

## 멀티태스킹 (multi-tasking)

↳ 동시에 여러 작업을 실행

구현방법:

- ① 멀티 프로세싱 (multi-processing)  
↳ 프로세스를 여러개 병렬하여 실행
- ② 멀티 스레딩 (multi-threading)  
↳ 프로세스의 작업 중 일부를 다른 쓰레드에  
멀티태스킹이 병렬화된 쓰레드를  
여러 개 실행

쓰레드

## "멀티 태스킹"

concurrency  
↳ S/W로 병행

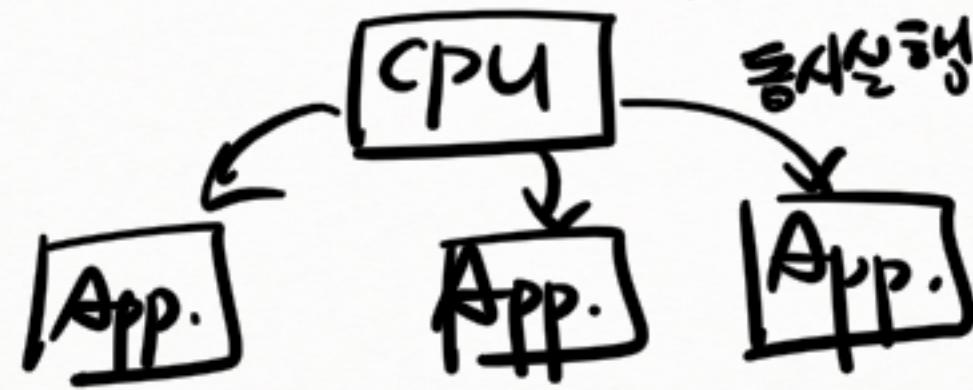
parallel  
↳ H/W 병렬



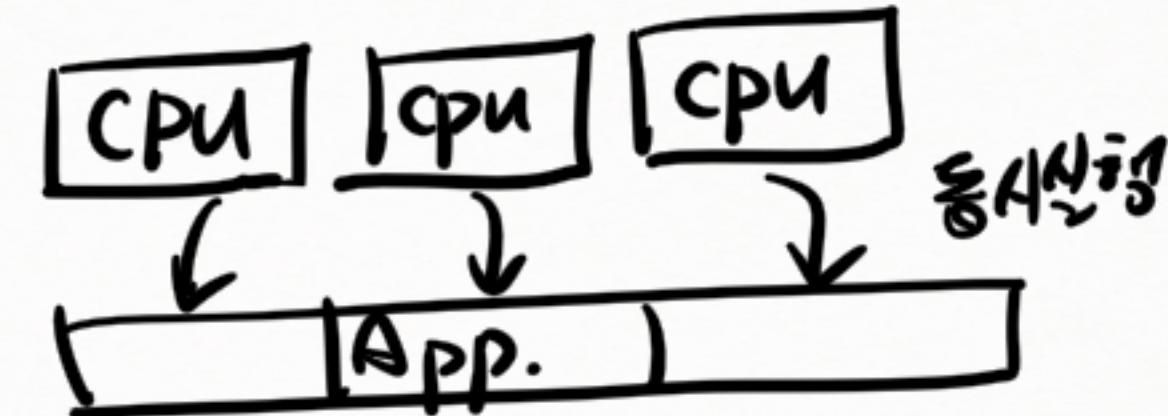
단수 처리

단수 처리

✓ Concurrency (병행 프로그래밍)

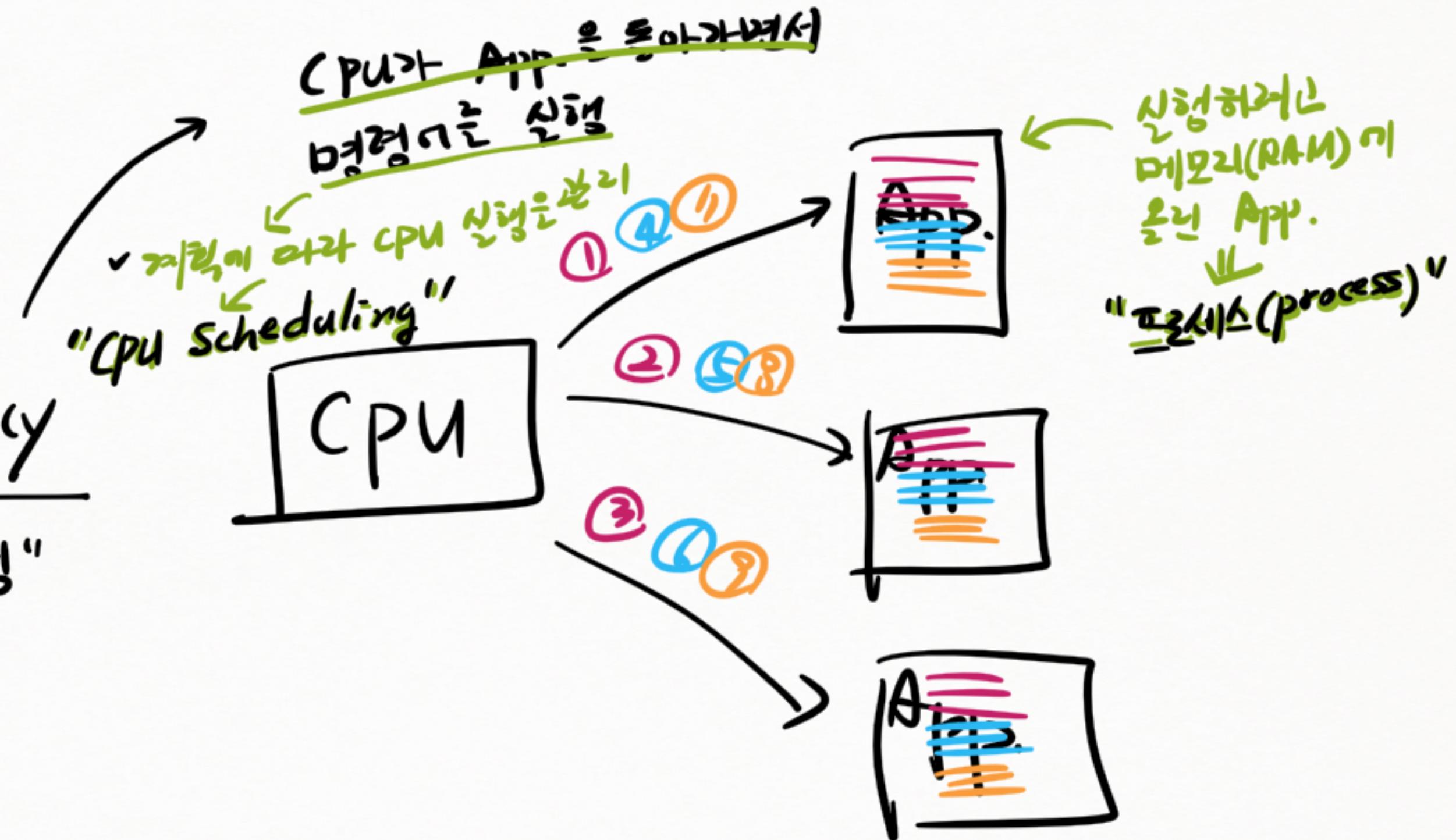


✓ Parallel (병렬 프로그래밍)



## Concurrency

"동시성"



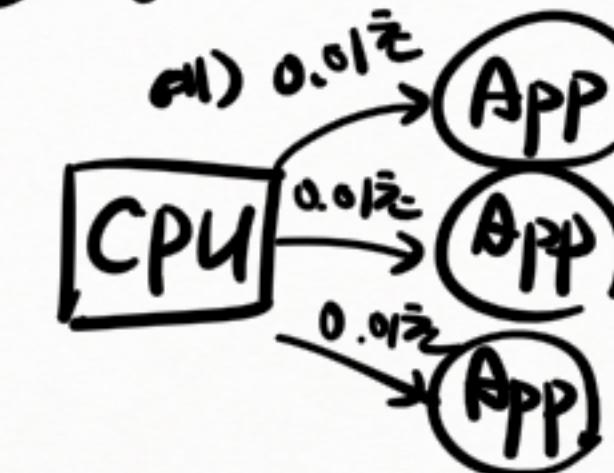
## \* CPU Scheduling

↳ 여러 개의 프로세스를 동시에 실행하기 위해  
CPU 사용을 관리하는 방법

### ① Round-Robin

↳ 각 프로세스에 고정한 실행시간(타임 슬라이스)으로

↳ Windows OS



### ② Priority + Aging

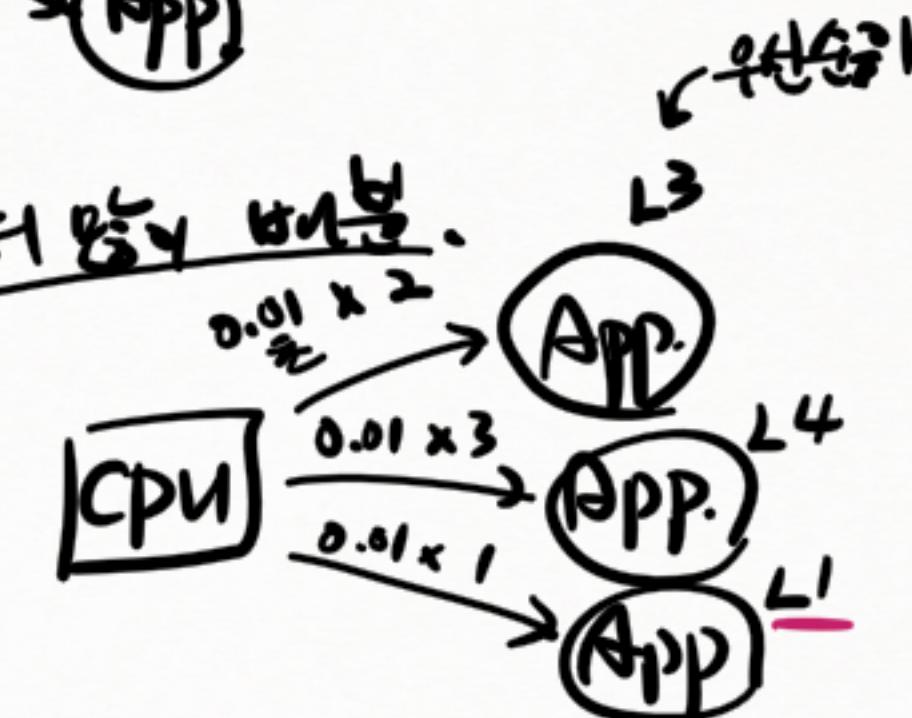
더 자주 실행

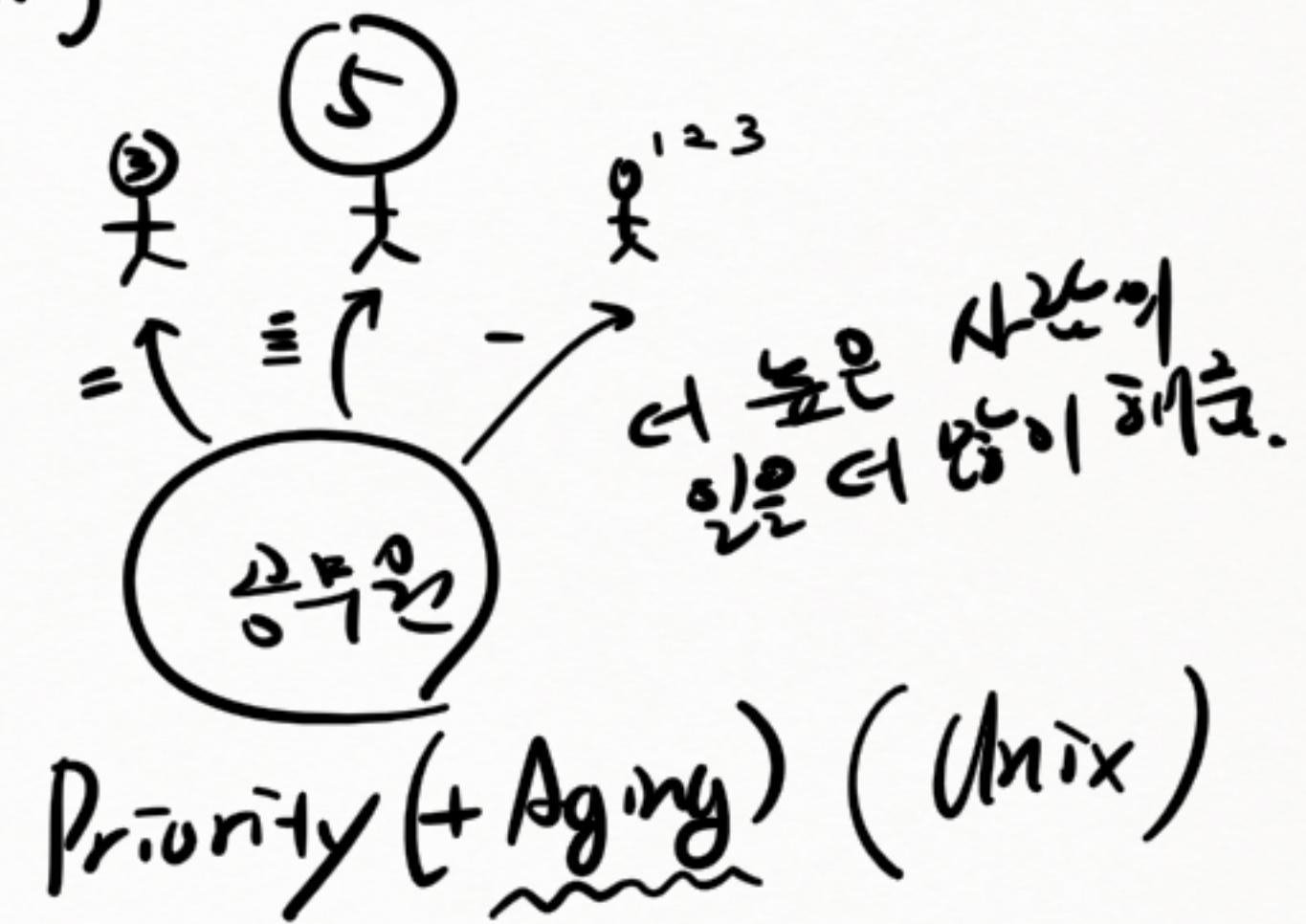
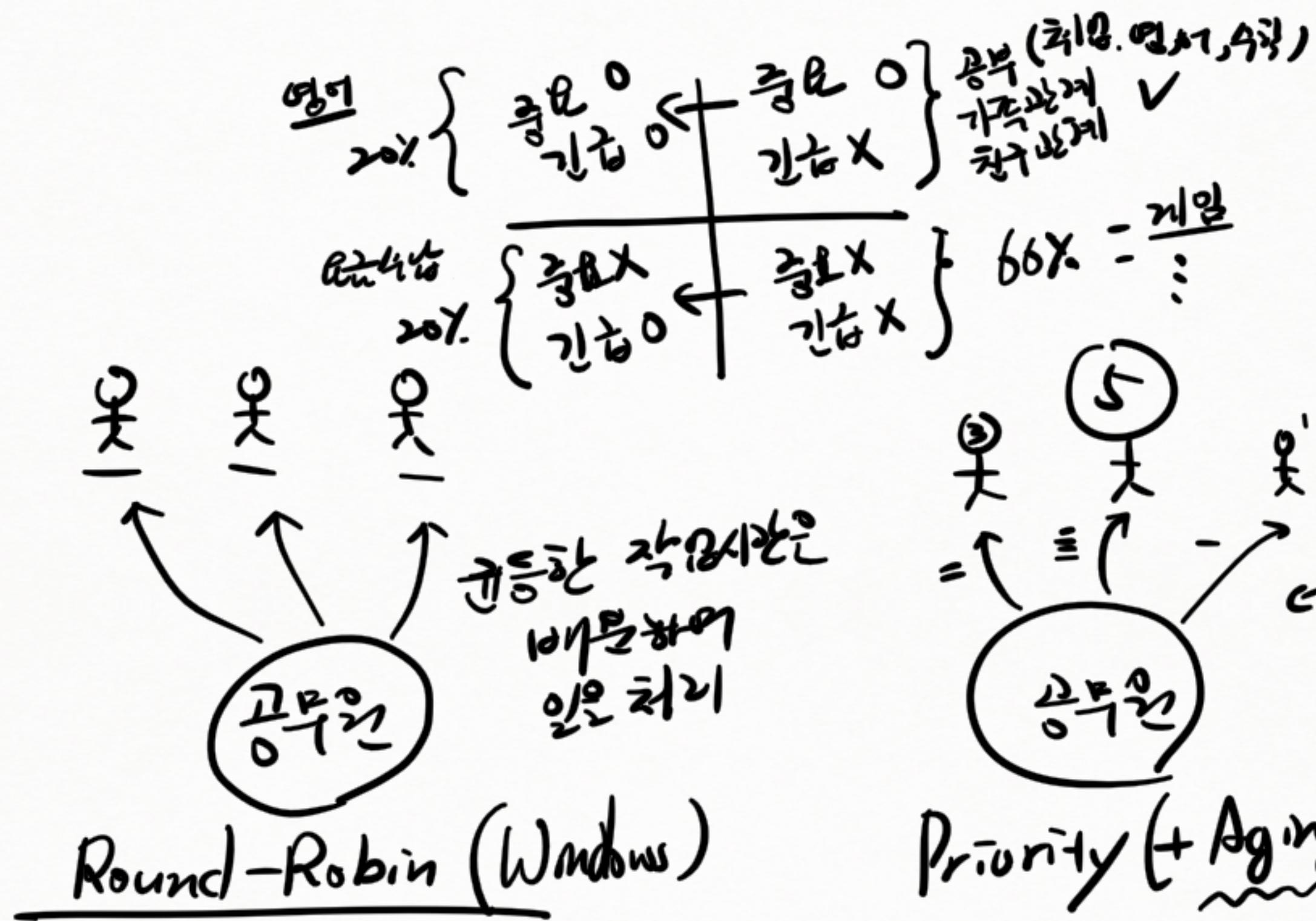
↳ 우선순위가 높은 프로세스는 CPU 사용시간을 더 많이 배분.

↳ 한 우선순위가 낮아서 실행이 연기될 때마다

우선순위 레벨(age)을 높여서 깛춰 실행하게 만든다

↳ Linux, macOS, Unix

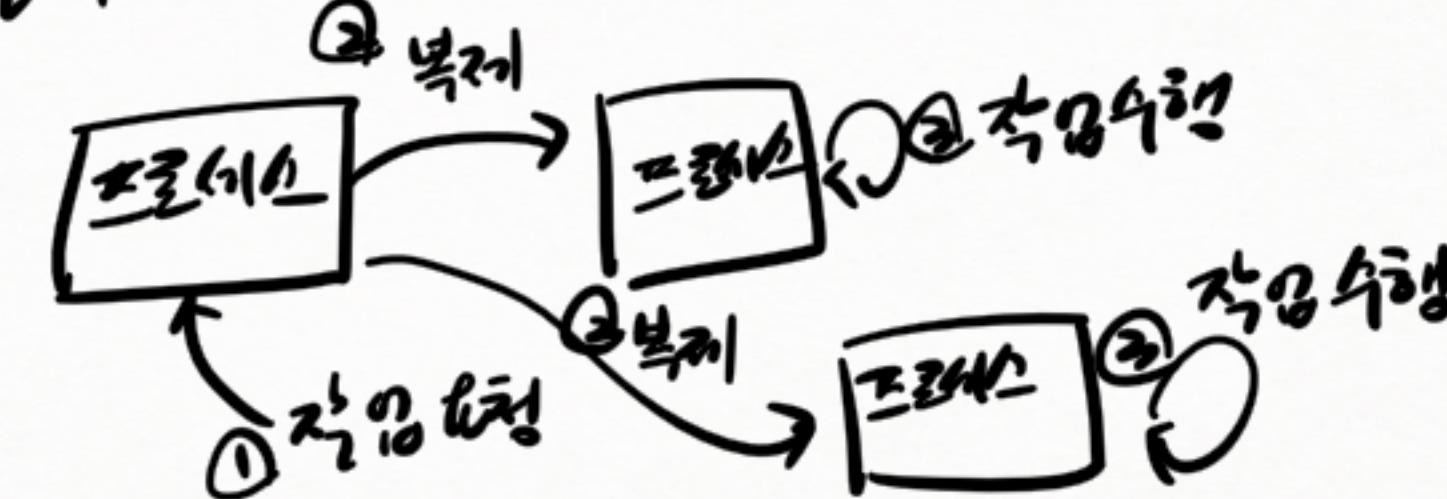




· “멀티태스킹” → 한정된 자원(CPU)을 사용하여  
“CPU 스케줄링 일관화”이 아니라  
(문제해결방법)  
여러 개의 프로세스를 동시에 실행하는 것!

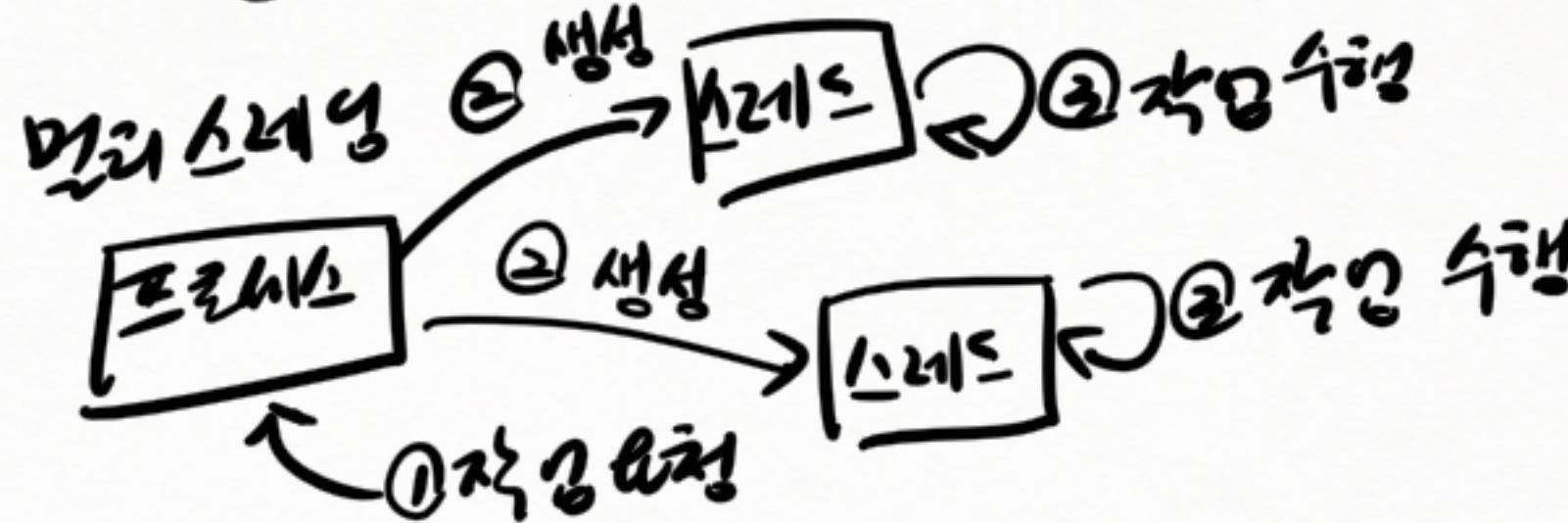
• 멀티태스킹  
1/10

✓ 멀티 프로세싱

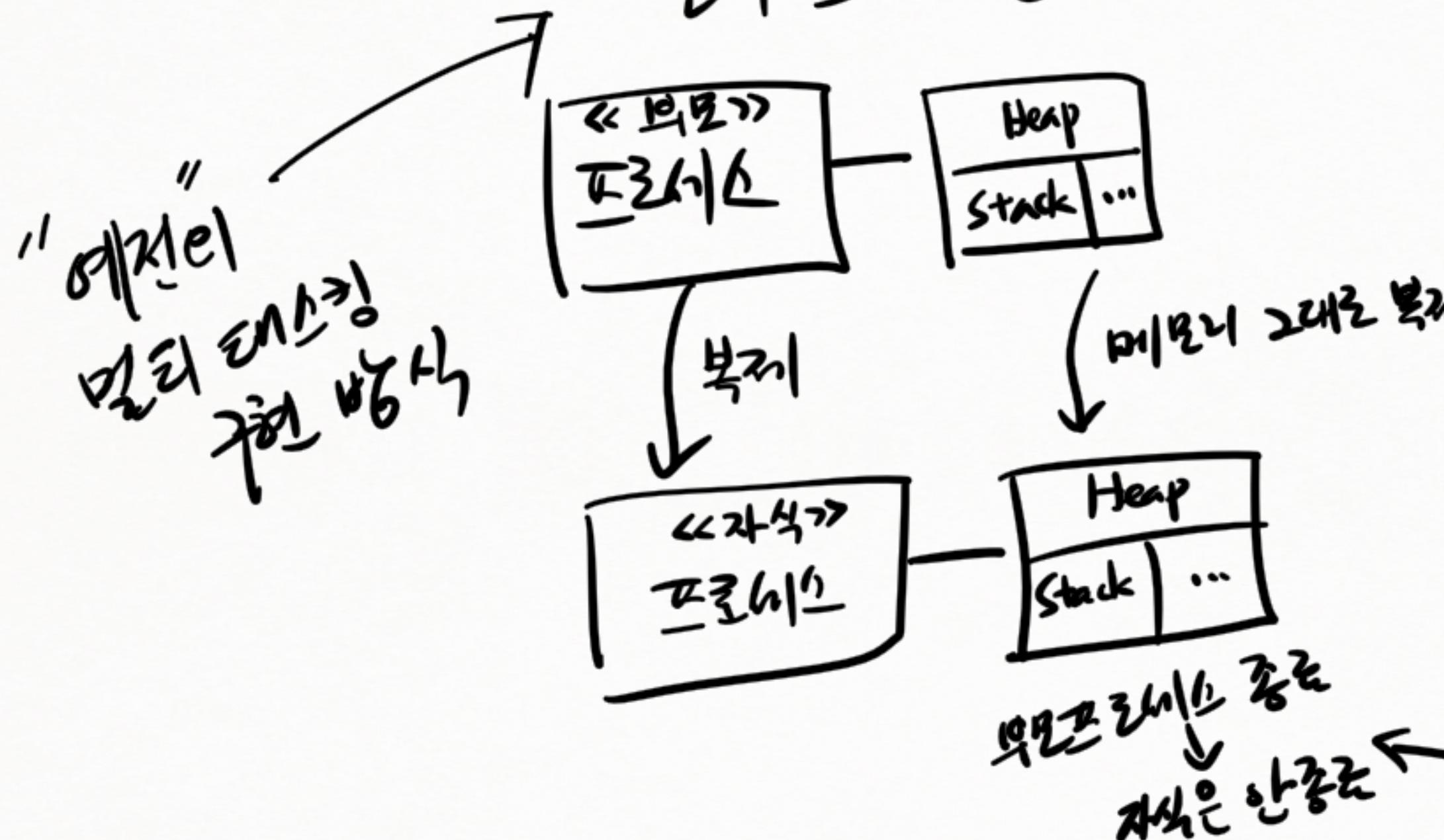


✓

멀티 스레딩



## 멀티 프로세싱

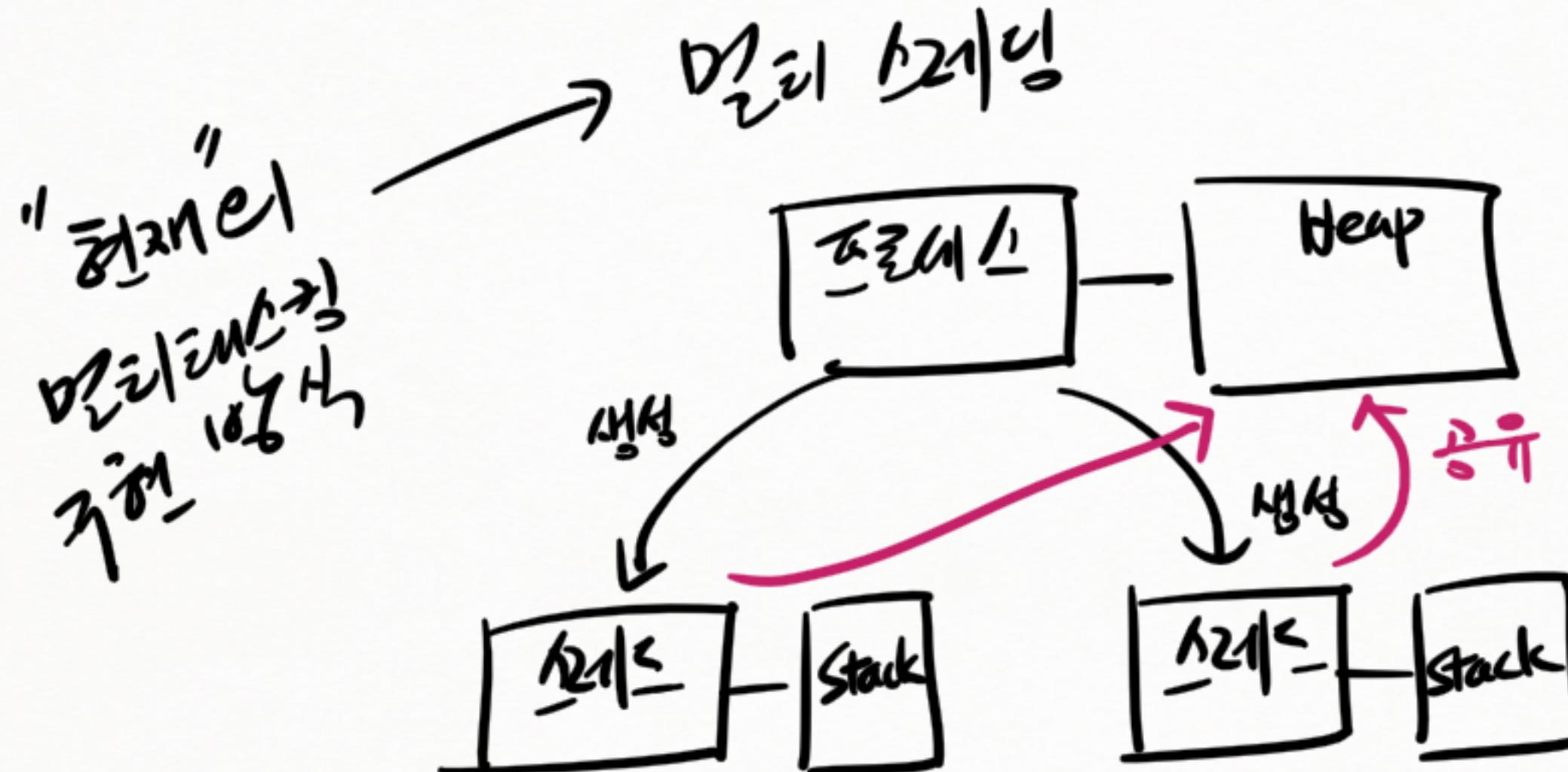


① 단순히 복제 (fork)

구현이 쉽다.  
(근대)

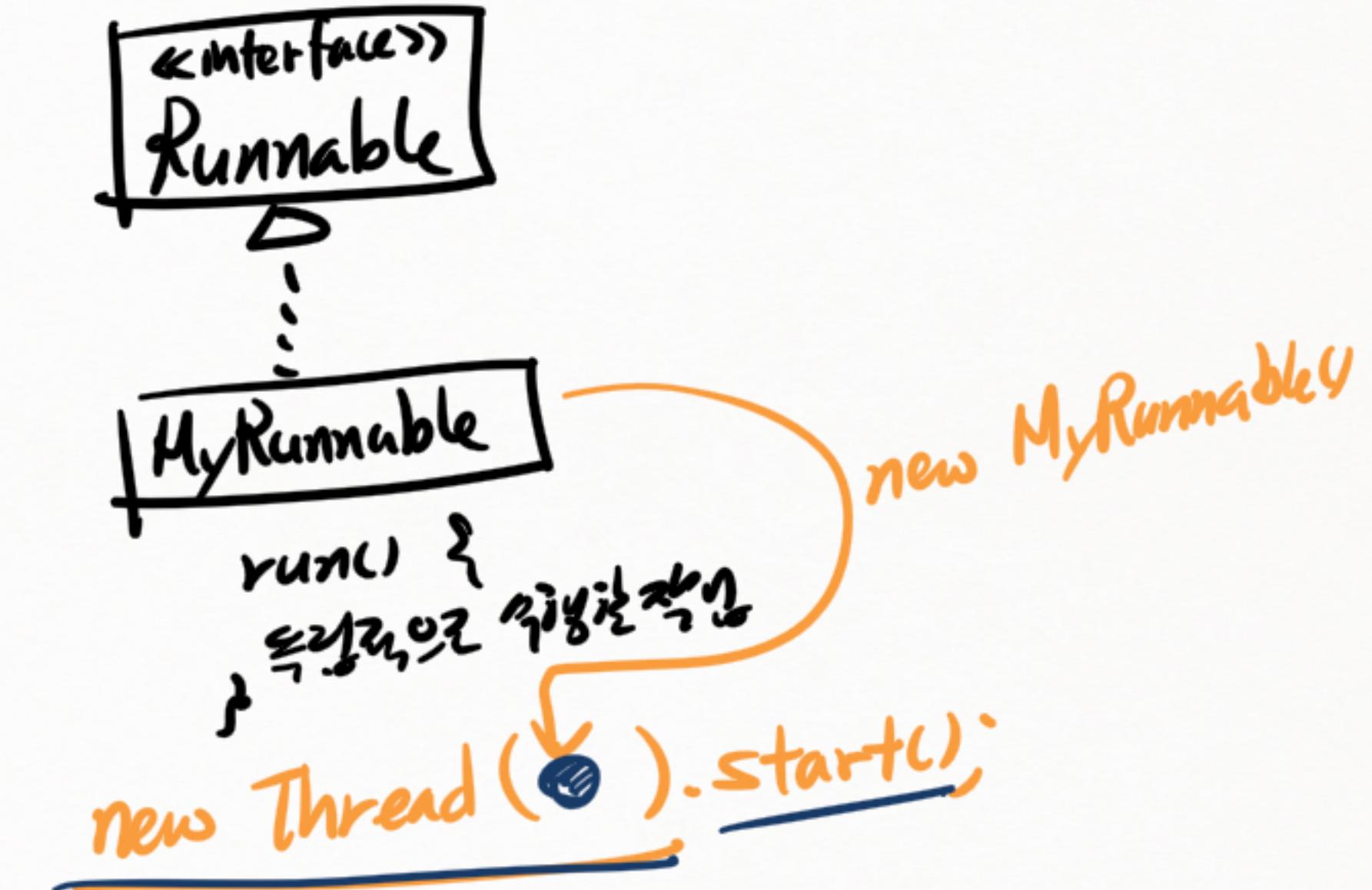
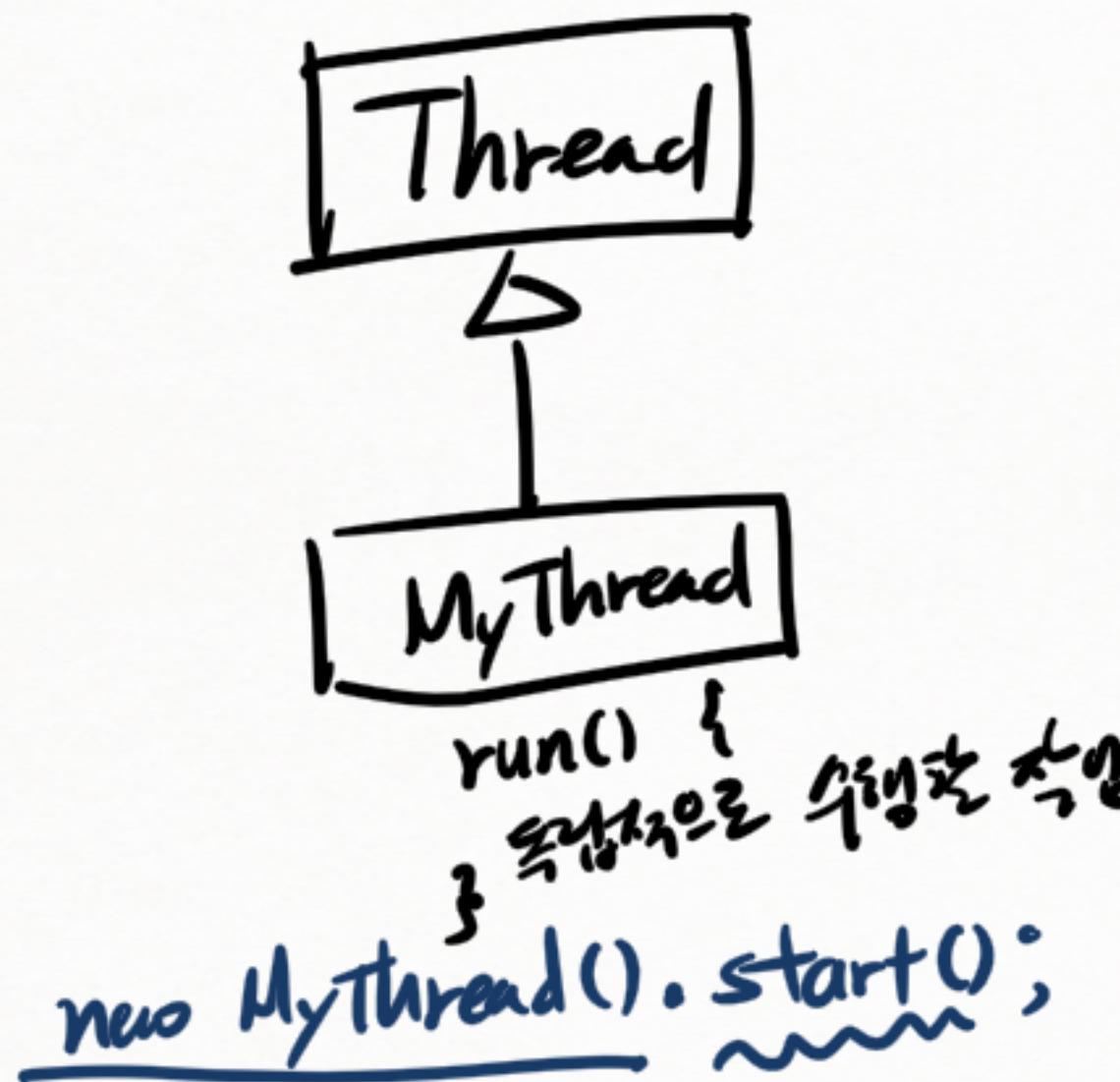
② 메모리를 그대로 복제

메모리 누수 가 산화  
자식 프로세스는 부모의  
종속되지 않는다.

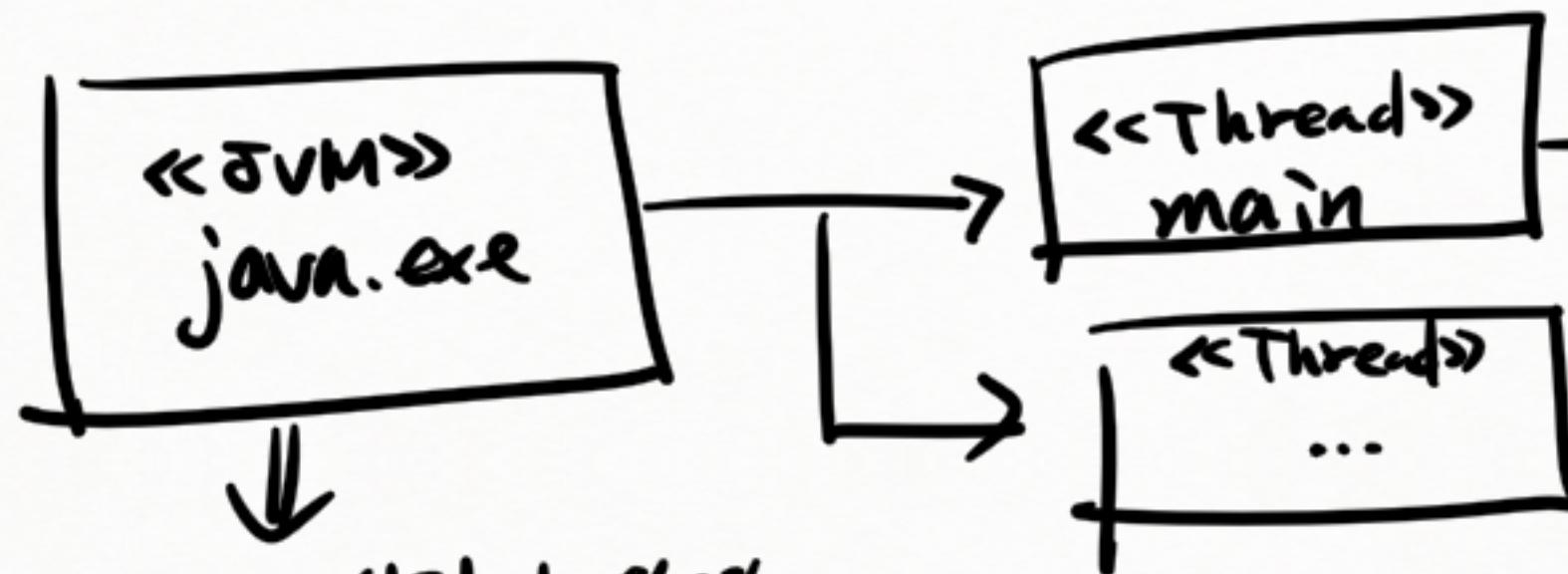


- ① 쓰레드 생성
  - ↓
  - + 현재 프로세스 복잡
- ② 쓰레드의 Heap을公用
  - ↓
  - 메모리 절약
- ③ 쓰레드는 프로세스의 종속
  - ↓
  - 프로세스 종속 → 쓰레드 종속

\* Java 스레드 구현



## \* JVM 과 스레드



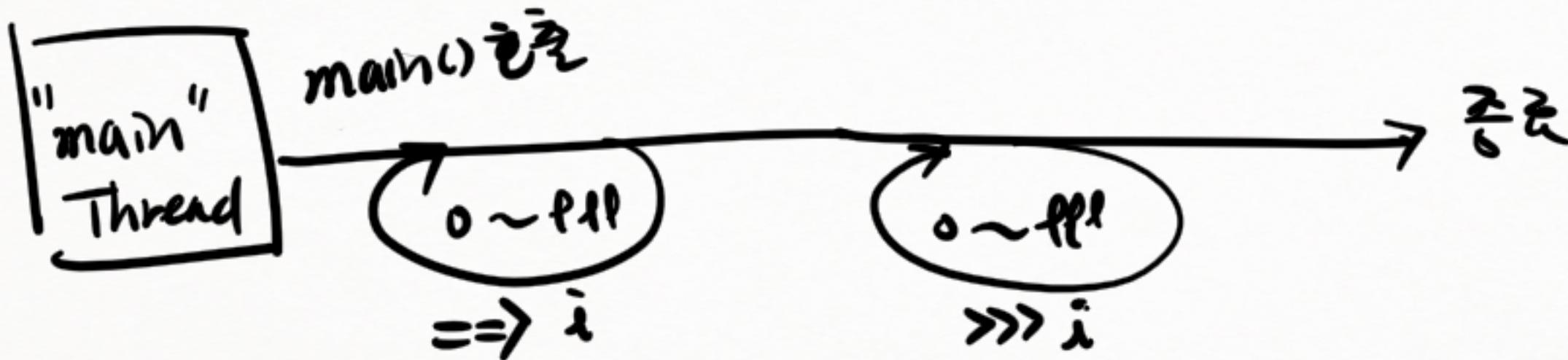
Java App. 실행이 필요한  
여러 스레드를 갖는다.

JVM이 직접 호출하지  
않는다.

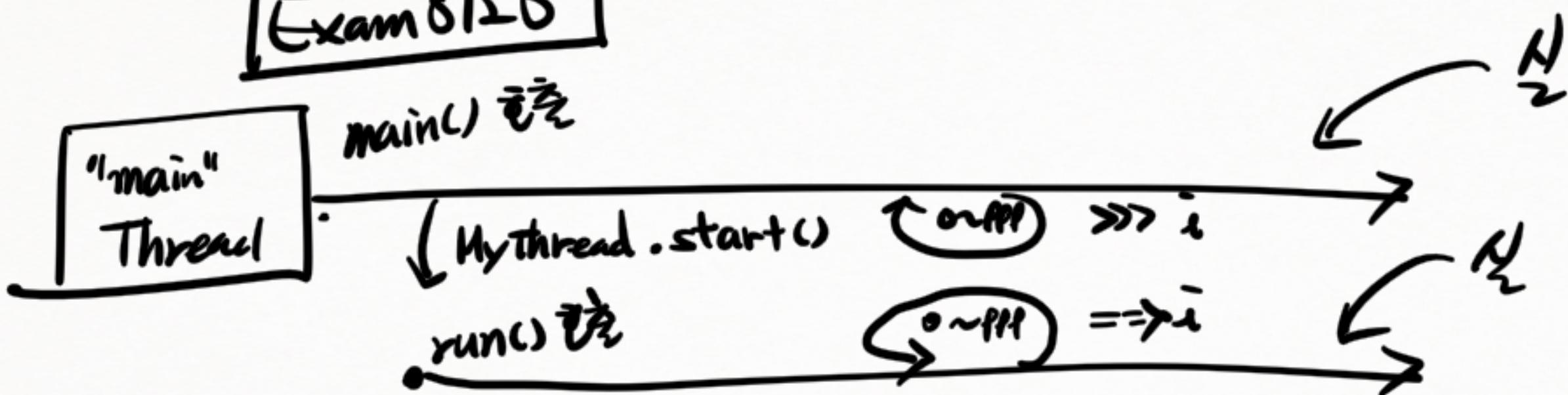
- 인스턴스의 러너던스 캐리즈를  
관리하는 스레드
- Garbage Collector 역할 수행  
스레드
- 외부 이벤트를 관리하는 스레드

:

### Exam 0110



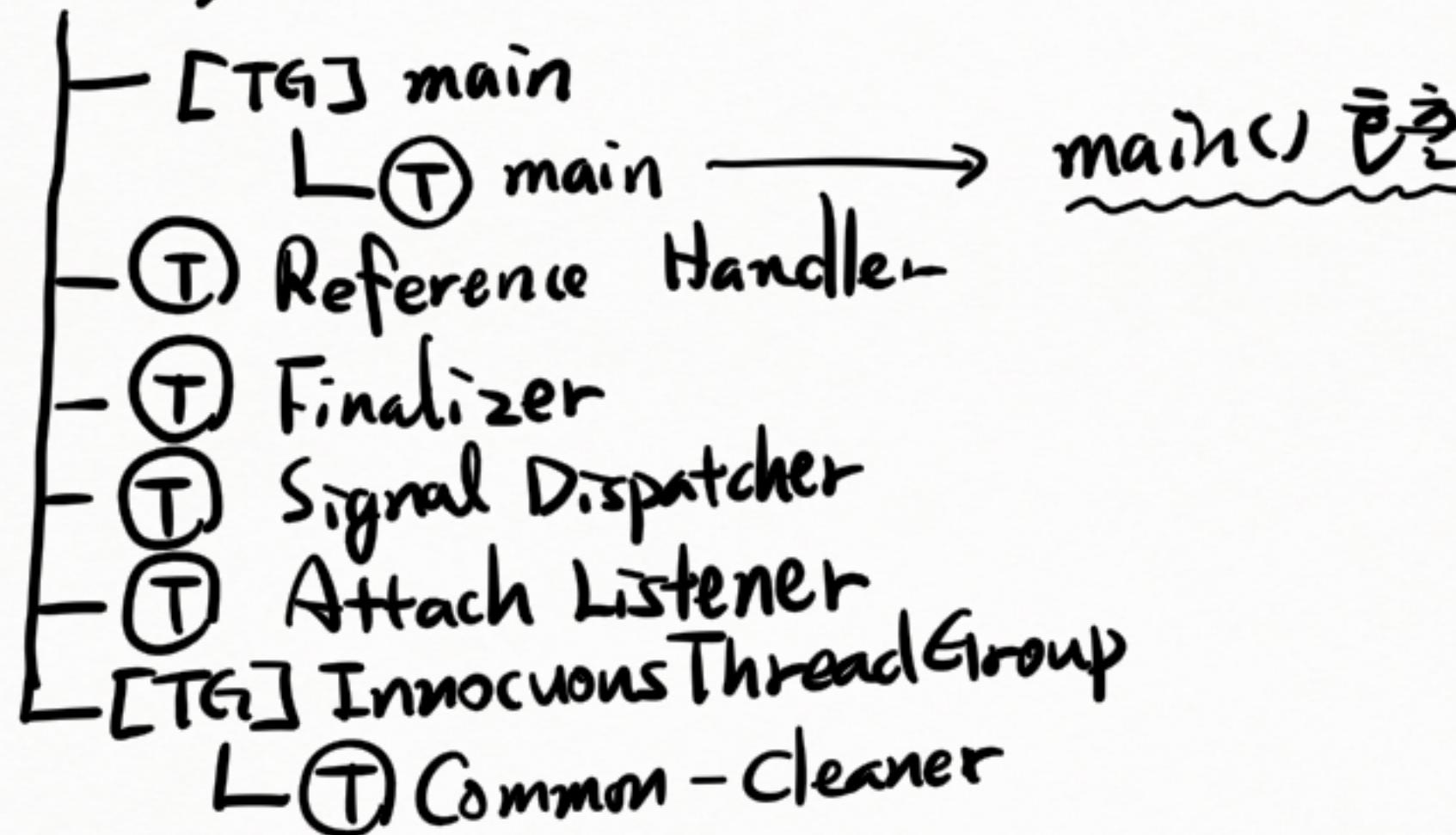
### Exam 0120



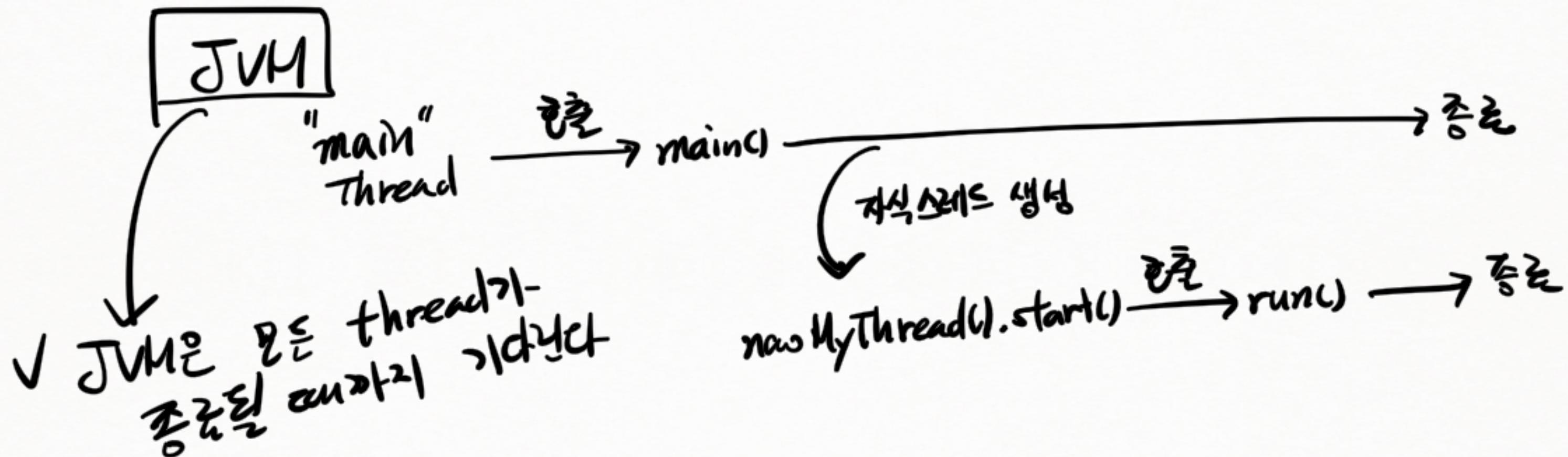
# JVM Thread > 총

JVM 프로세스

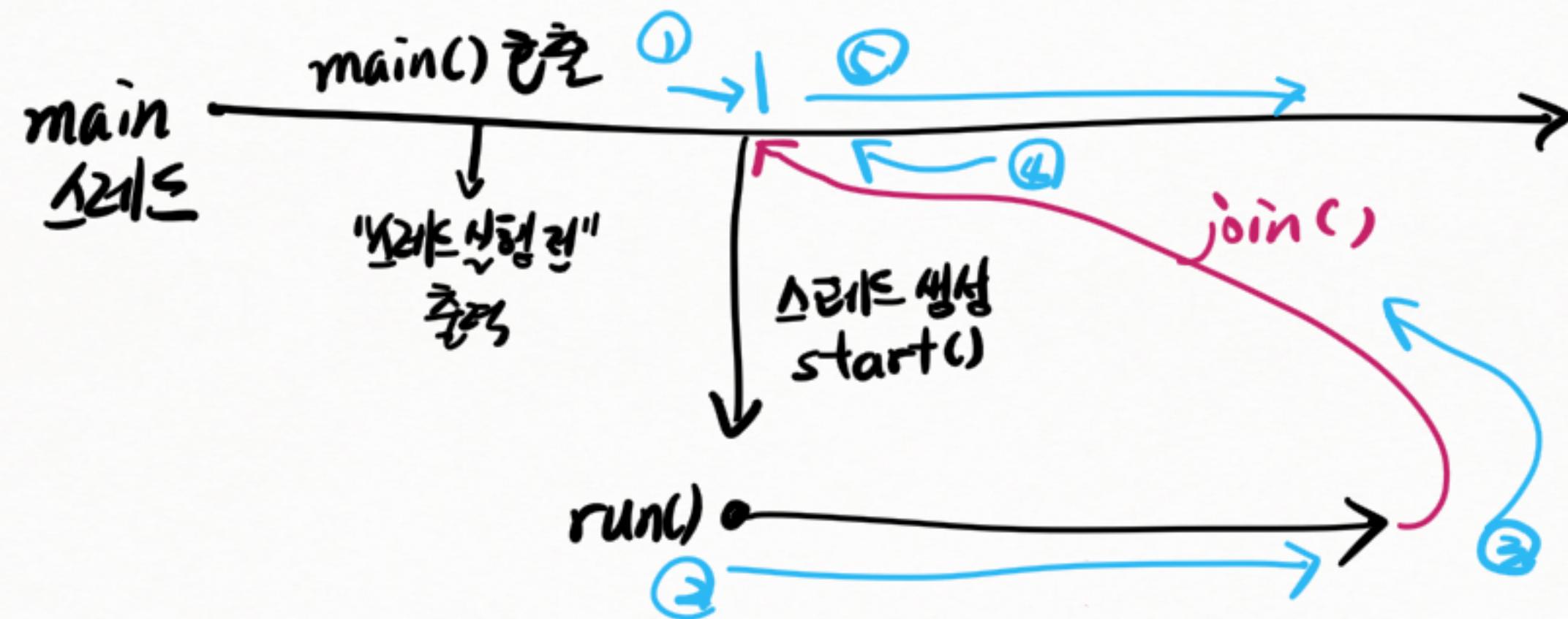
[TG] system



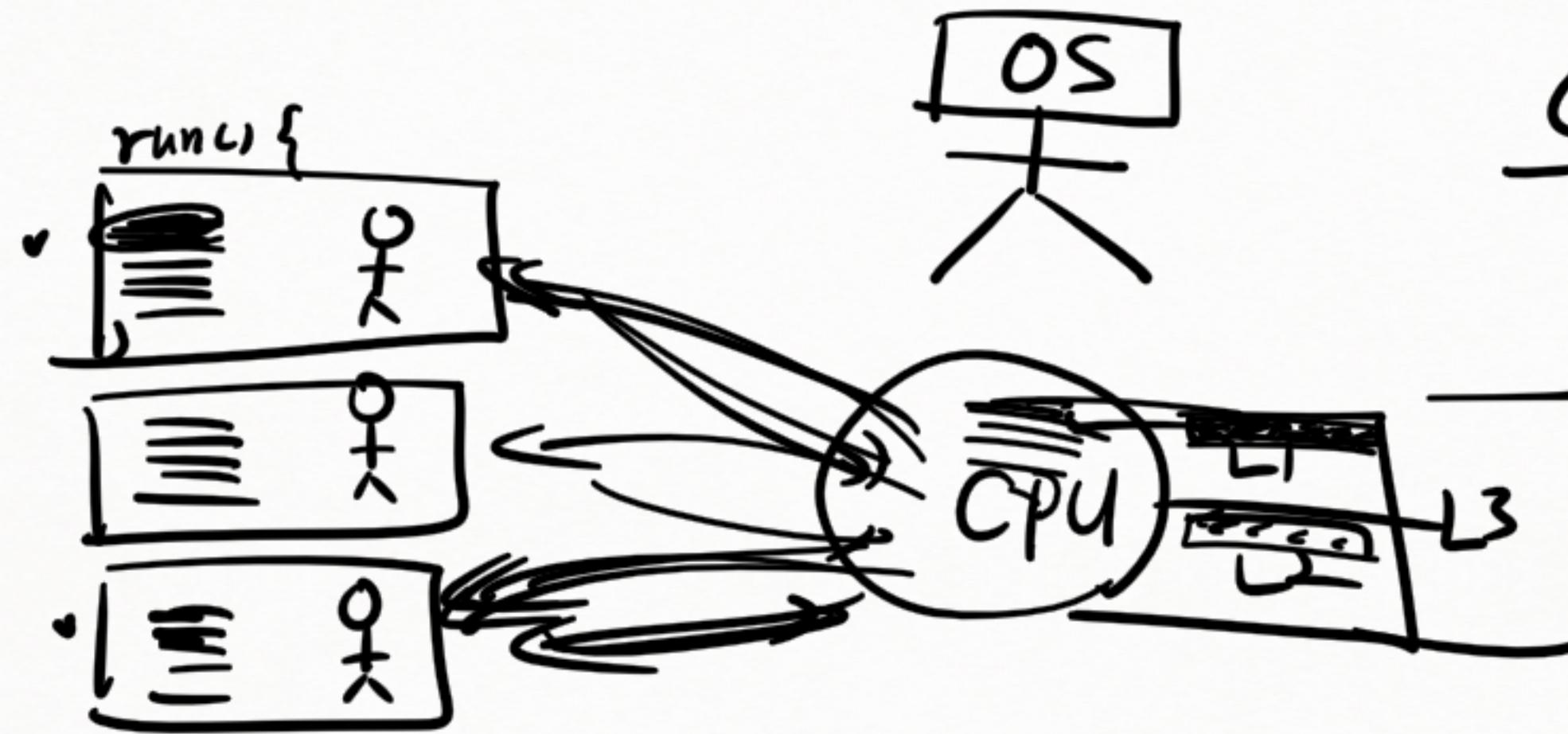
## 프로세스와 스레드의 실행 종료



## Thread::join()

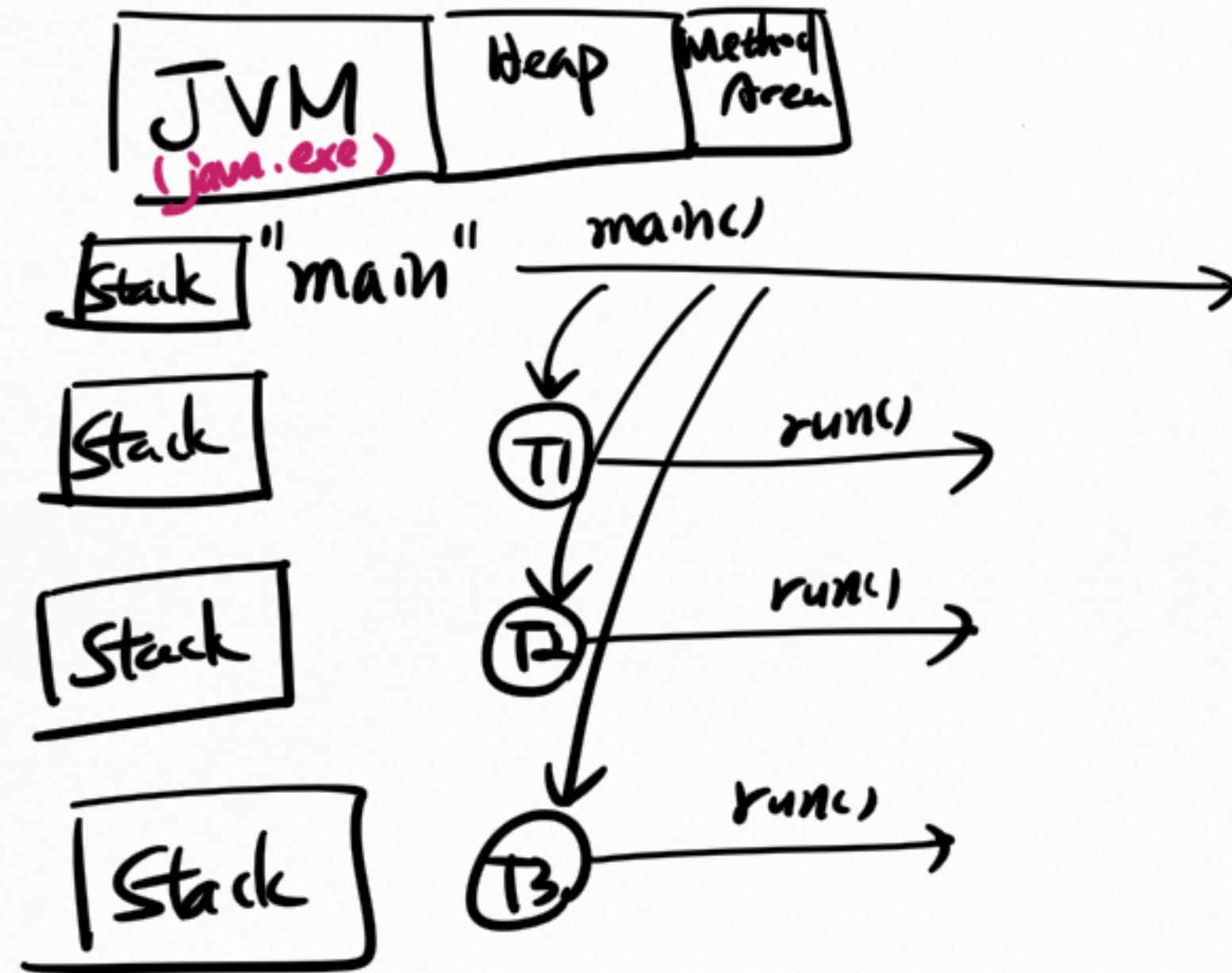


의인화



비선형 OS ← DOS + Windows 3.x

선형 OS ← Windows NT -

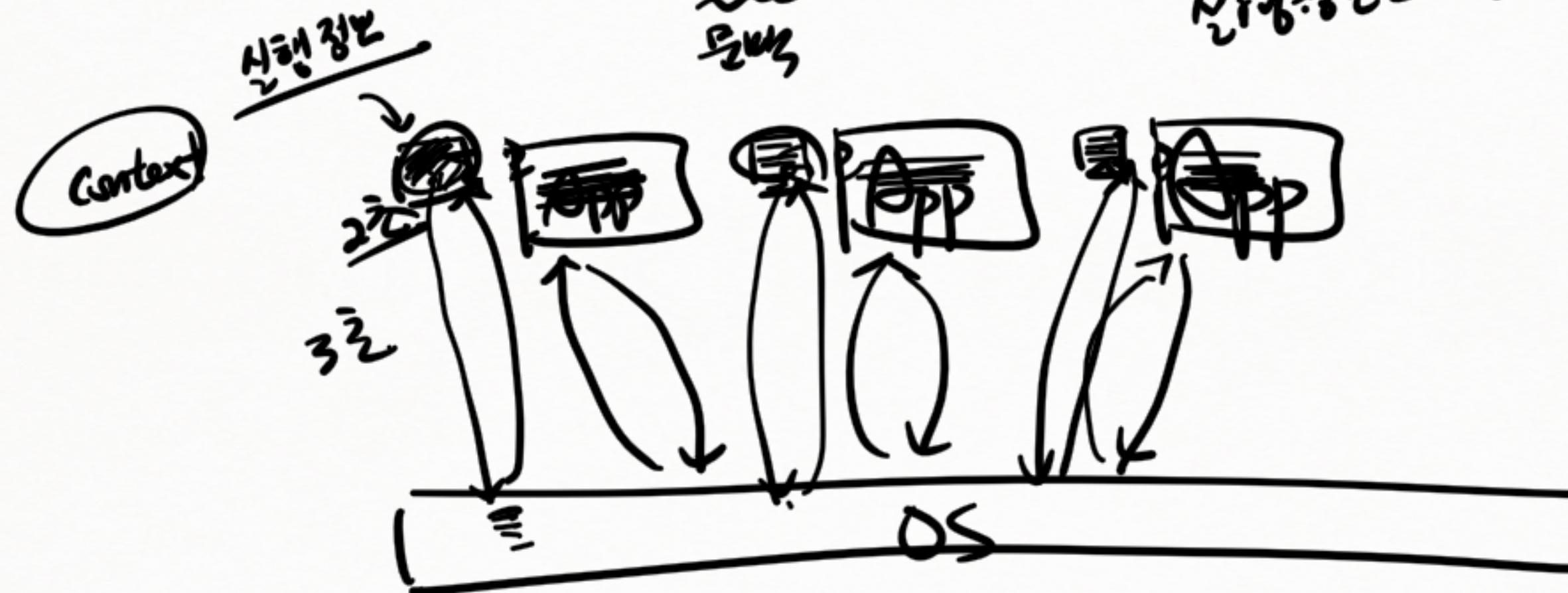


## Context Switching

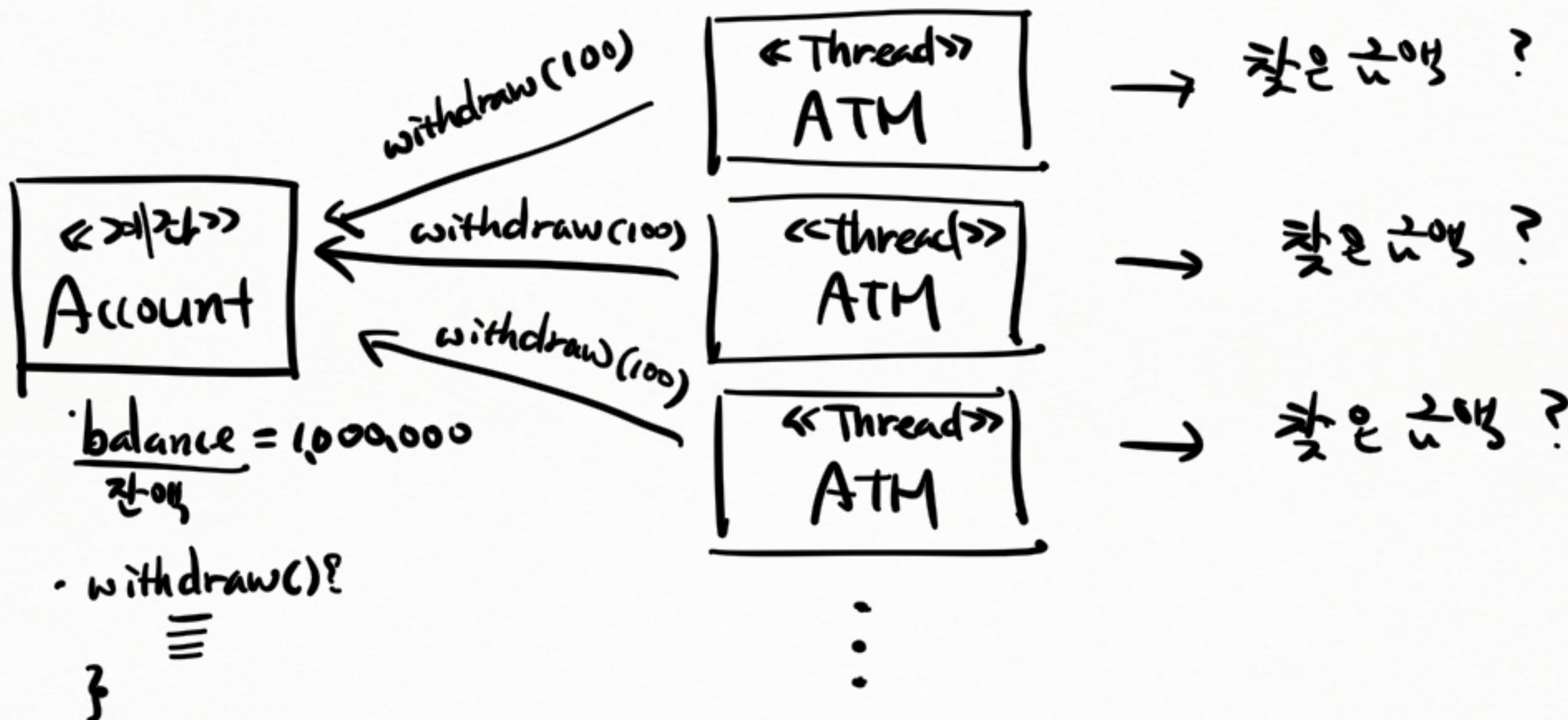
환경  
상호작용  
도구

## Switching

→ 프로세스나 스레드를 실행하는 시스템  
환경변경을 수행하는 것.

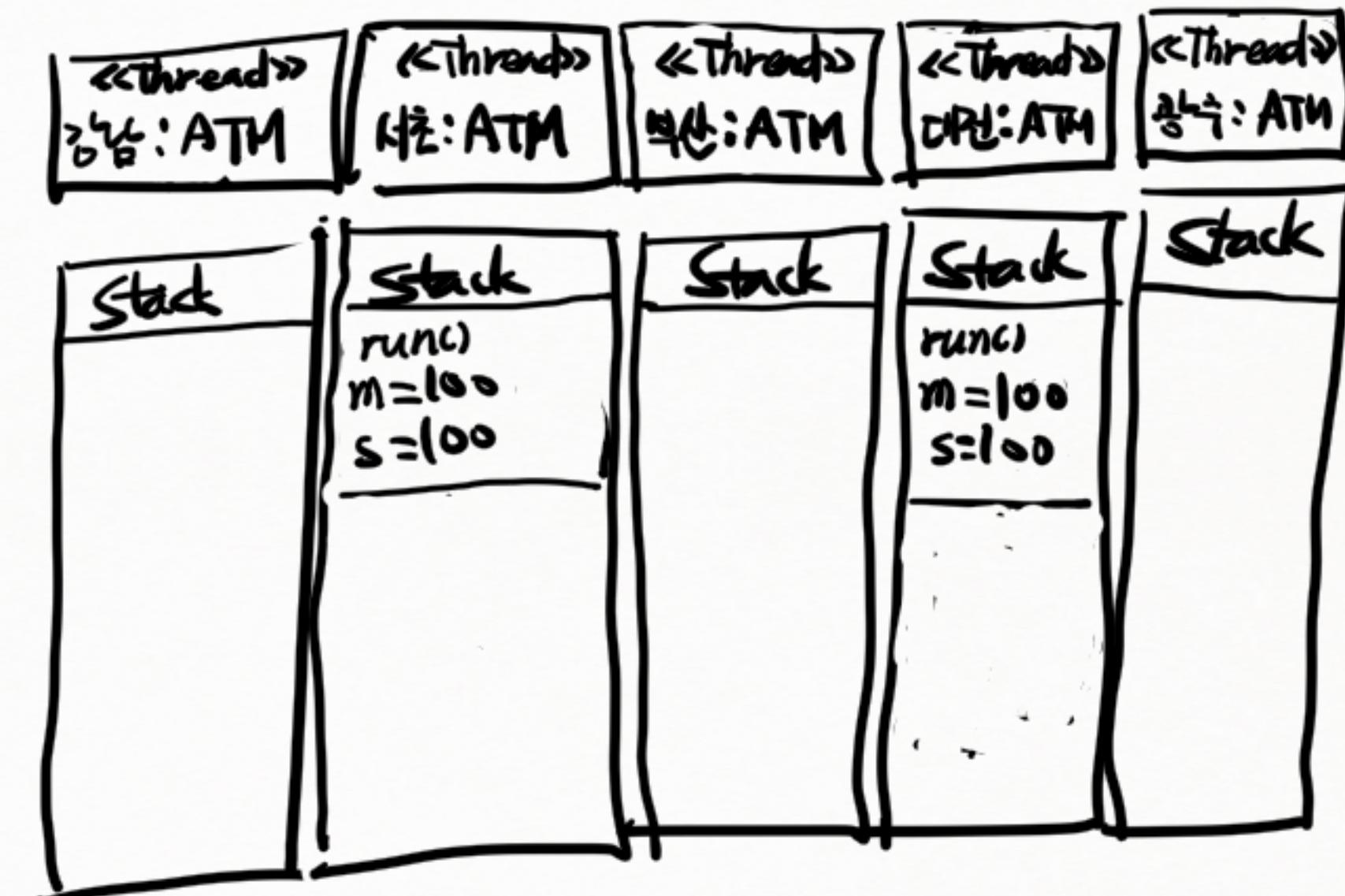
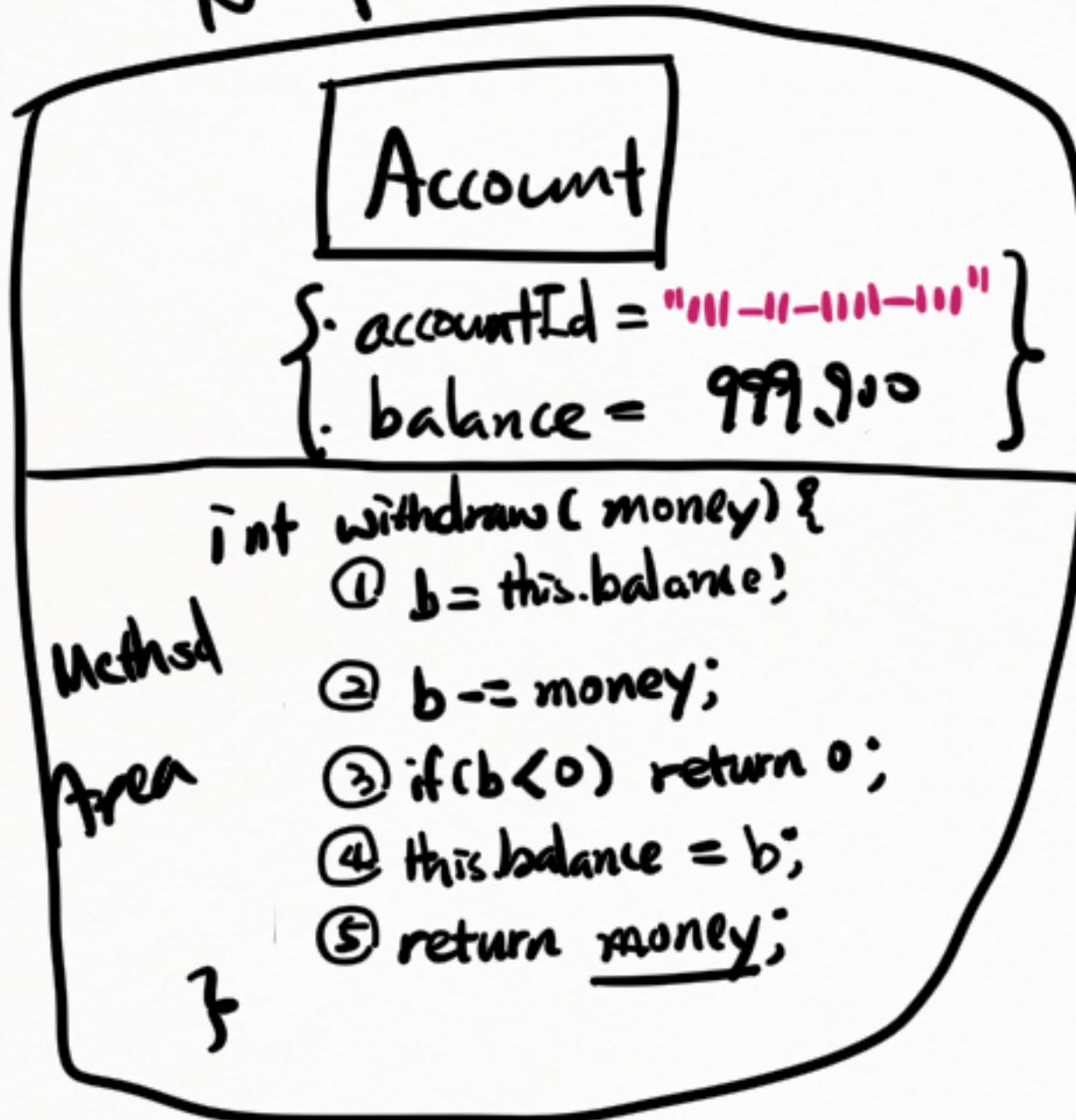


## \* 비동기 실행의 문제점

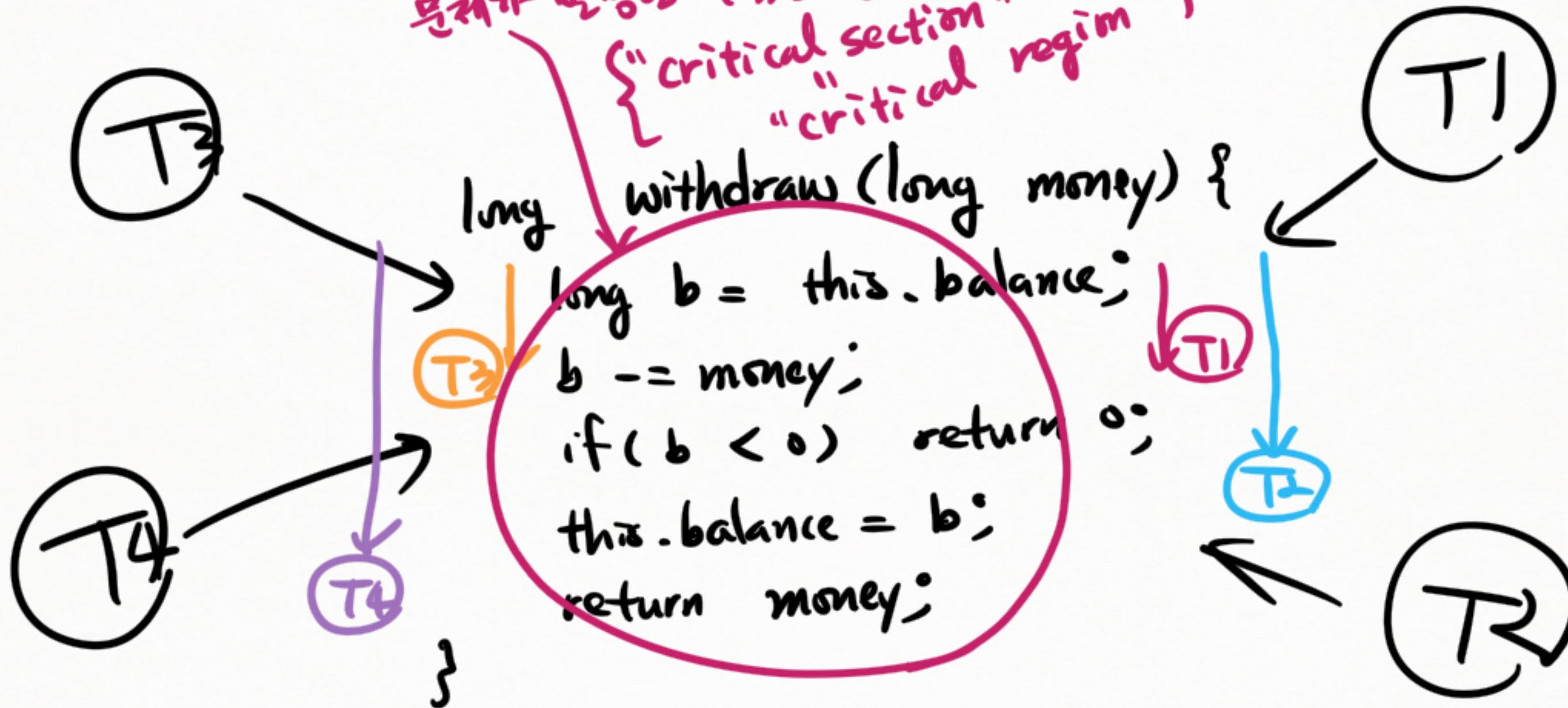


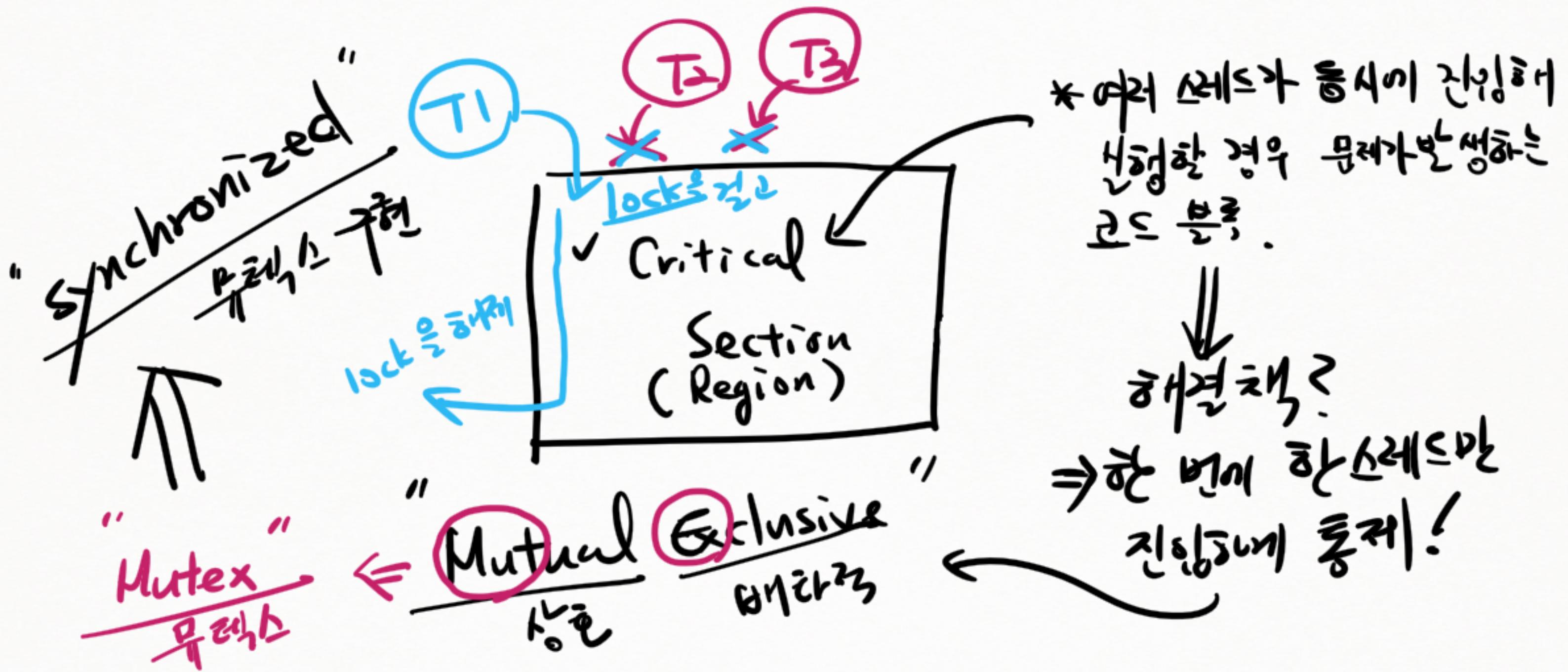
Heap

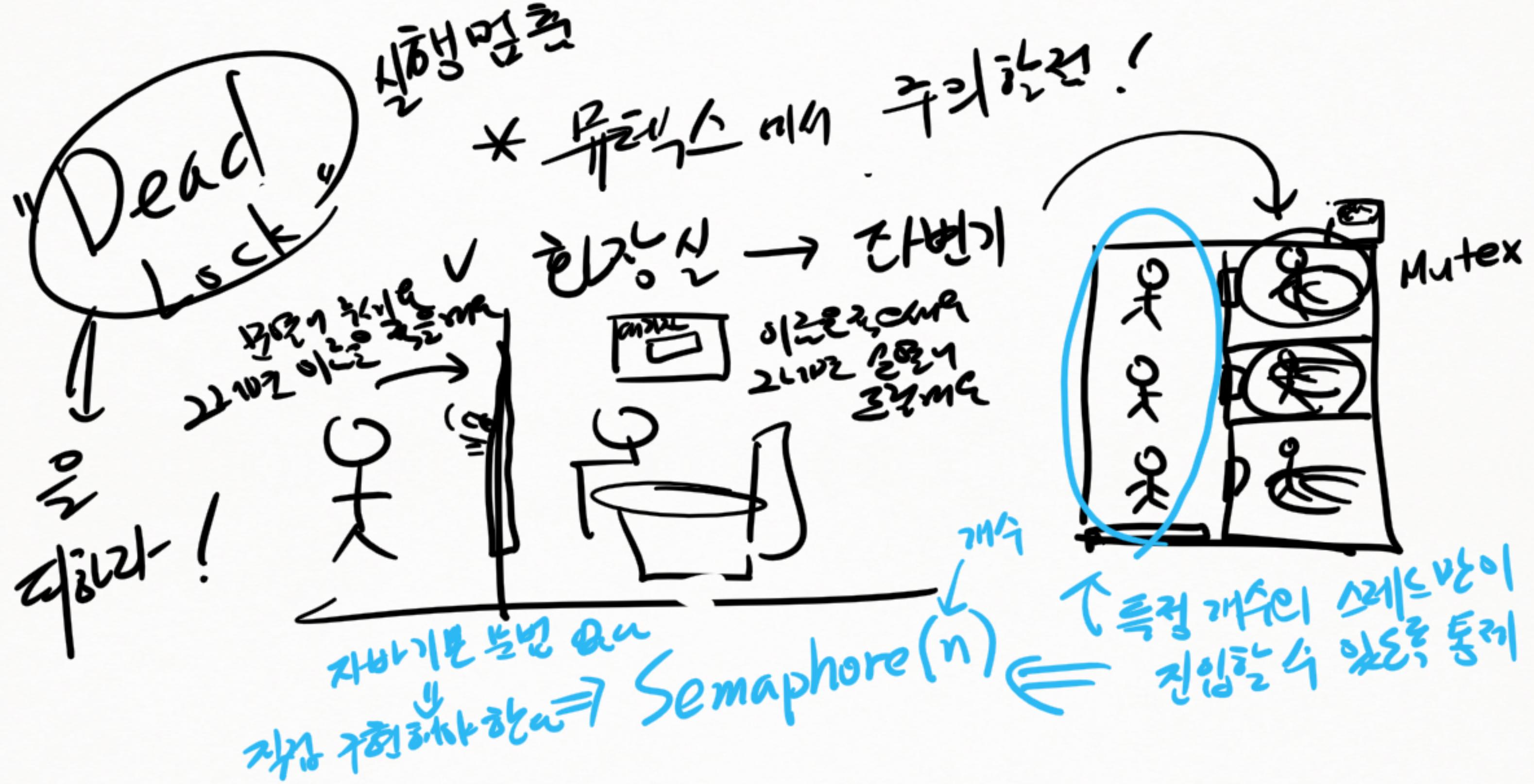
\* 스레드의 Stack 모임



여기 스레드가 동시에 진입하여 실행할 때  
문제가 발생할 수 있는 구간을  
“critical section”  
“critical region”이라 부른다.





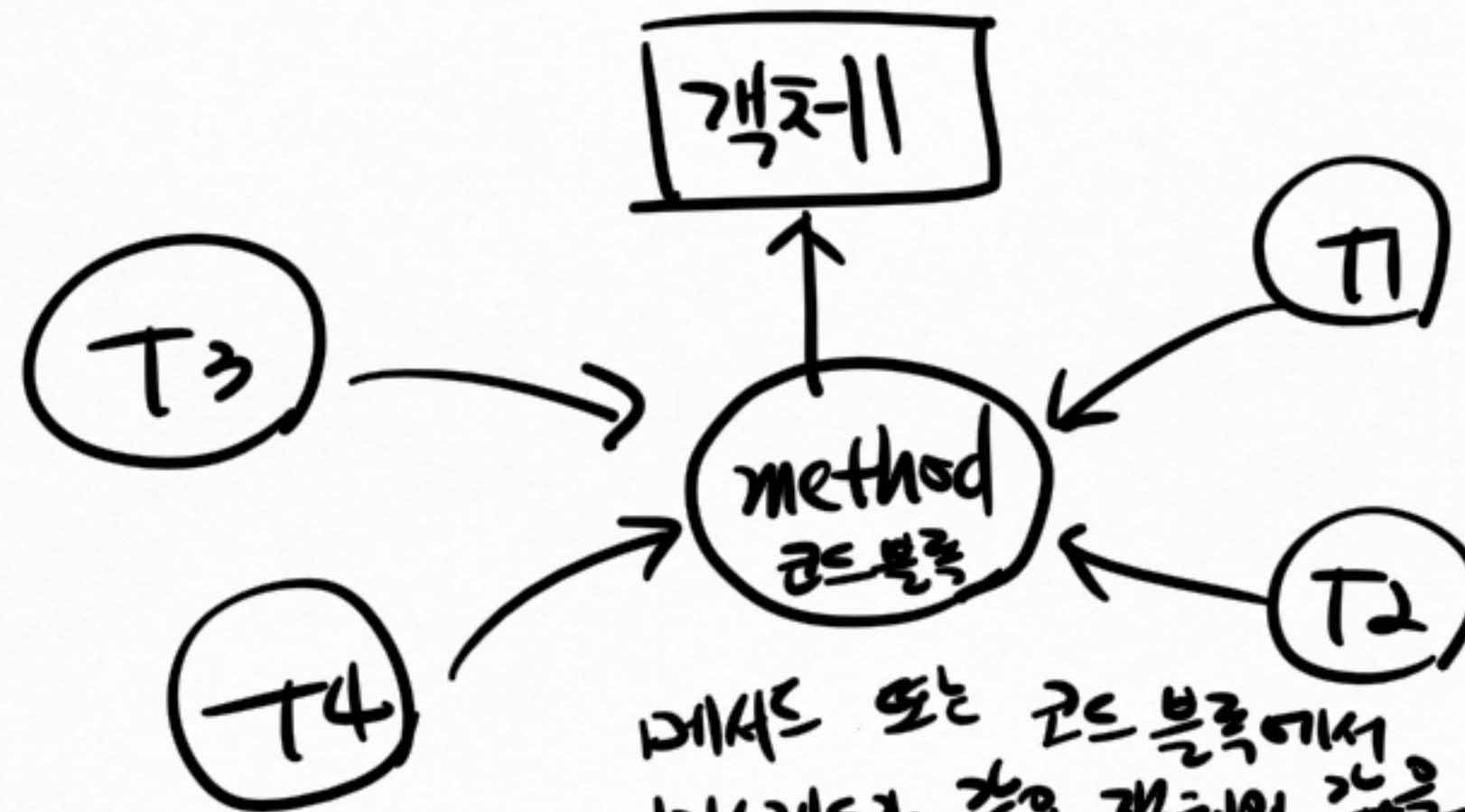


Semaphore( $n$ ) → 한대  $n$  개의 스레드 진입 가능

Semaphore(1) → 한대 1개의 스레드만 진입 가능

||  
"Mutex"

\* 멀티스레드 실행



메서드 또는 코드블록에서  
여러스레드가 같은 객체의 값을  
동시에 변경하여 흔적  
문제 발생

즉 Mutex 형태로 만든다.  
↑  
해당 객체나 코드블록을  
synchronized로  
동기화 처리하면 된다