

# GC의 흐름으로 읽는 배치 처리 효율화

우아한테크코스 7기 BE 칼리

# 발표 목차

## 어떤 내용을 다룰까?

1. GC란?
2. OOM과 GC 관계
3. GC의 동작 원리
4. GC의 추이로 보는 배치 튜닝



# 1. GC란?



# 1. GC란?

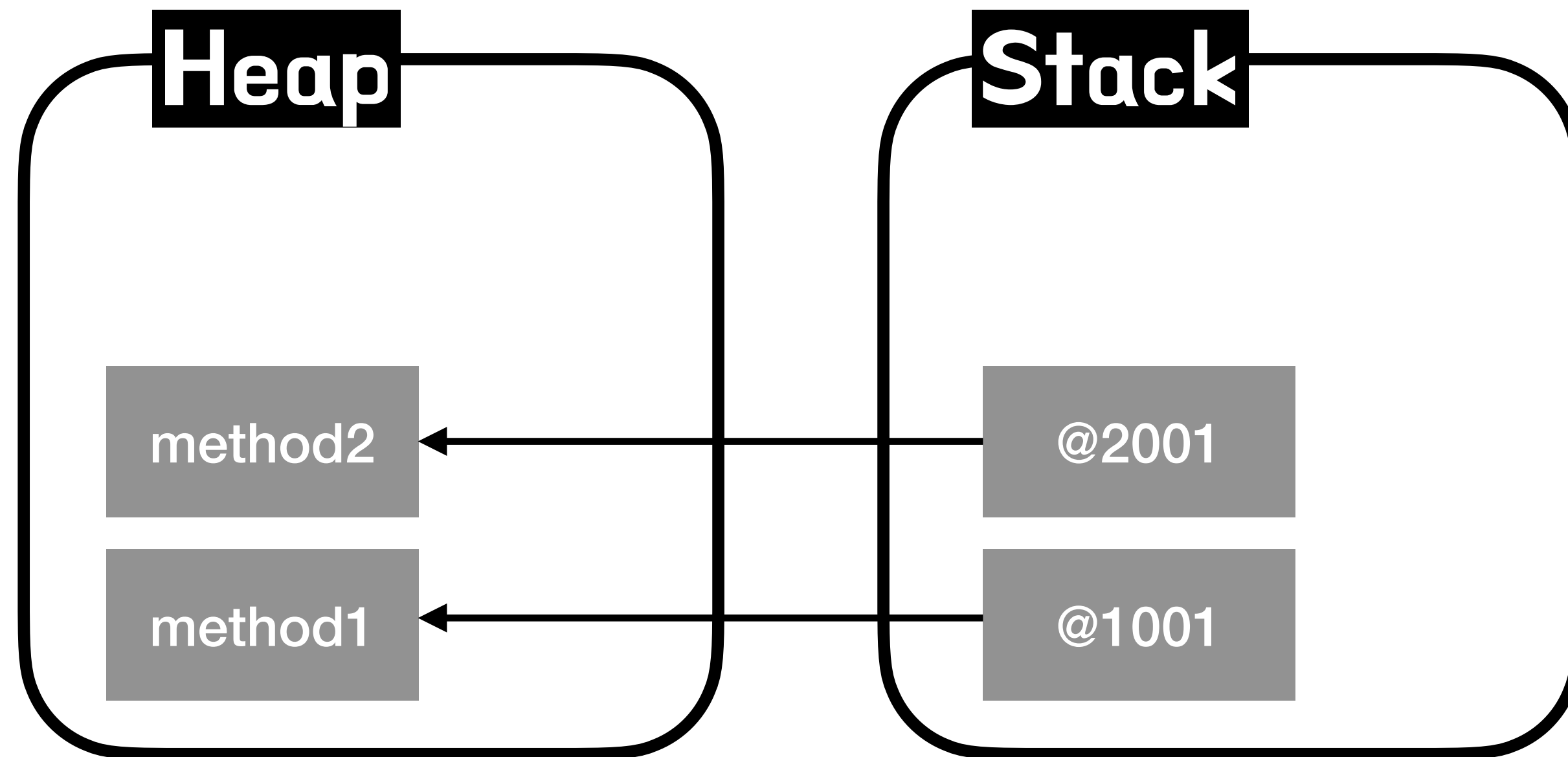
필요 없게 된 메모리 객체를 모아 주기적으로 제거하는 프로세스

Garbage Collection



# 1. GC란?

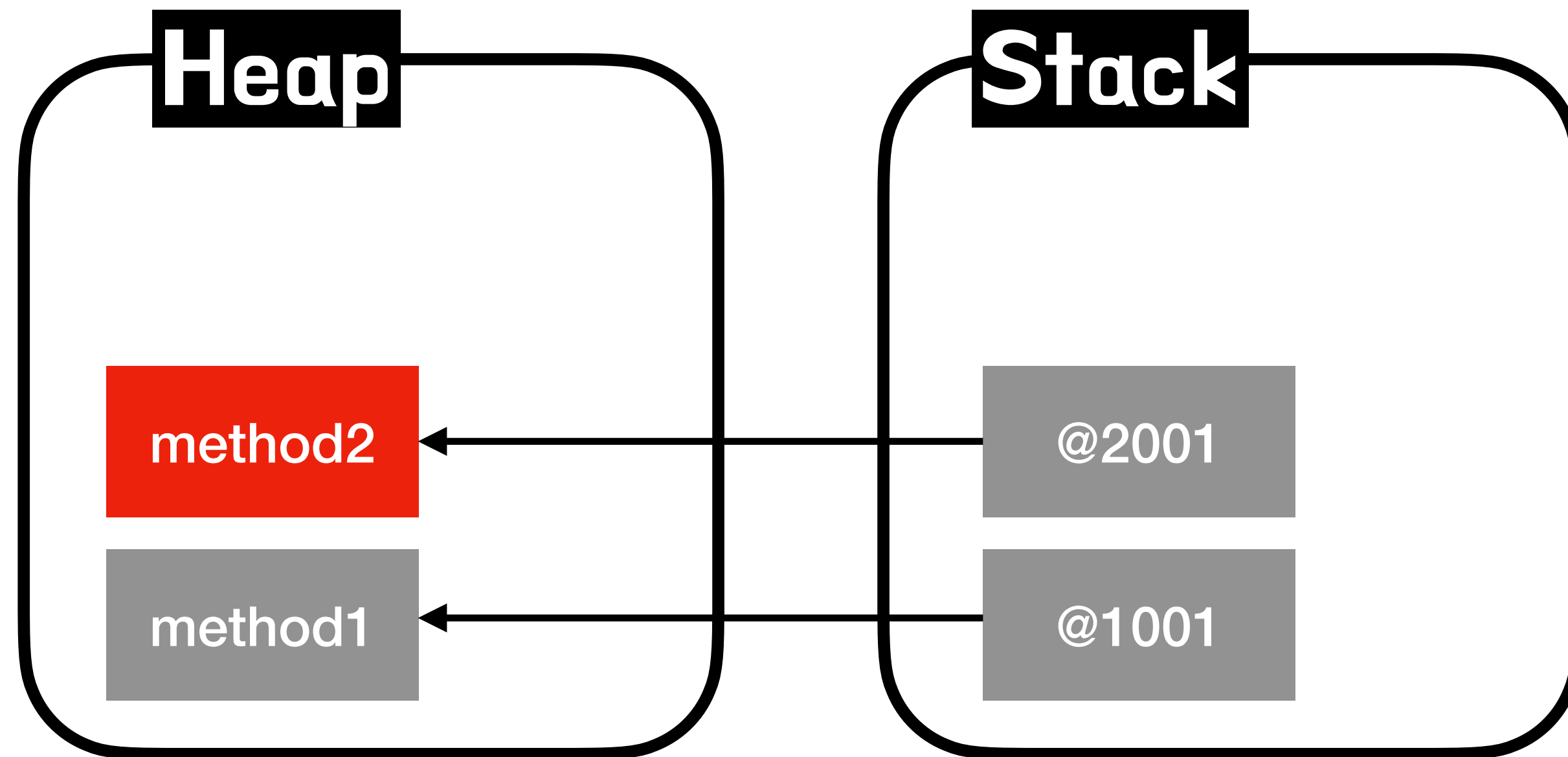
필요 없게 된 메모리 객체를 모아 주기적으로 제거하는 프로세스





# 1. GC란?

필요 없게 된 메모리 객체를 모아 주기적으로 제거하는 프로세스





## 2. OOM과 GC 관계



## 2. OOM과 GC 관계

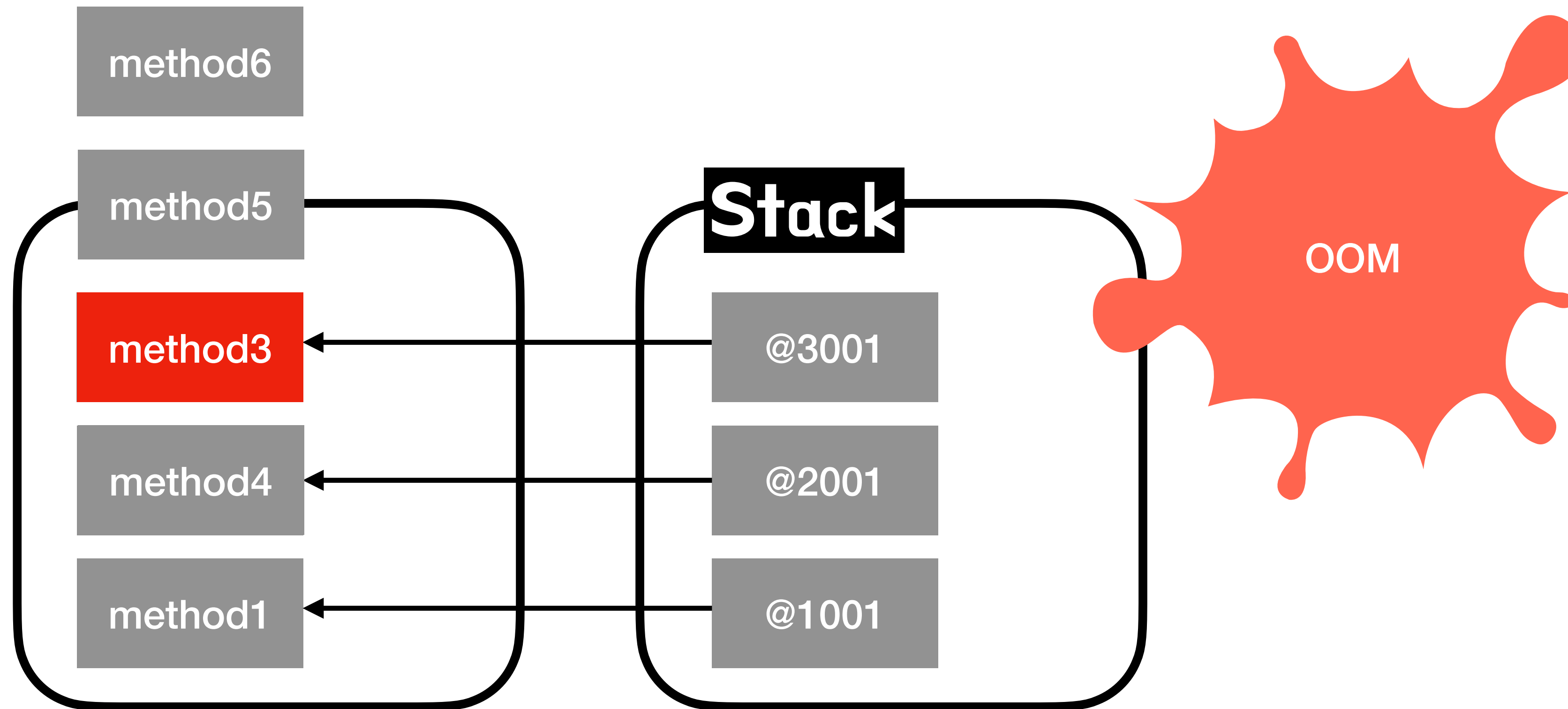
OOM(Out Of Memory)





## 2. OOM과 GC 관계

메모리 적재 속도 > GC 속도





# 3. GC의 동작 원리



# 3. GC의 동작원리

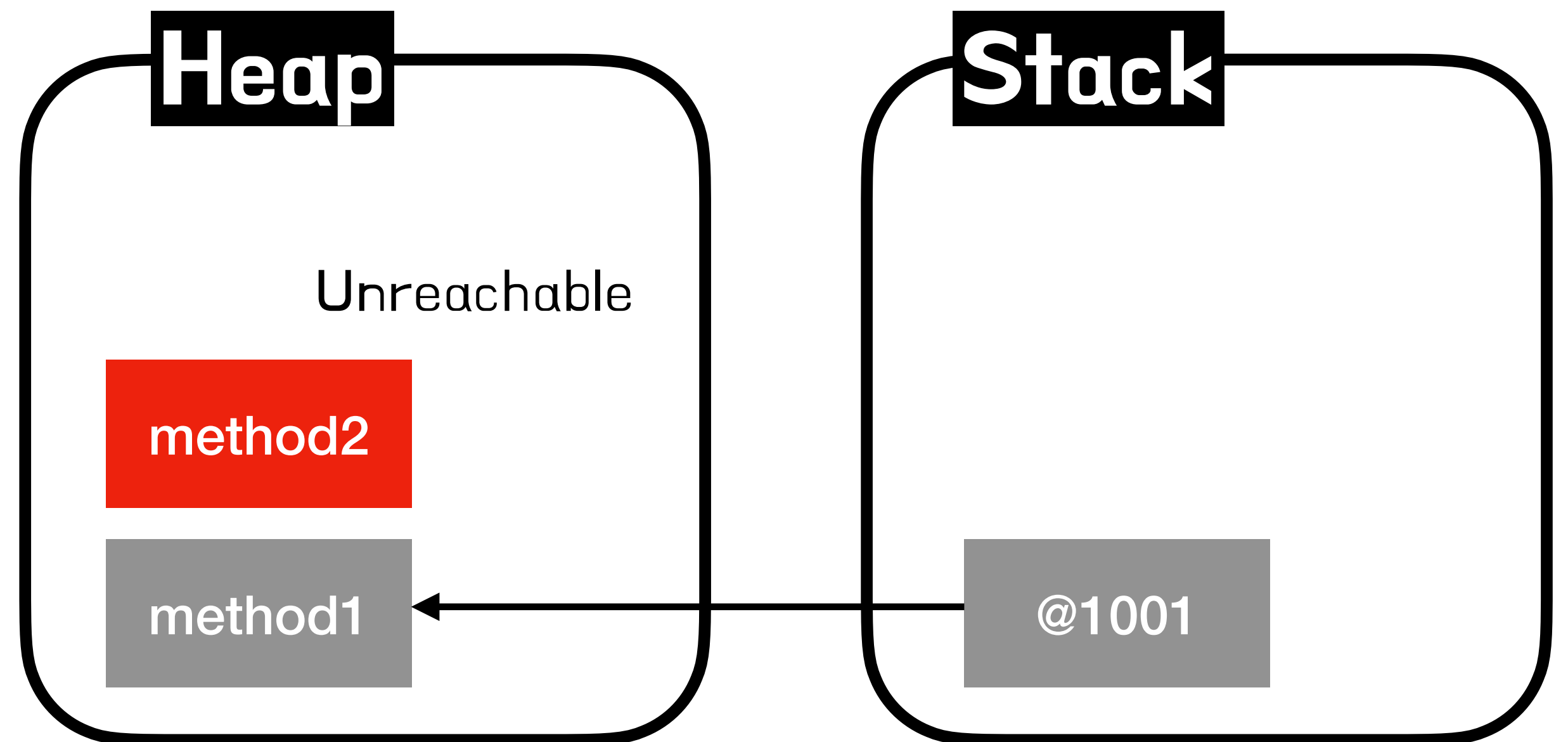
## 1) 가비지 컬렉션 대상 : Reachability

### 도달 능력 (Reachability)

: 특정 객체가 garbage 인지 아닌지 판단하는 개념

- Reachable: 객체가 참조되고 있는 상태
- Unreachable: 객체가 참조되지 않고 있는 상태

Reachable





# 3. GC의 동작원리

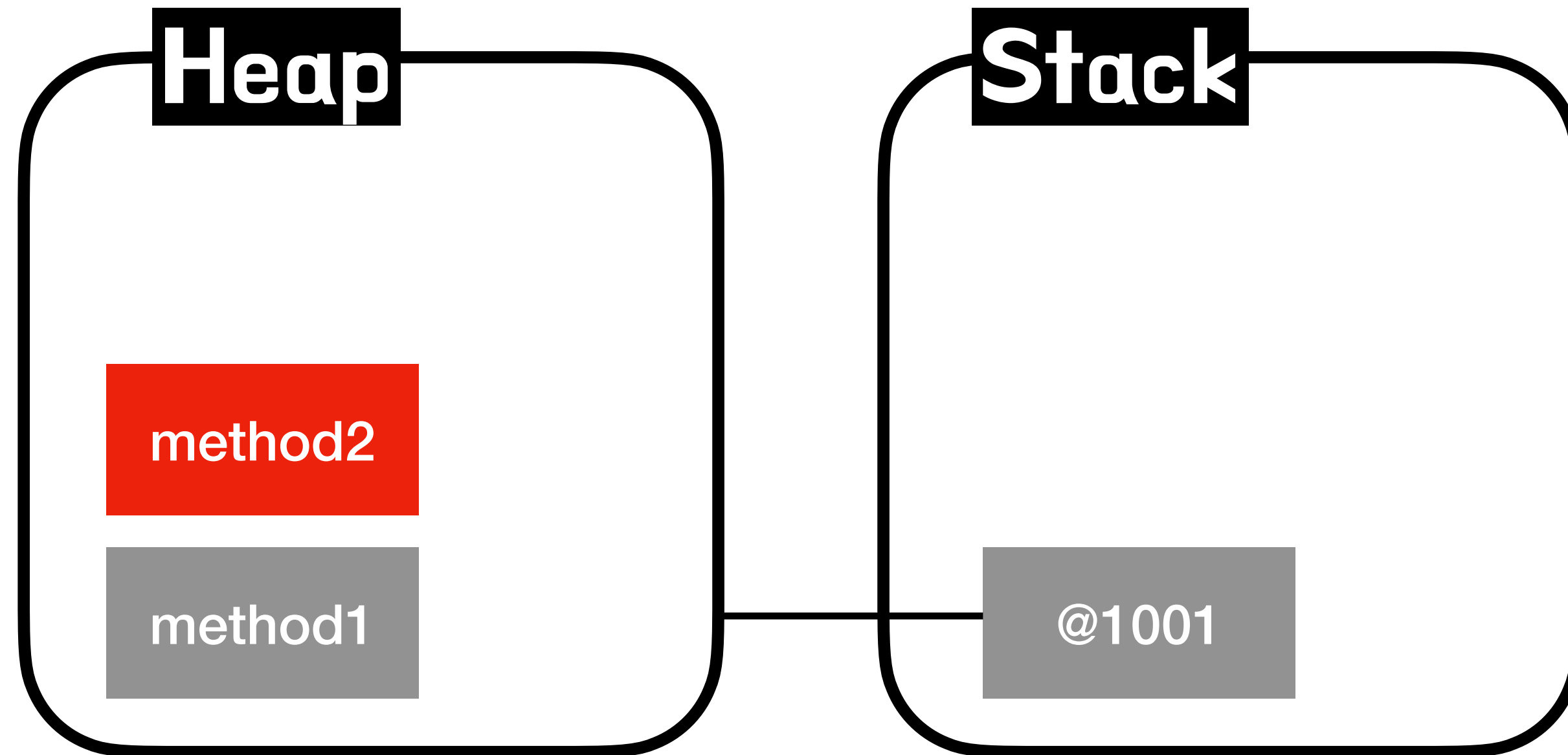
## 2) 가비지 컬렉션 청소 방식 : Mark And Sweep

Unreachable 한 객체 : "가비지 컬렉션 대상"



### 3. GC의 동작원리

#### 2) 가비지 컬렉션 청소 방식 : Mark And Sweep





# 3. GC의 동작원리

## 2) 가비지 컬렉션 청소 방식 : Mark And Sweep





# 3. GC의 동작원리

## 2) 가비지 컬렉션 청소 방식 : Mark And Sweep

Mark

Heap

method1

method2

method3

method4

method5

Sweep

Heap

method1

method2

method3

method4

method5



# 3. GC의 동작원리

## 2) 가비지 컬렉션 청소 방식 : Mark And Sweep

Mark

Heap

method1

method2

method3

method4

method5

Sweep

Heap

method1

method4

Compact

Heap

method1

method4





# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Weak Generational Hypothesis

- 대부분의 객체는 금방 접근 불가능한 상태(Unreachable)이 된다
- 오래된 객체에서 새로운 객체로의 참조는 아주 적게 존재한다



# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정





# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Young

- 새롭게 생성된 객체가 할당(Allocation)되는 영역
- 많은 객체가 Young 영역에 생성되었다가 사라짐
- Young 영역에 대한 GC: Minor GC

Heap

Young

Old

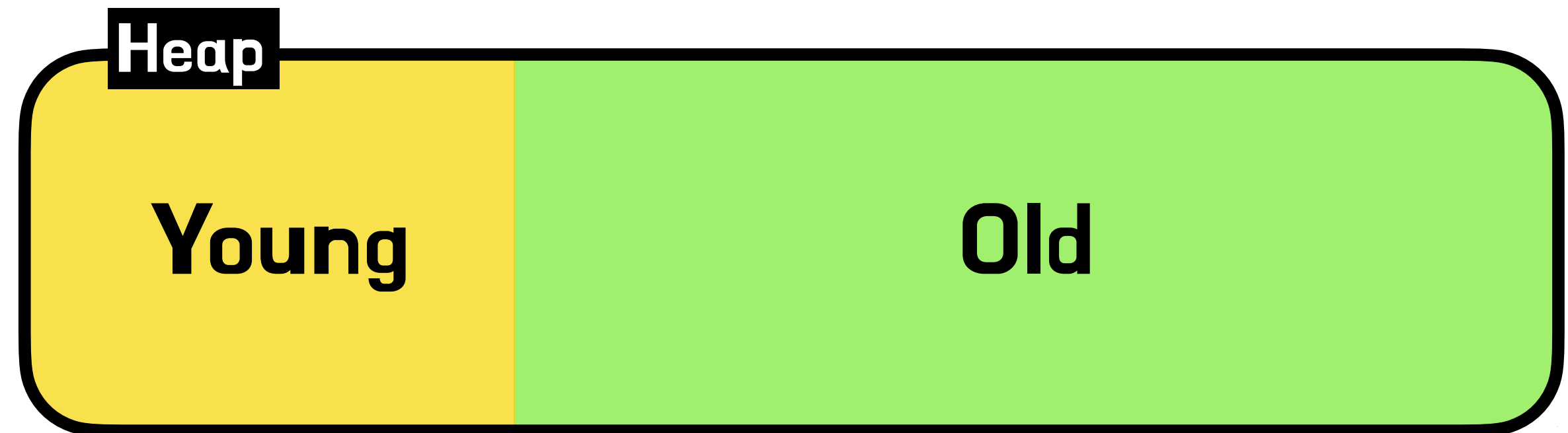


# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Old

- 살아남은 객체가 복사되는 영역
- Young 영역보다 크게 할당 -> 가비지 적게 발생
- Old 영역에 대한 GC: Major GC (Full GC)





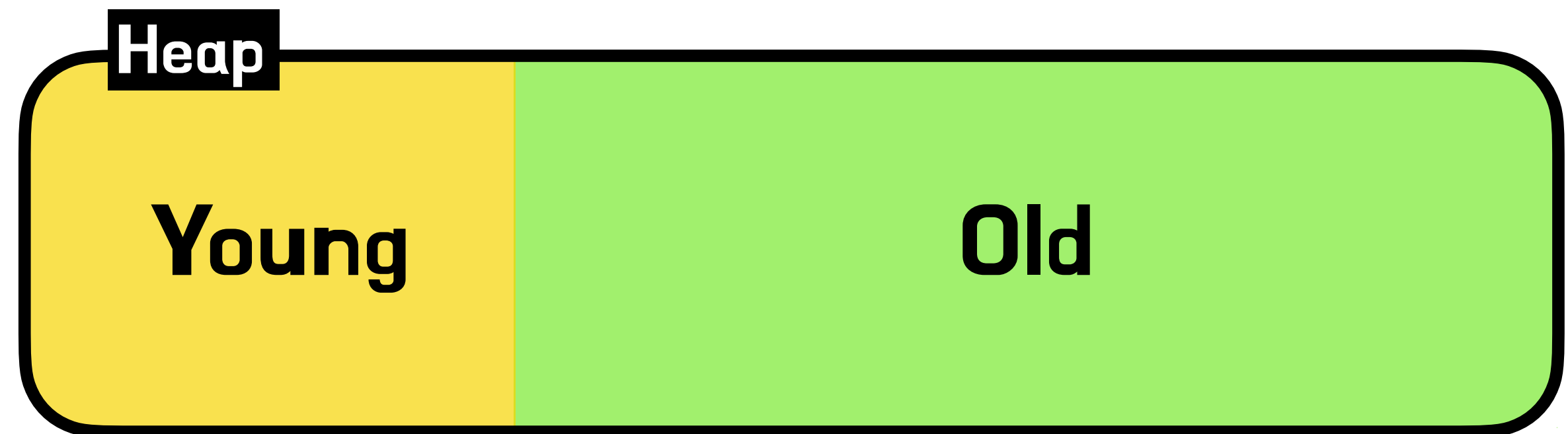
# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Old

- 살아남은 객체가 복사되는 영역
- Young 영역보다 크게 할당 -> 가비지 적게 발생
- Old 영역에 대한 GC: Major GC (Full GC)

1. 생존 주기
2. 횟수 빈도 차이





# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정



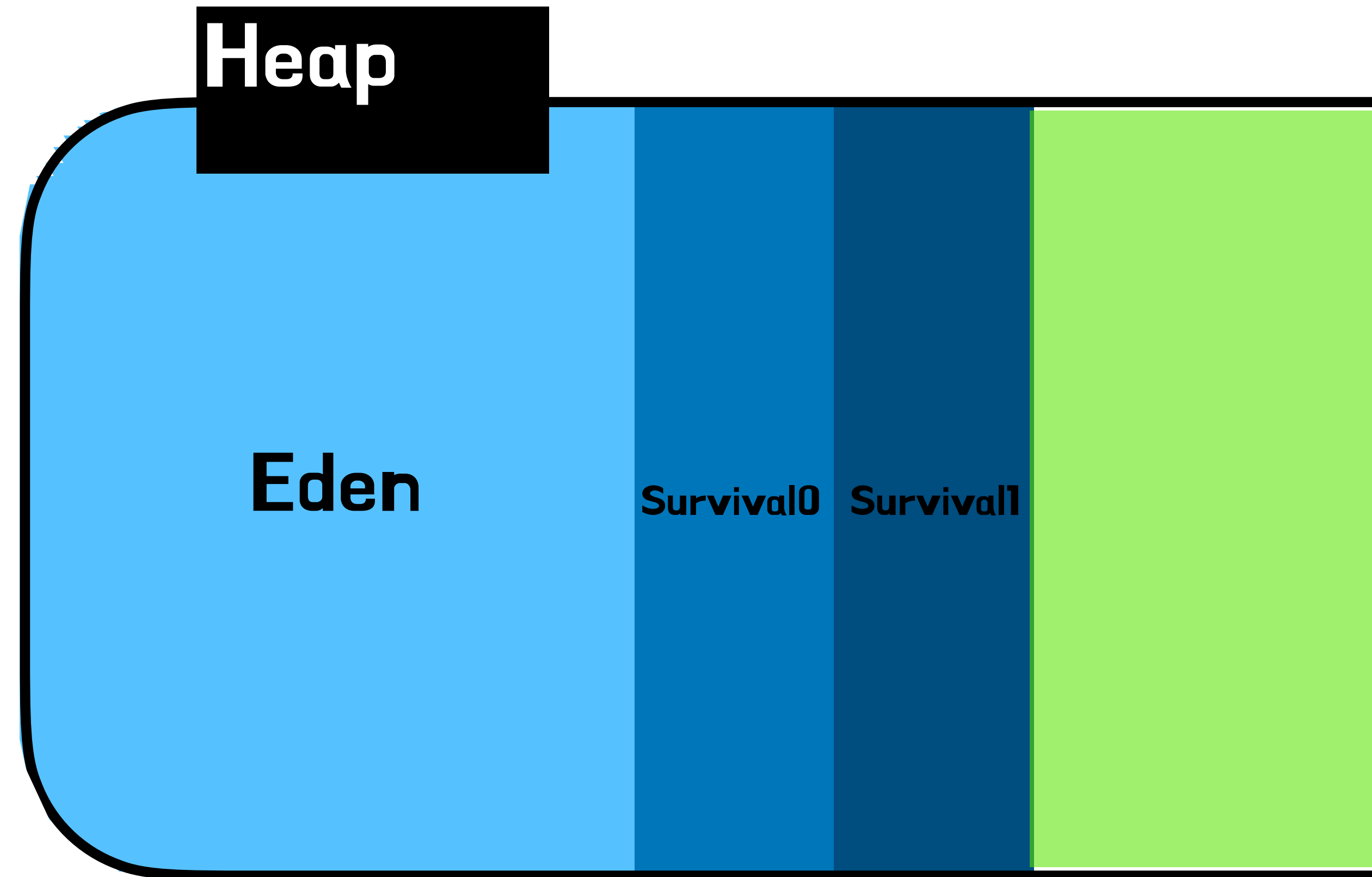


# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Eden

- new를 통해 새로 생성된 객체가 위치
- 정기적인 쓰레기 수집 후 살아남은 객체들은 Survivor 영역으로 보냄



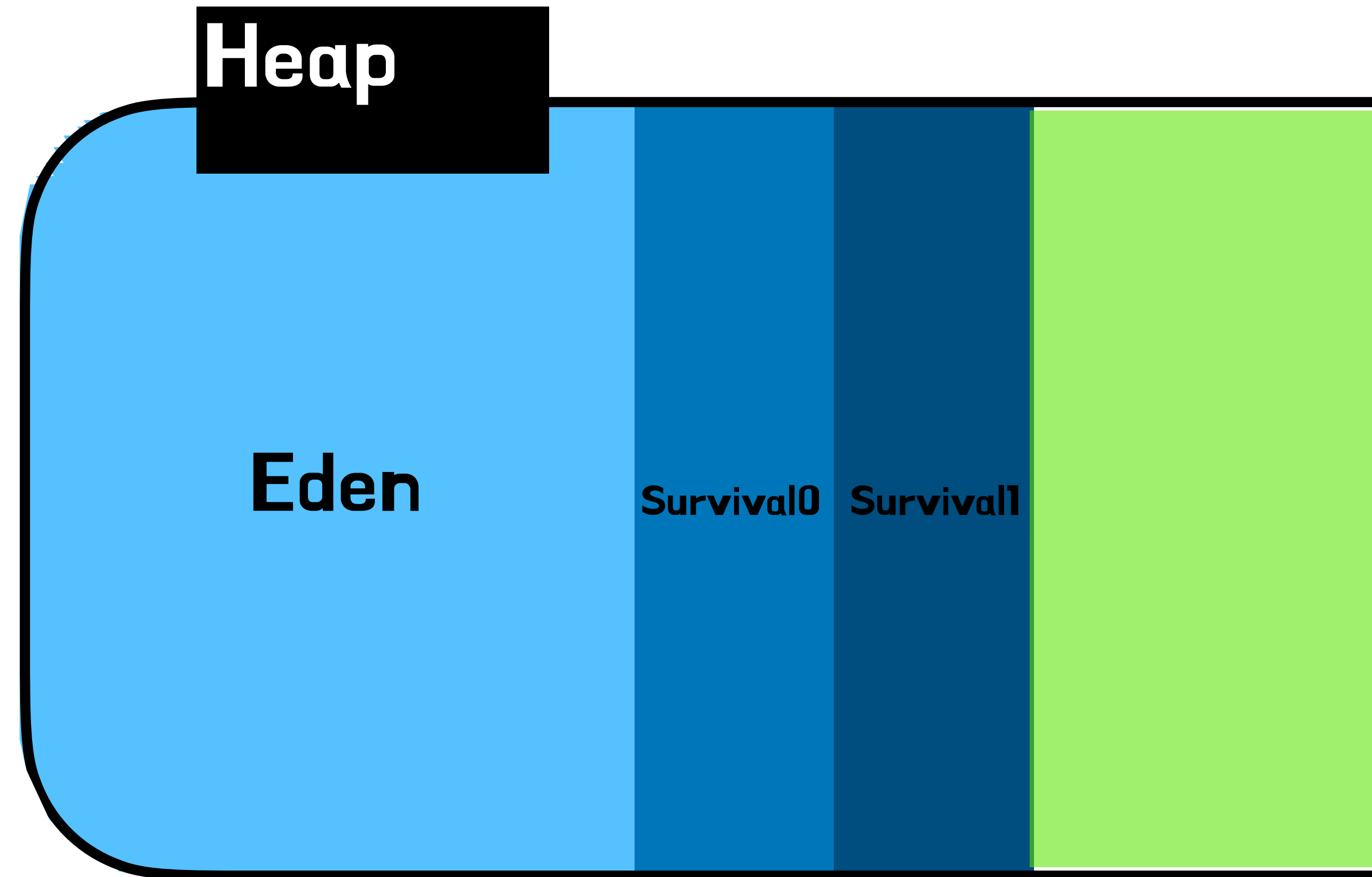


# 3. GC의 동작원리

## 3) 가비지 컬렉션 동작 과정

Survivor 0 / Survivor 1

- 최소 1번 GC 이상 살아남은 객체가 존재하는 영역
- Survivor 0 또는 Survivor 1 둘 중 하나에는 꼭 비어 있어야 함







## 4. GC 추이로 보는 배치 튜닝



## 4. GC 추이로 보는 배치 튜닝

### GC 로그 옵션

ENTRYPOINT

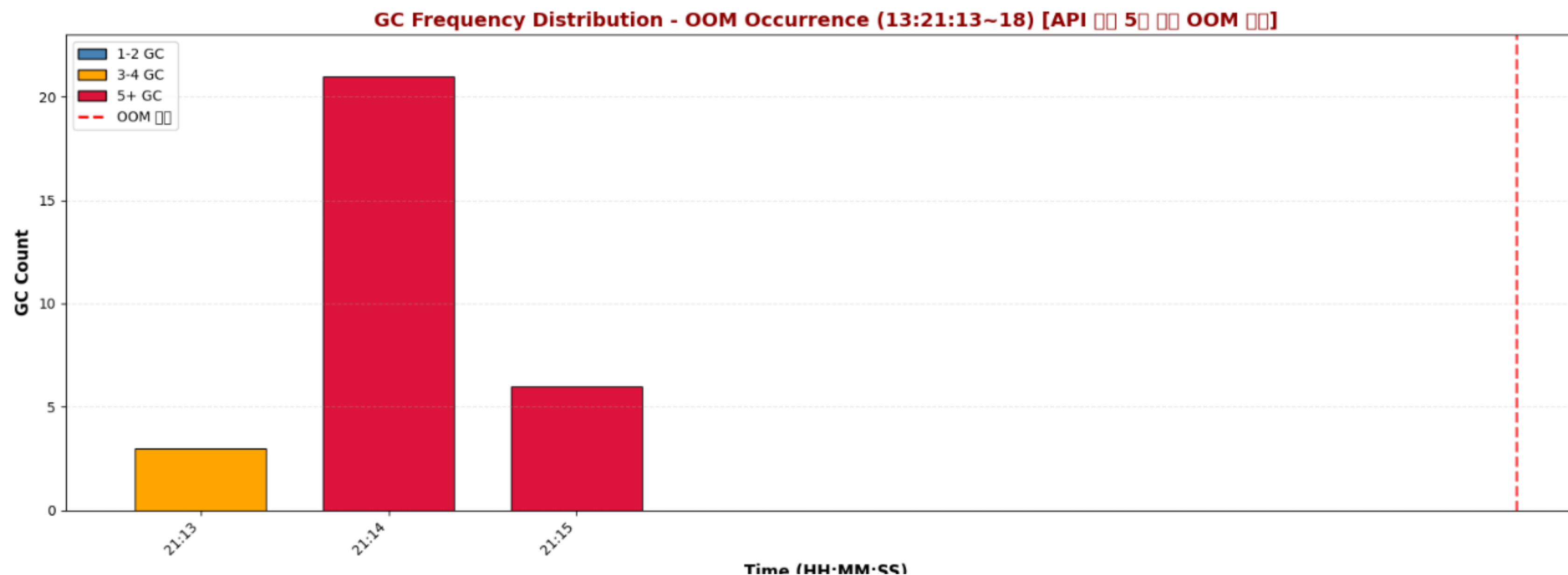
```
-Xlog:gc*:file=/logs/gc.log:time,uptime,level,tags
```



# 4. GC 추이로 보는 배치 튜닝

## 1) OOM 발생 상황

```
.Yzc3Mjc3MTktNjZi0S00NzFhLWJjNzUtYWl5YjZjN2QwNzZmIiwibWVtYi  
message=java.lang.OutOfMemoryError: Java heap space
```

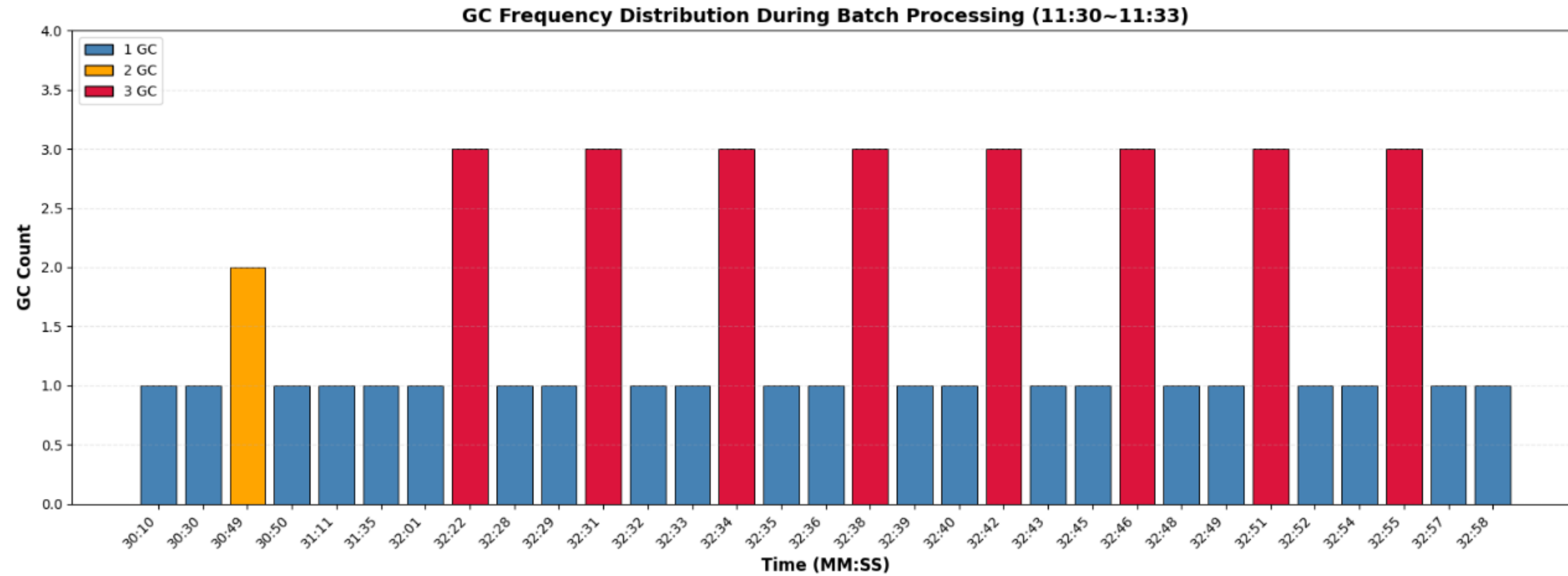


- 13:21:13초: Young GC 3개 (점상)
- 13:21:14초: GC 21개 폭주 (13.8초 이내)
  - Young GC → Mixed GC → Full GC 반복 (GC(43)~GC(111))
- 13:21:15초: Full GC 계속 (총 ~68개의 Full GC)
- 13:21:18초: ERROR 로그 = OOM 발생



# 4. GC 추이로 보는 배치 튜닝

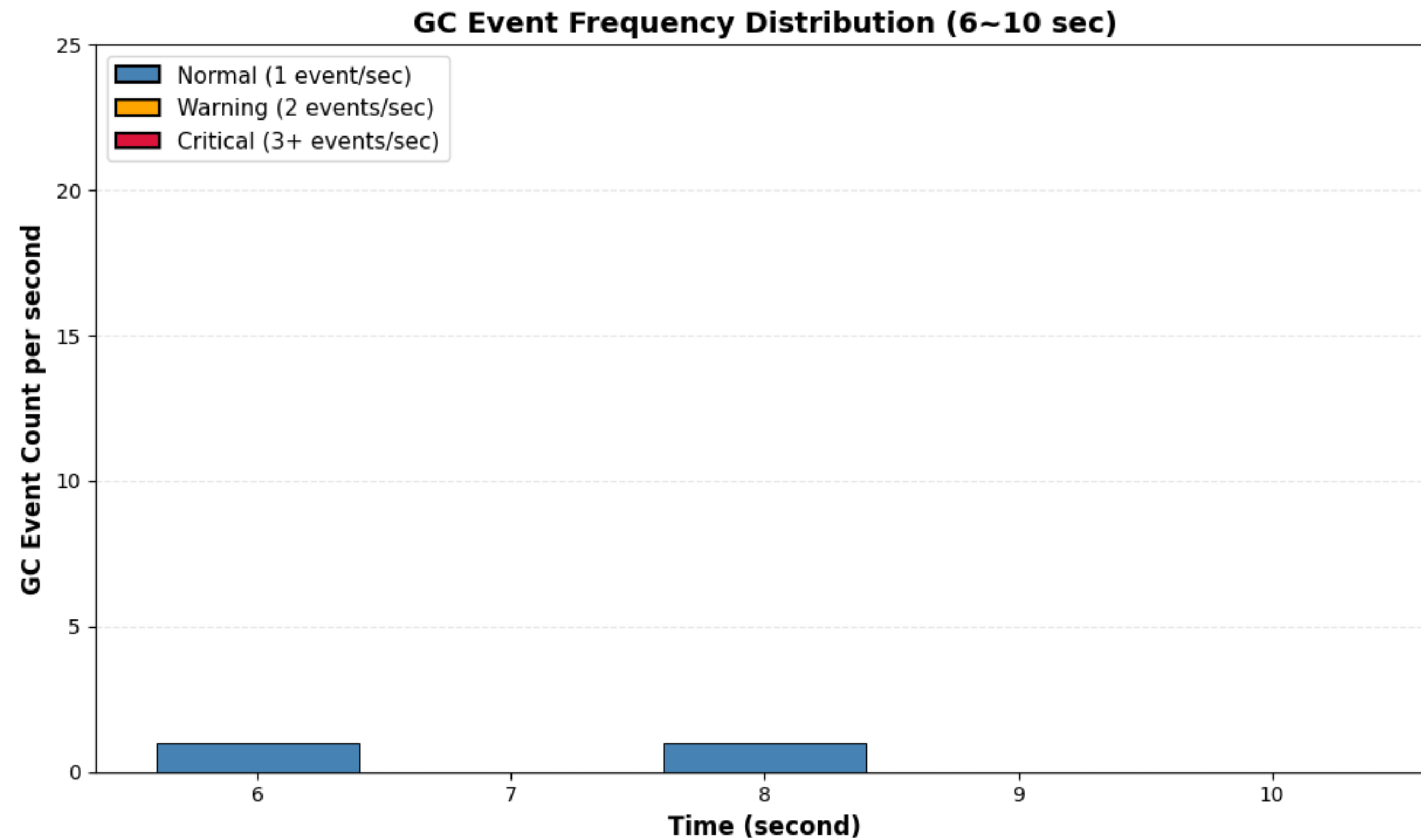
## 2) 배치 사이즈 1000으로 설정





# 4. GC 추이로 보는 배치 튜닝

## 3) 배치 사이즈 500으로 설정



**11**  
**^**



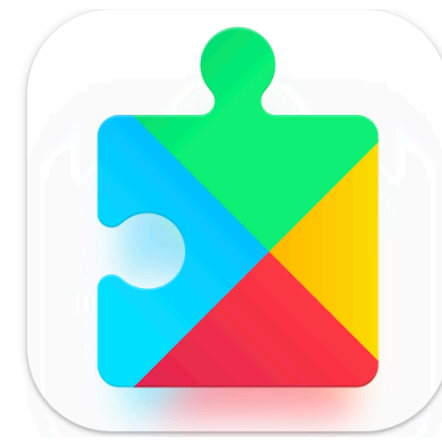
## 4. GC 추이로 보는 배치 튜닝

GC 딥다이브



## 2. 왜 FCM 인가?

### 푸시 알림 서비스의 생태계



**Google Play Services**