Garbage Collector

छोटा सा Introduction

By
Majrul Ansari
(Java Collector)

Source: Internet & Explorer

Stack & Heap

- Memory is basically divided into stack and heap.
- Methods are loaded in the stack. Object references are kept in the stack of target method whereas the actual object is allocated in the heap.
- An object can be created using “new” keyword or using Reflection API’s “newInstance” method.

- Java HotSpot VM uses a generational GC as default.
- The heap is divided into two physical areas, which are referred to as generations, one young and one old.

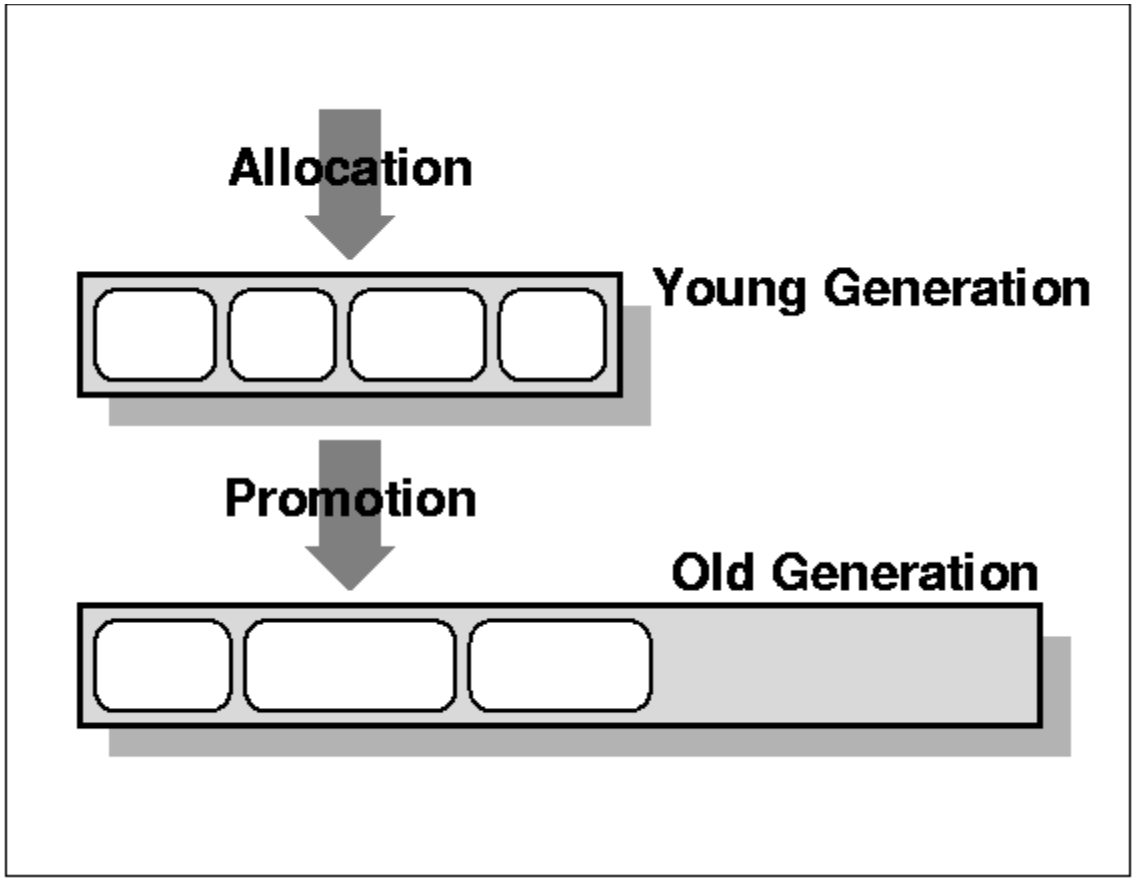
Young generation

- Most newly allocated objects are allocated in the young generation, which is typically small and collected frequently. Since most objects in it are expected to die quickly, the number of objects that survive a young generation collection (also referred to as a minor collection) is expected to be low. In general, minor collections are very efficient because they concentrate on a space that is usually small and is likely to contain a lot of garbage objects.

Old Generation

- Objects that are longer-lived are eventually promoted, or tenured, to the old generation. This generation is typically larger than the young generation and its occupancy grows more slowly. As a result, old generation collections (also referred to as major collections) are infrequent, but when they do occur they are quite lengthy.

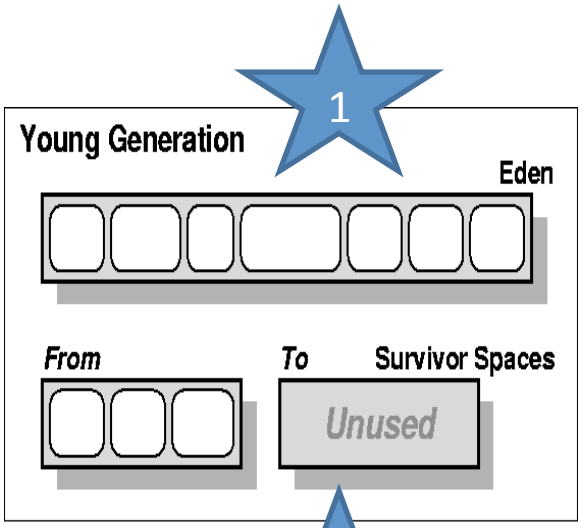
Young & Old Generation

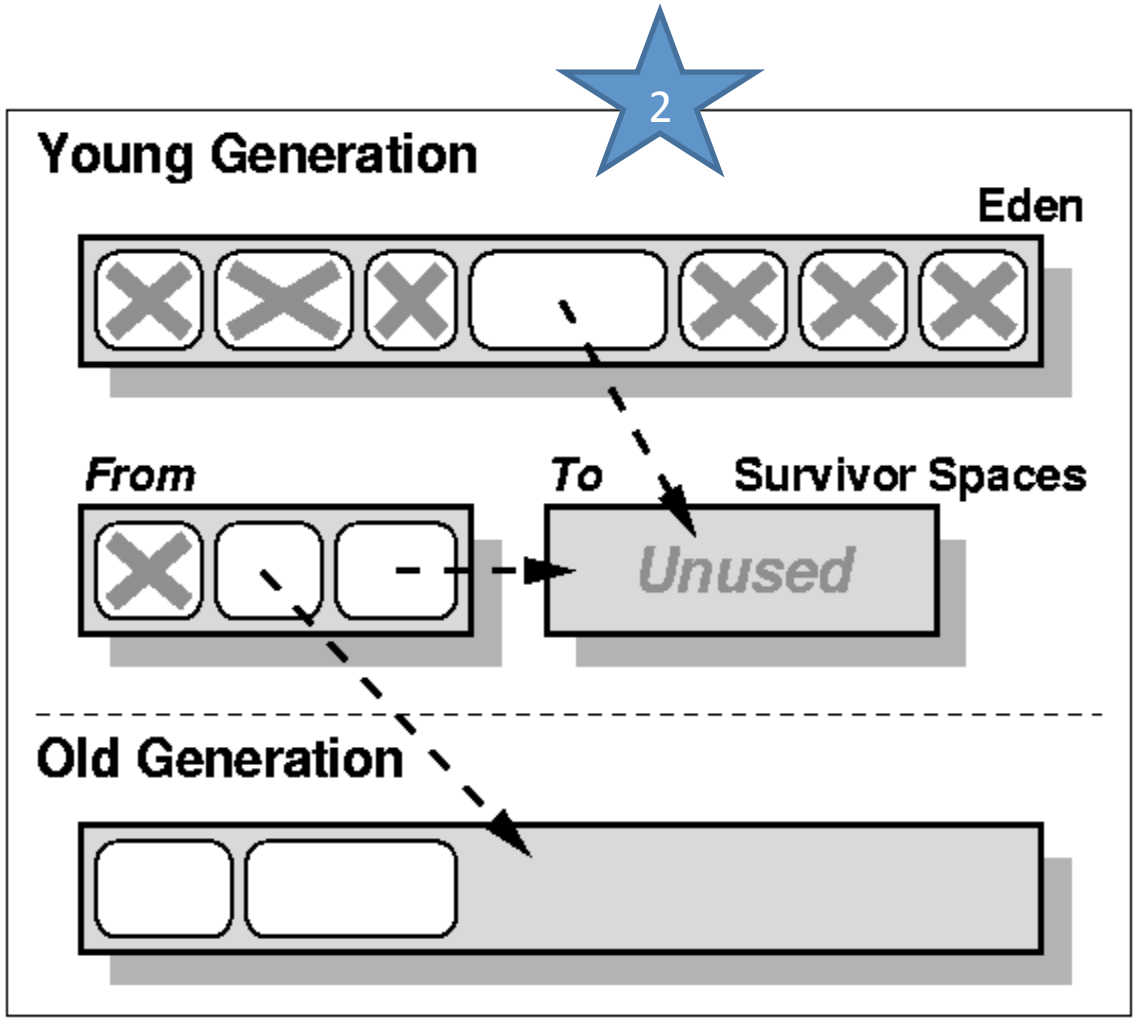


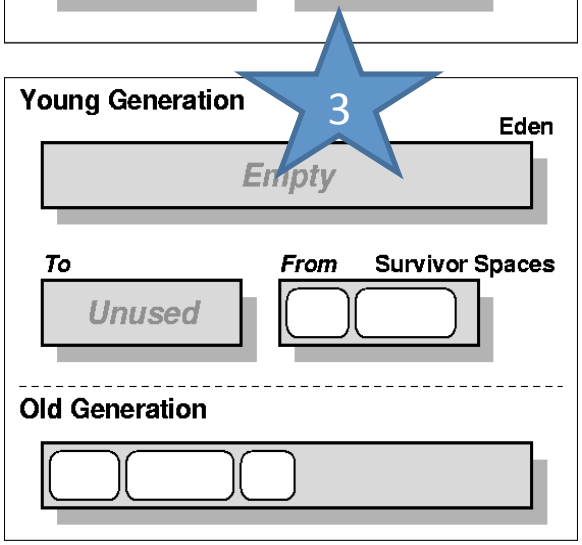
The Young Generation

- The young generation is split into three areas:
  - The Eden: Most new objects are allocated here (large objects may be directly allocated into the old generation). The Eden is always empty after a minor collection.
  - The Two Survivor spaces: These hold objects that have survived at least one minor collection but have been given another chance to die before being promoted to the old generation.

Young Generation







How minor collection works

- Live objects in the Eden that survive the collection are copied to the unused survivor space. Live objects in the survivor space that is in use, which will be given another chance to die in the young generation, are also copied to the unused survivor space. Finally, live objects in the survivor space that is in use, which are deemed "old enough," are promoted to the old generation.

Cont'd…

- At the end of the minor collection, the two survivor spaces swap roles. The Eden is entirely empty; only one survivor space is in use; and the occupancy of the old generation has grown slightly. Because live objects are copied during its operation, this type of collector is called a copying collector.
- Because live objects are copied during its operation, this type of collector is called a copying collector.

Fast Allocation strategy

- The operation of the allocator is tightly coupled with the operation of the garbage collector. The collector has to record where in the heap the free space it reclaims is located. In turn, the allocator needs to discover where the free space in the heap is before it can re-use it to satisfy allocation requests. The copying collector that collects the young generation of the Java HotSpot VM has the advantage of always leaving the Eden empty, which allows allocations into the Eden to be very efficient by using the bump-the-pointer technique.
- According to this technique, the end of the last allocated object being tracked (usually called top) and when a new allocation request needs to be satisfied, the allocator needs only to check whether it will fit between top and the end of the Eden. If it does, top is bumped to the end of the newly allocated object.

Further Reading

- The Serial Collector
- The Parallel Collector
- The Concurrent Mark-Sweep Collector