1장 +) GC

Garbage Collector (G.C)

> 옛 GC와 최근 GC와의 차이

GC 유튜브 - 테코톡

> **Mark and Sweep** 과정이라고 함

GC 과정

1. GC가 Stack의 모든 변수를 스캔하면서 각각 **어떤 객체를 참조하고 있는지** 찾아서 **마킹**
2. Reachable Object가 참조하고 있는 객체도 찾아서 마킹
3. **마킹되지 않은 객체**를 **Heap에서 제거**한다.

Heap

Young Gen (Eden, survival 0, 1) , Old Gen으로 나눠져있음

GC 과정

New Gen → Eden에 새로운 객체가 할당된다. → Eden이 다 차면 GC 발생(Minor GC)

→ Eden영역에서만 Mark & Sweep 발생 → 생존자만 Survival 0으로 이동 → 이 과정 반복

→ Survival 0 영역이 가득 차면 이 영역에 다시 Mark & Sweep 과정 → 생존자 Survival 1으로 이동후 Age 증가 → Eden에서 Mark & Sweep이 발생하는데 Survival 1으로 이동 → 가득

차면 Mark & Sweep 하고 Survival 0으로 이동 후 Age값 증가 → 위 과정을 반복할 때 특정 Age값을 넘으면 Old Gen으로 이동 → Old Gen이 가득 차면 GC 실행 (Major GC)

❓ 왜 Minor GC와 Major GC를 나누는가

GC설계에는 아래 두가지 가설을 두고 만들었다.

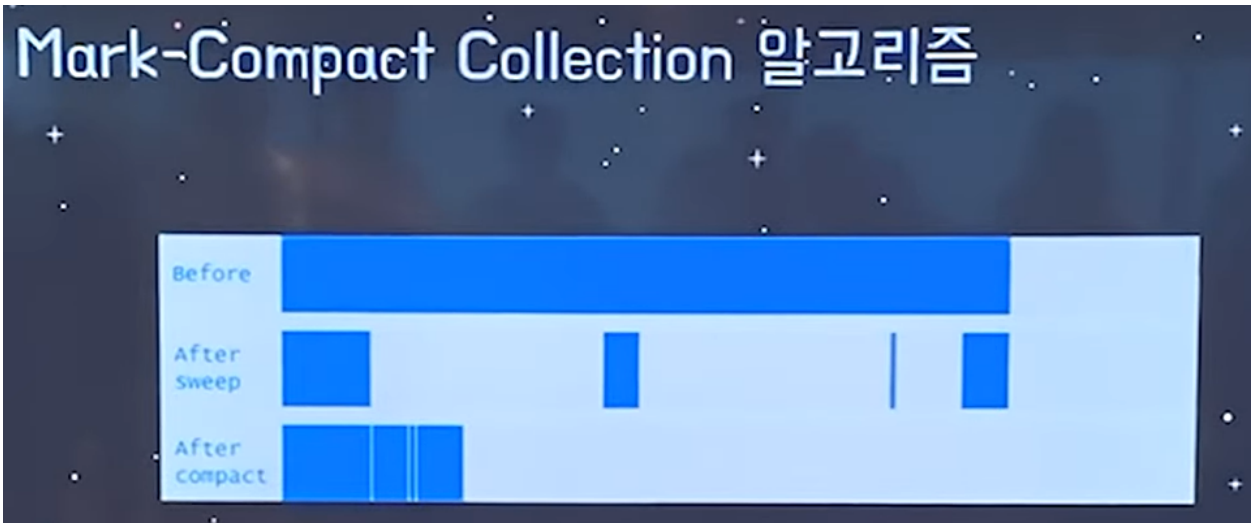
- 대부분 객체는 금방 접근 불가능한 상태(unreachable)가 된다. (= 금방 garbage가 된다)
- 오래된 객체에서 젊은 객체로의 참조는 아주 적게 존재한다.

GC 종류

Serial GC

- GC를 처리하는 스레드가 1개이다
- CPU 코어가 1개만 있을 때 사용
- Mark-Compact collection 알고리즘 사용

❓ Mark-Compact collection 알고리즘



▼ After sweep을 compact에 몰아버림

Parallel GC

- GC 처리하는 스레드가 여러개
- 속도가 빠름
- CPU 코어가 여러개일때 사용

Concurrent Mark Sweep GC (CMS GC)

- stop-the-world를 줄임으로 응답시간이 빨라진다
- 다른 GC보다 메모리와 CPU를 많이 사용한다
- Compaction 단계가 제공되지 않음

G1 GC

- 각 영역을 Region 영역으로 나눈다.
- GC가 일어날 때 전체영역을 탐색하지 않는다 → stop-the-world 시간을 줄인다
- Java 9+의 default GC

❓ Stop-The-World

- GC를 실행하기 위해 JVM이 애플리케이션 실행을 멈추는 것
- stop-the-world가 발생하면 GC를 실행하는 스레드를 제외한 나머지 스레드는 모두 작업을 멈춤
- GC 작업을 완료한 후 작업 재개

옛 GC

최근 GC