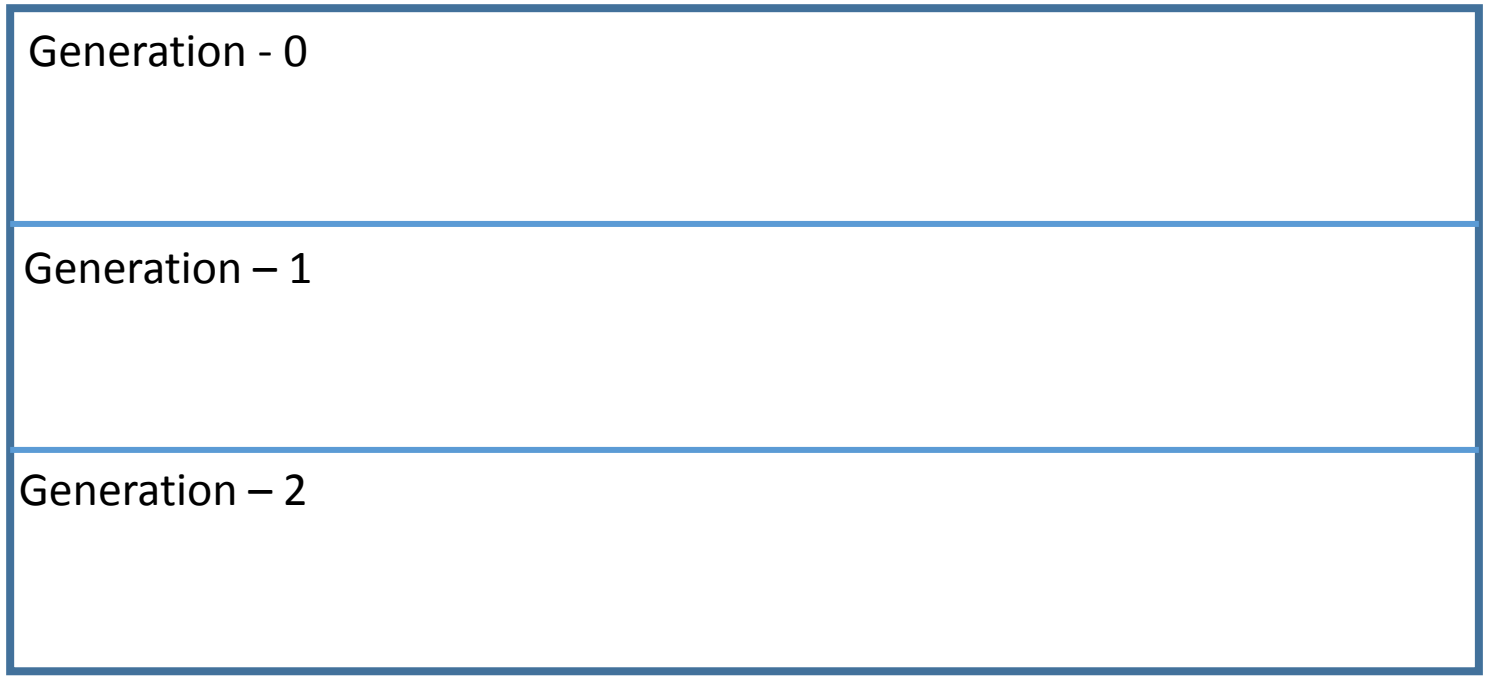
The heap memory is organized into 3 generations so that various objects with different lifetimes can be handled appropriately during garbage collection.

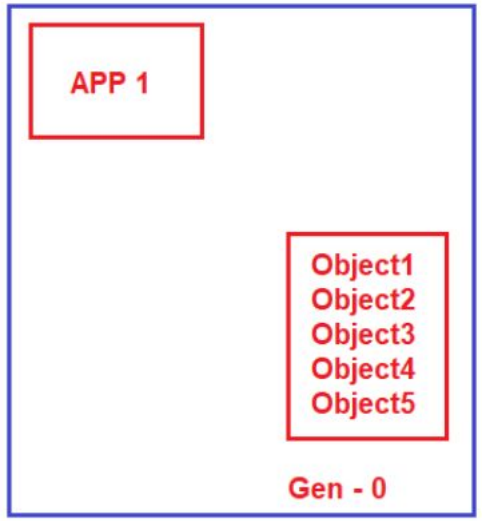
Generation 0

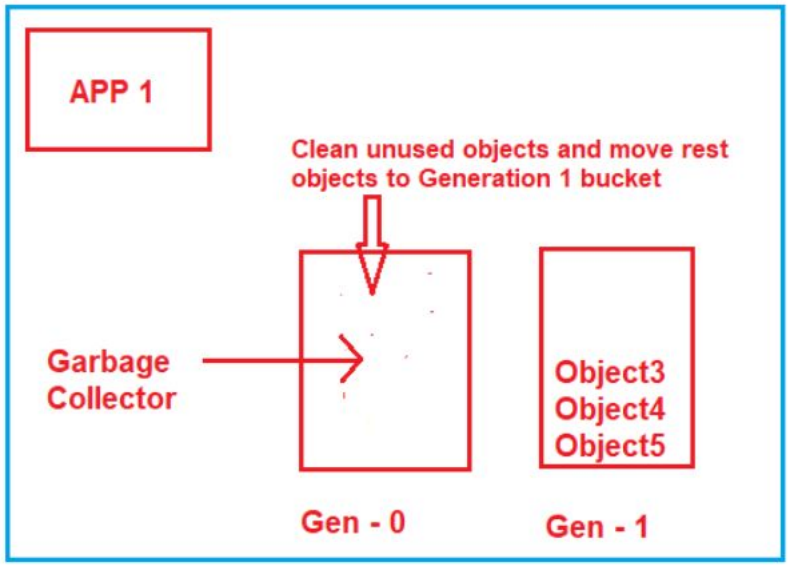
Generation 1

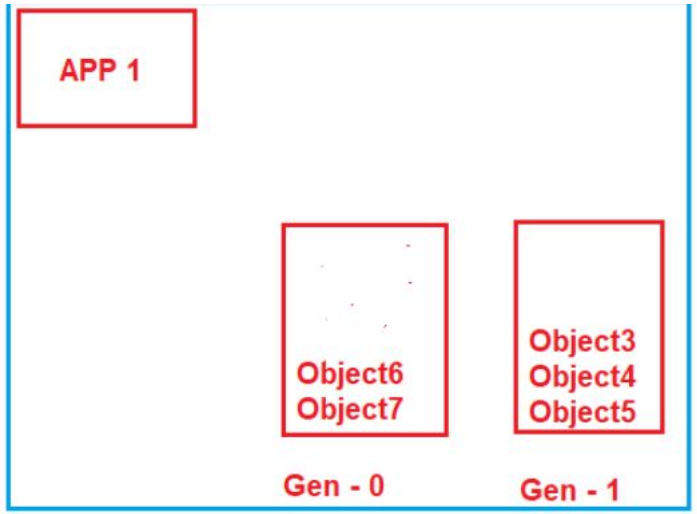
Generation 2

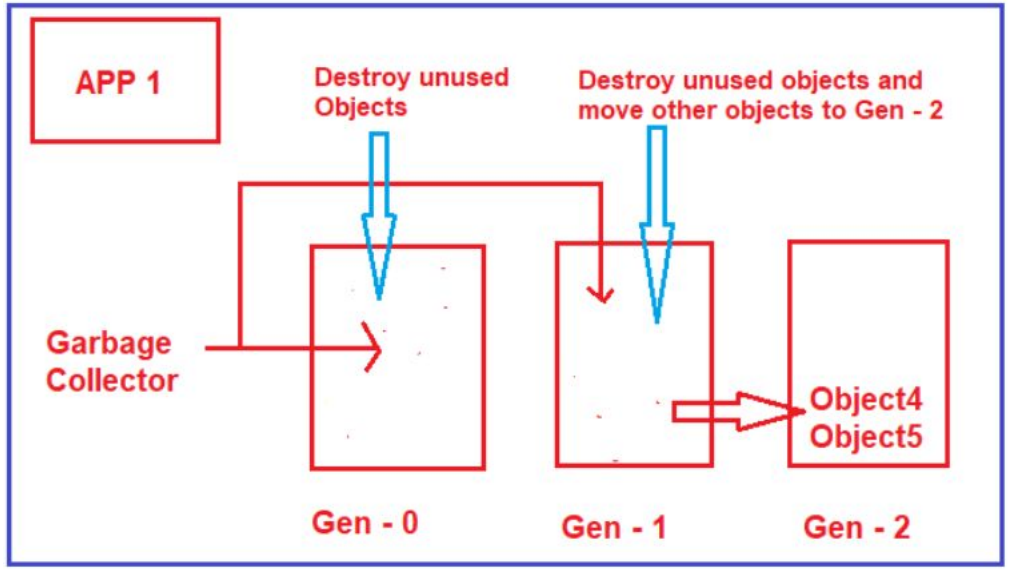
Heap Memory




APP 1
Object1
Object2
Object3
Object4
Object5
Gen - 0


APP 1
Clean unused objects and move rest objects to Generation 1 bucket
Garbage Collector
Object3
Object4
Object5
Gen - 0
Gen - 1


APP 1
Object6
Object7
Object3
Object4
Object5
Gen - 0
Gen - 1


APP 1
Destroy unused Objects
Destroy unused objects and move other objects to Gen - 2
Garbage Collector
Object4
Object5
Gen - 0
Gen - 1
Gen - 2

Why do we need Generations?

- Normally, when we are working with big applications, they can create thousands of objects. So, for each of these objects, if the garbage collector goes and checks if they are needed or not.

- it’s really a bulky process. By creating such generations what it means if an object in Generation 2 buckets it means the Garbage Collector will do fewer visits to this bucket.

- The reason is, if an object move to Generation 2, it means it will stay more time in the memory. It’s no point going and checking them again and again.