



Data Analysis

(Parallel Processing and Synchronization 1)

Fall, 2020

Calendar

달력

양음력변환

날짜계산

전역일계산

만나이계산

오늘

<

2020.09

>

☐ 음력

☐ 손없는날

☒ 기념일

일	월	화	수	목	금	토
30	31	1 소개	2 음 7.15	3 환경 세팅	4 지식재산...	5
6	7 백로	8 복습 1	9	10 9.1 복습 2	11	12
13	14	15	16	17 음 8.1	18	19 청년의 날
3주차						
20	21 치매극복...	22	23	24	25	26
4주차						
27	28	29	30	1	2	3
5주차						

Calendar

달력	양음력변환	날짜계산	전역일계산	만나이계산		
<div>오늘<2020.10></div> <div><input type="checkbox"/> 음력<input type="checkbox"/> 손없는날<input checked="" type="checkbox"/> 기념일</div>						
일	월	화	수	목	금	토
27	28	29	30	1 음 8.15 추석 국군의 날	2 노인의 날	3 개천절
4	5 세계 한...	6주차			9 한글날	10
11	12	13	14	15 체육의 날	16 부마민주...	17 음 9.1 문화의 날
18	19	20	21	22	23 상강	24 국제연합일
25 독도의날 중양절	26	27 금유의 날	28 교정의 날	29 지방자치...	30	31 음 9.15
9차						

Calendar

달력

양음력변환

날짜계산

전역일계산

만나이계산

오늘

<

2020.11

>

☐ 음력
☐ 손없는날
☒ 기념일

일	월	화	수	목	금	토
1	2	3	4	5	6	7
		10주차				입동
8	9	10	11	12	13	14
	소방의 날	11주차				
15	16	17	18	19	20	21
음 10.1		12주차				
22	23	24	25	26	27	28
소설		13주차				
29	30	1	2	3	4	5
음 10.15						

Calendar

달력

양음력변환

날짜계산

전역일계산

만나이계산

오늘

<

2020.12

>

☐ 음력
☐ 손없는날
☒ 기념일

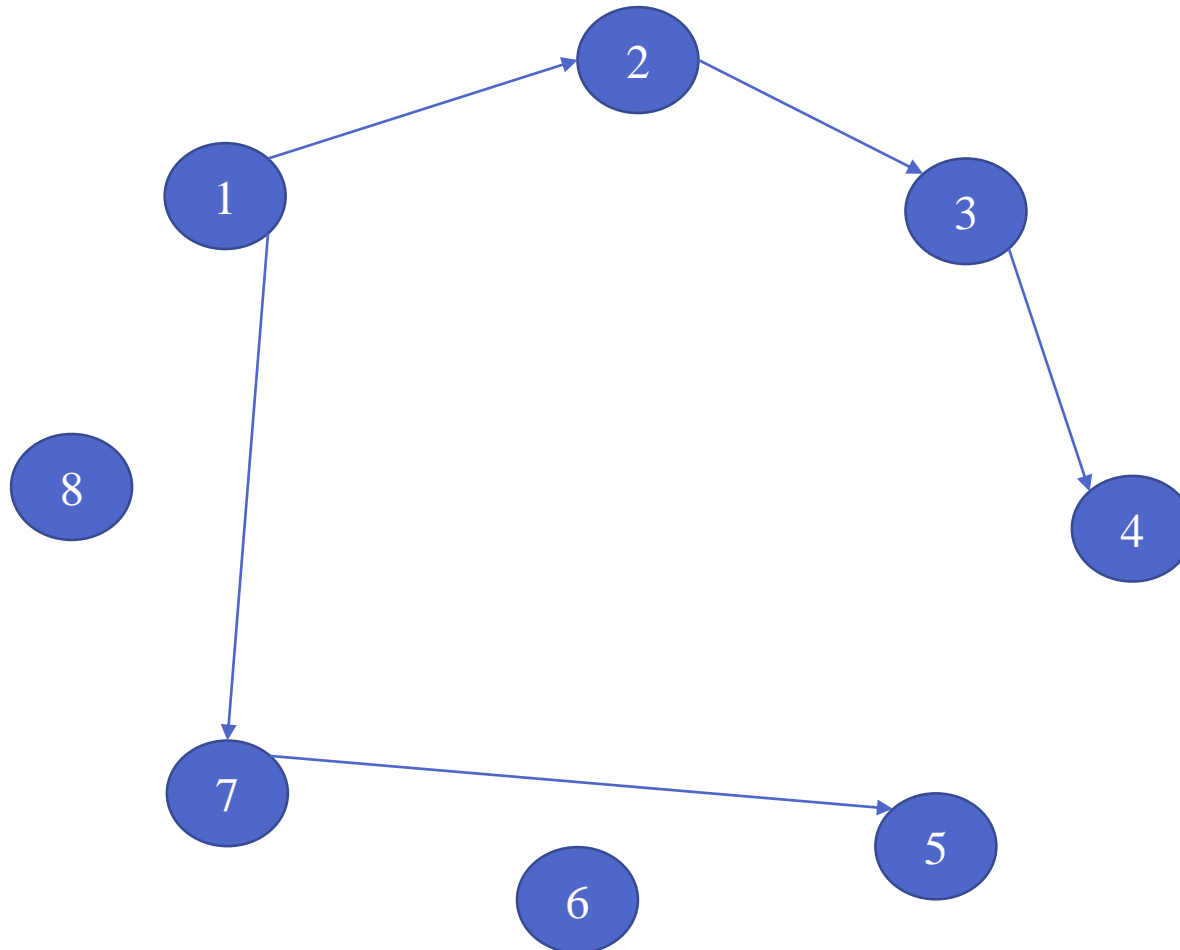
일	월	화	수	목	금	토
29	30	1	2	3	4	5 무역의 날
14주차						
6	7 대설	8	9	10	11	12
15주차						
13	14	15 음 11.1	16	17	18	19
16주차: 기말고사 주간						
20	21 동지	22	23	24	25 성탄절	26
27 원자력의...	28	29 음 11.15	30	31	1	2

Bigger Data

- LiveJournal social network
 - Directed LiveJournal friendship social network
 - `<from int>\t<to int>`
 - <http://snap.stanford.edu/data/soc-LiveJournal1.html>
- Nodes: 4M
- Edges: 68M ($68M * 2 * 4\text{bytes} + a$)
- 1GB
- Try if your machine has good performance

Revisit Question #1

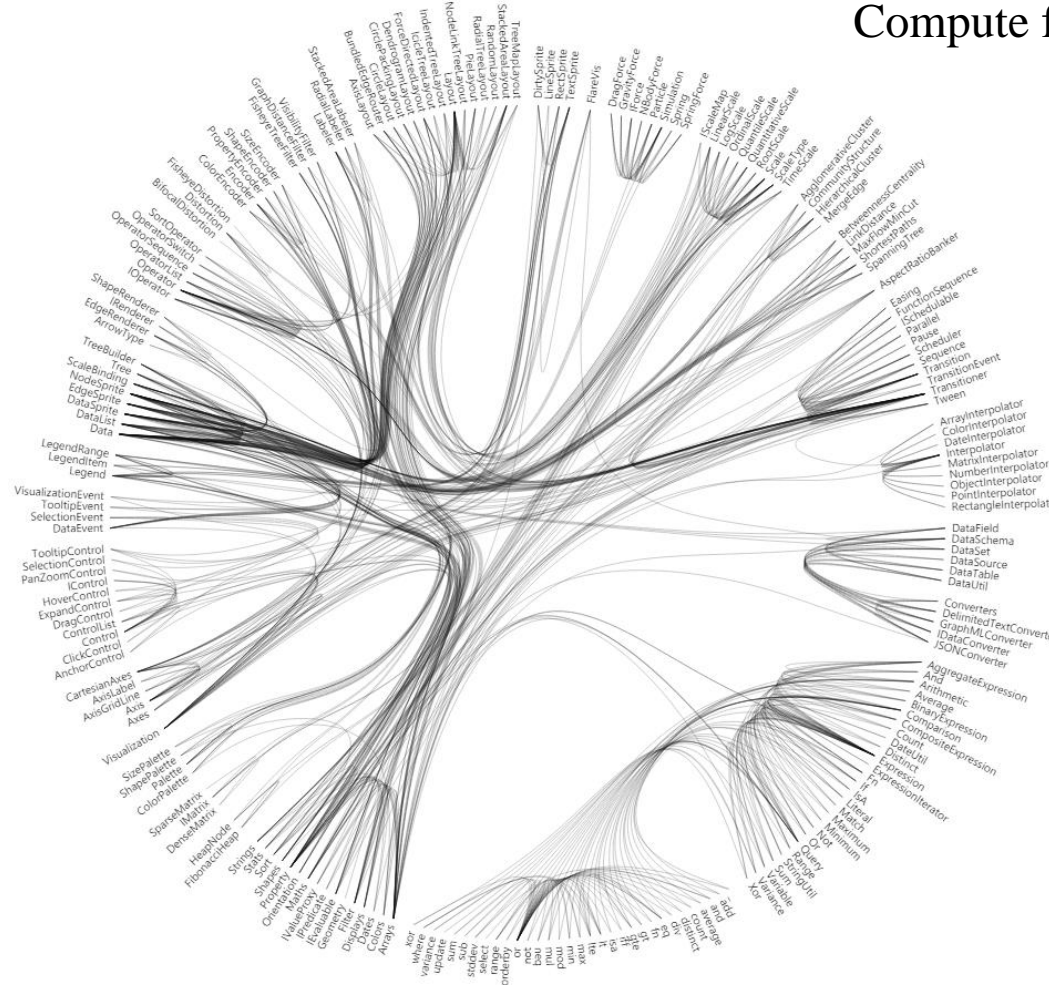
- Compute the number of friend of friend of friend?
 - Friend of friend of 1: [4, 5]



Revisit Question #1

- Compute the number of friend of friend of friend?
 - HashMap<Integer, HashSet<Integer>>

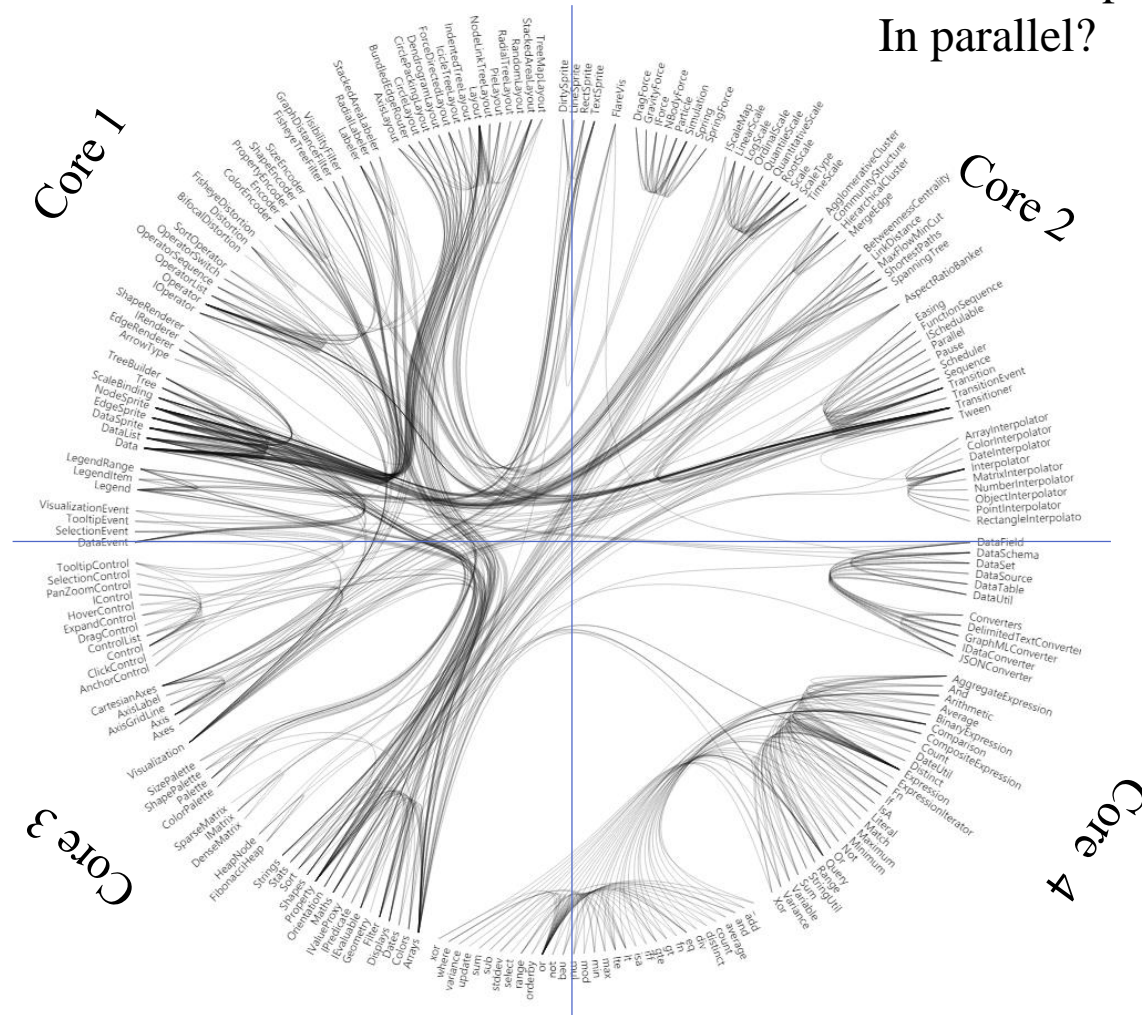
For each key 'k',
Compute friend of friend of k



Revisit Question #1

- Compute the number of friend of friend of friend?
 - HashMap<Integer, HashSet<Integer>>

How about processing it
In parallel?



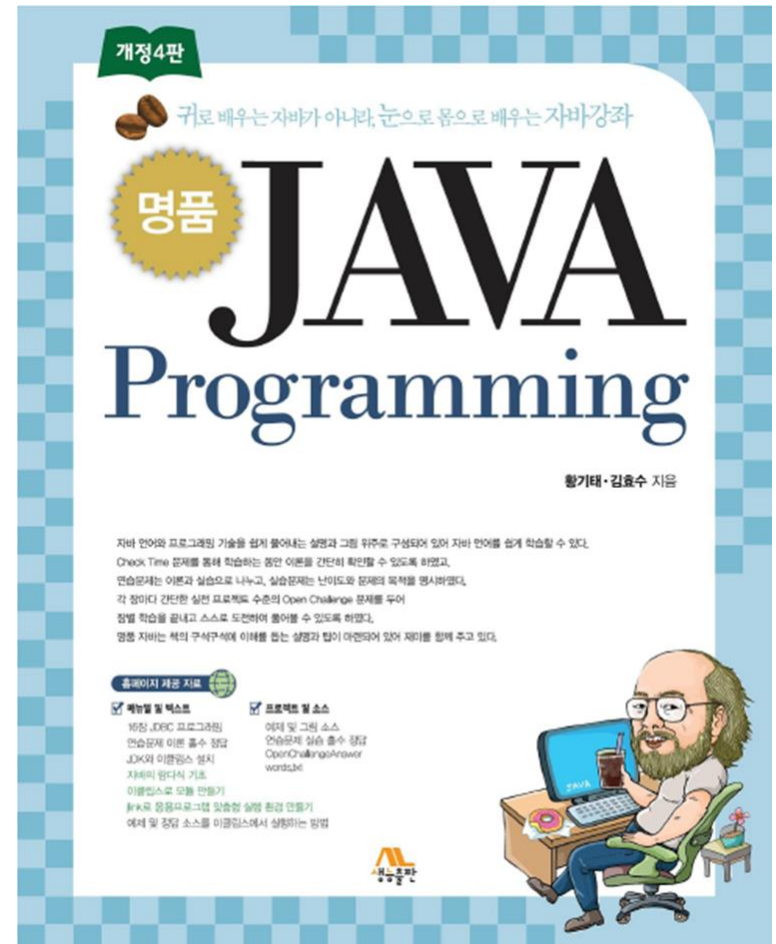
Multi-thread programming

Table of Contents

- Parallel Processing and Synchronization
 - Motivating example
 - Thread programming
 - Synchronization problem
 - ...

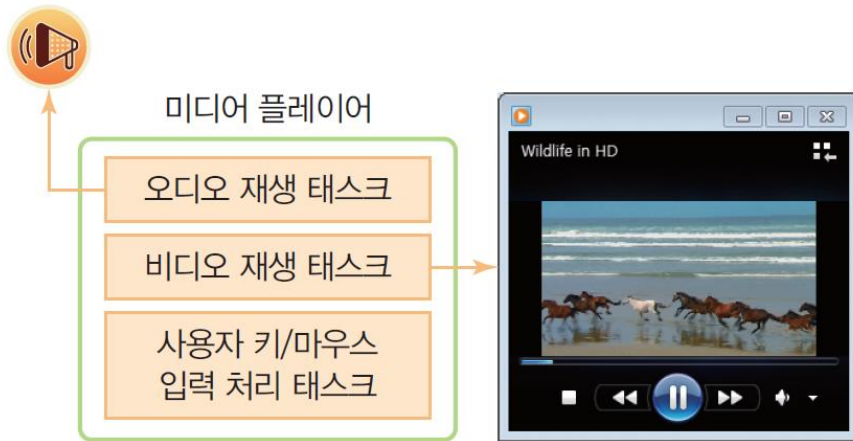
Reference

- 명품 Java Programming (황기태, 김효수)
- <https://www.booksr.co.kr/html/book/book.asp?seq=697068>



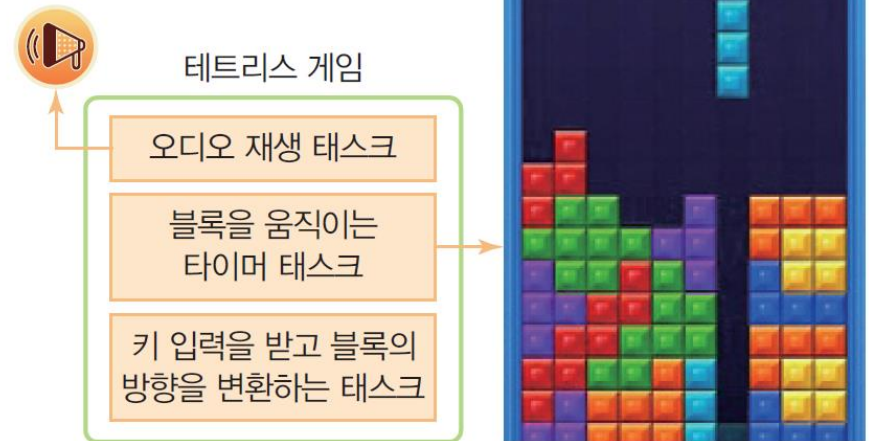
Multi-tasking

- One program executes multiple tasks



(a) 미디어 플레이어의 멀티태스킹

* 3개의 태스크 동시 실행

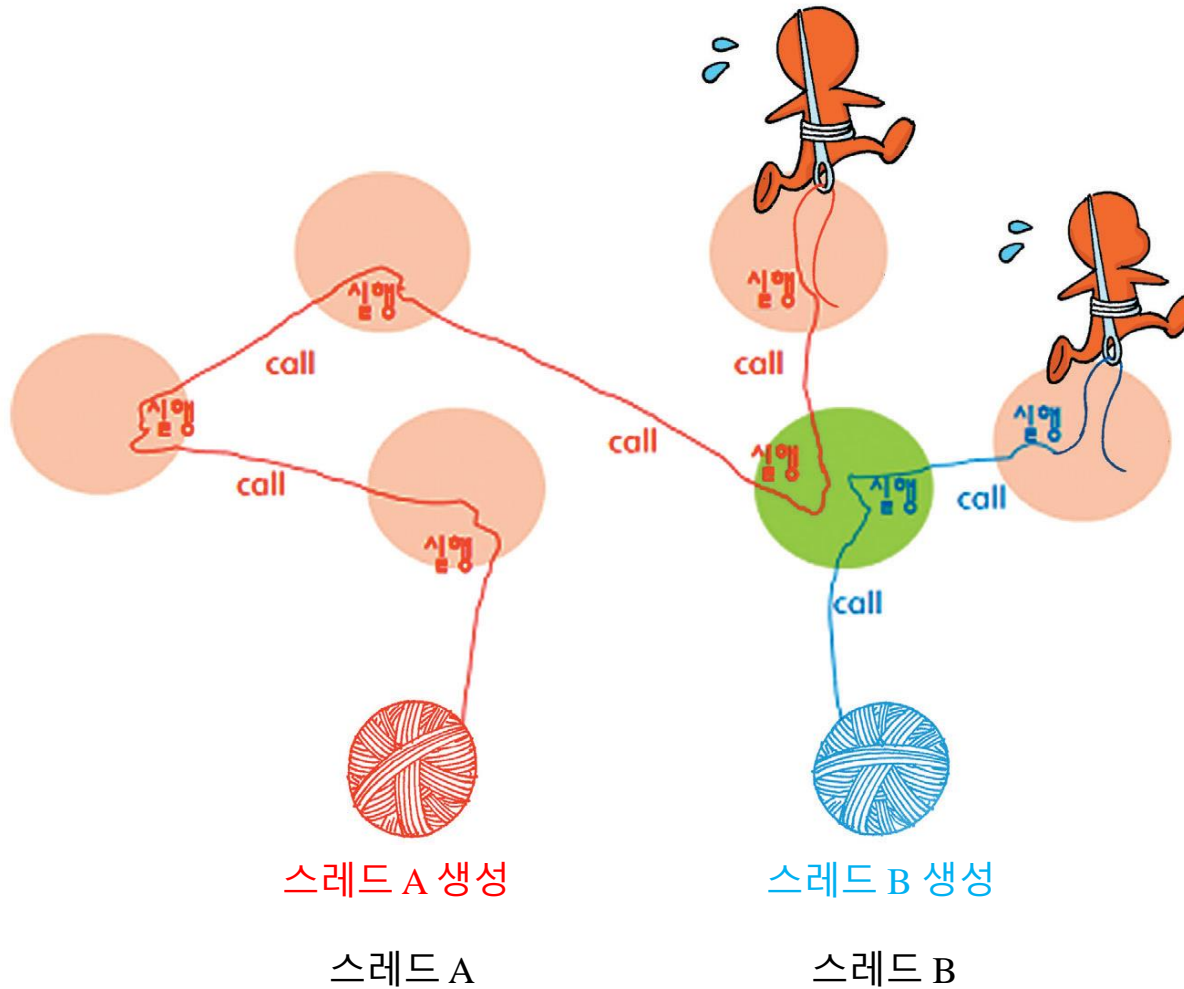


(b) 테트리스 게임의 멀티태스킹

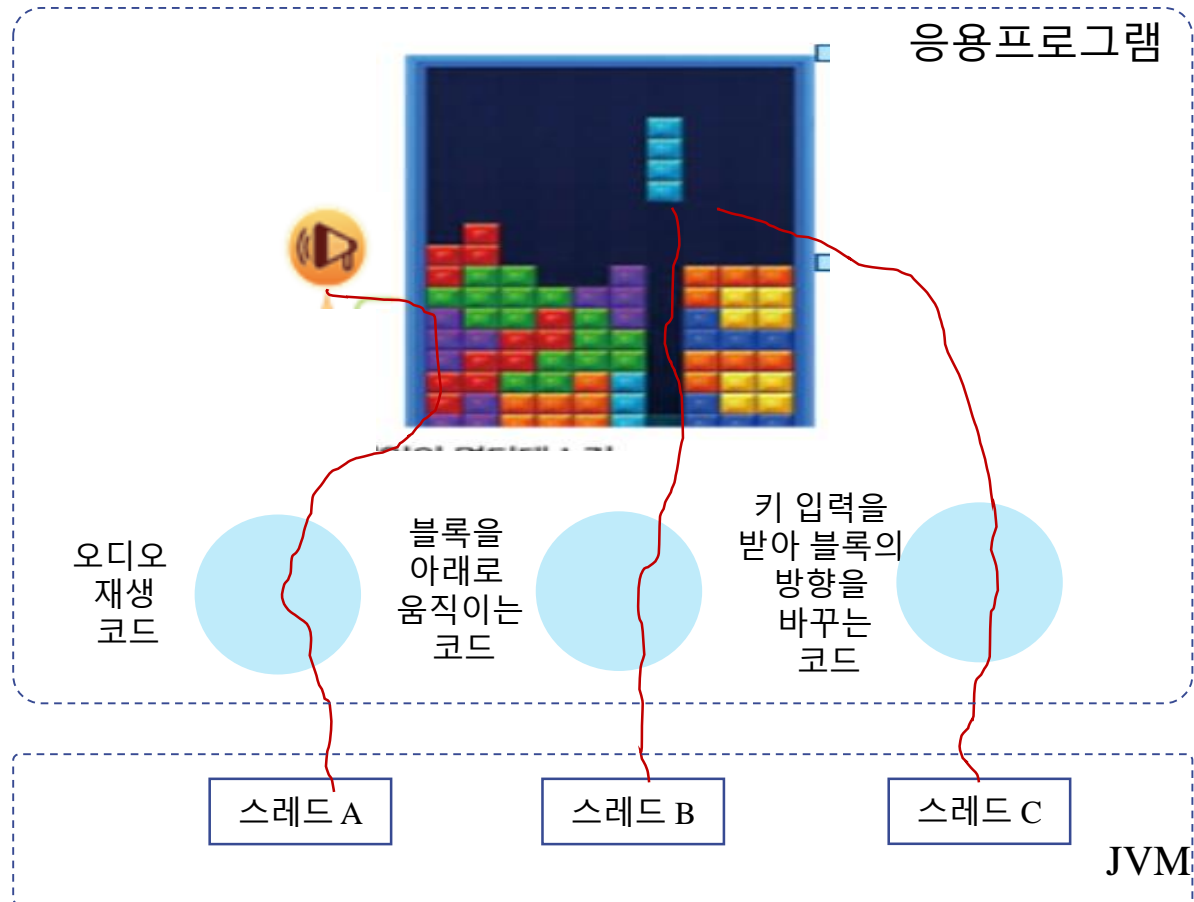
* 3개의 태스크 동시 실행

Thread

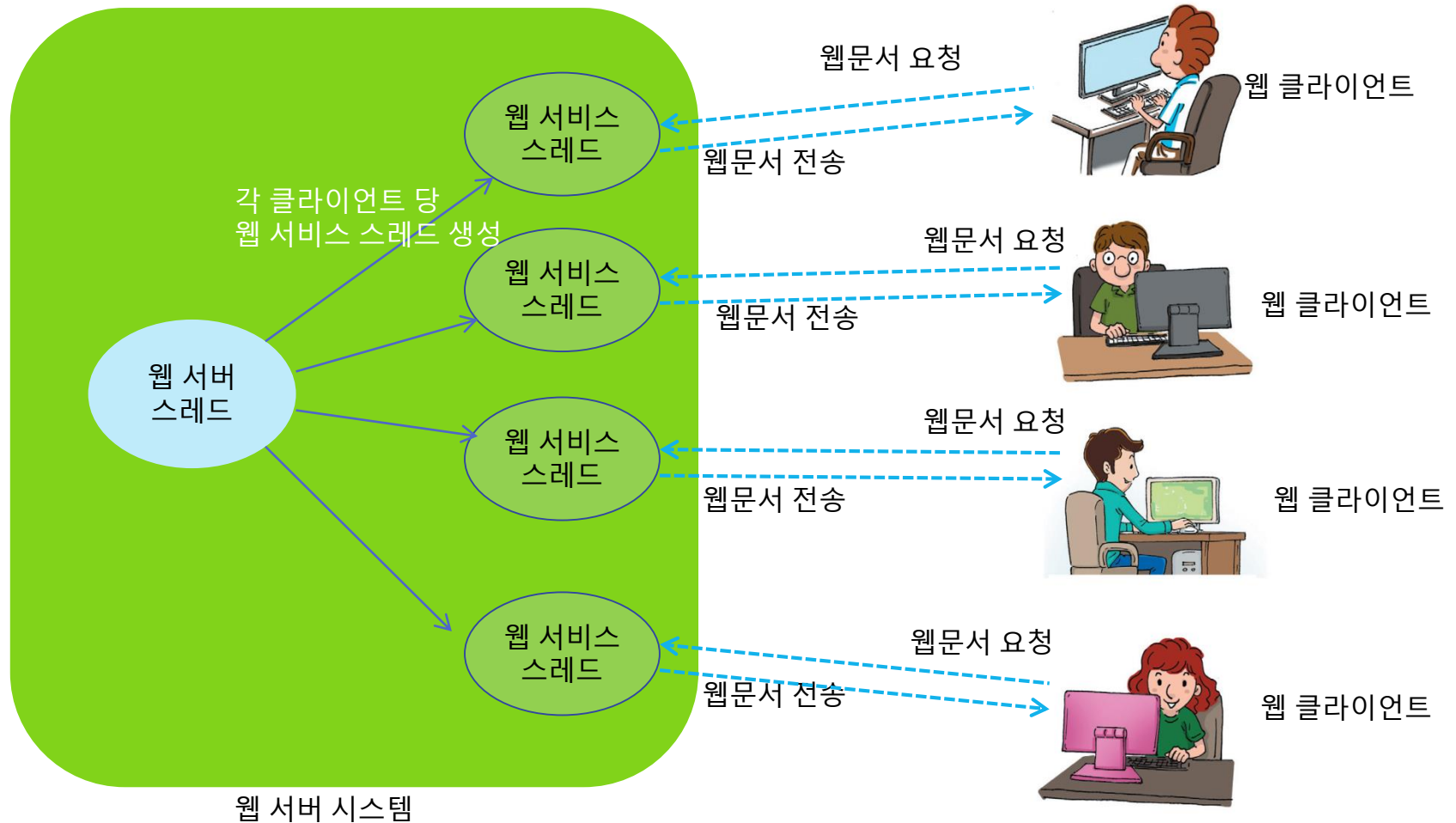
- Java provides Thread for multi-tasking



Thread Example

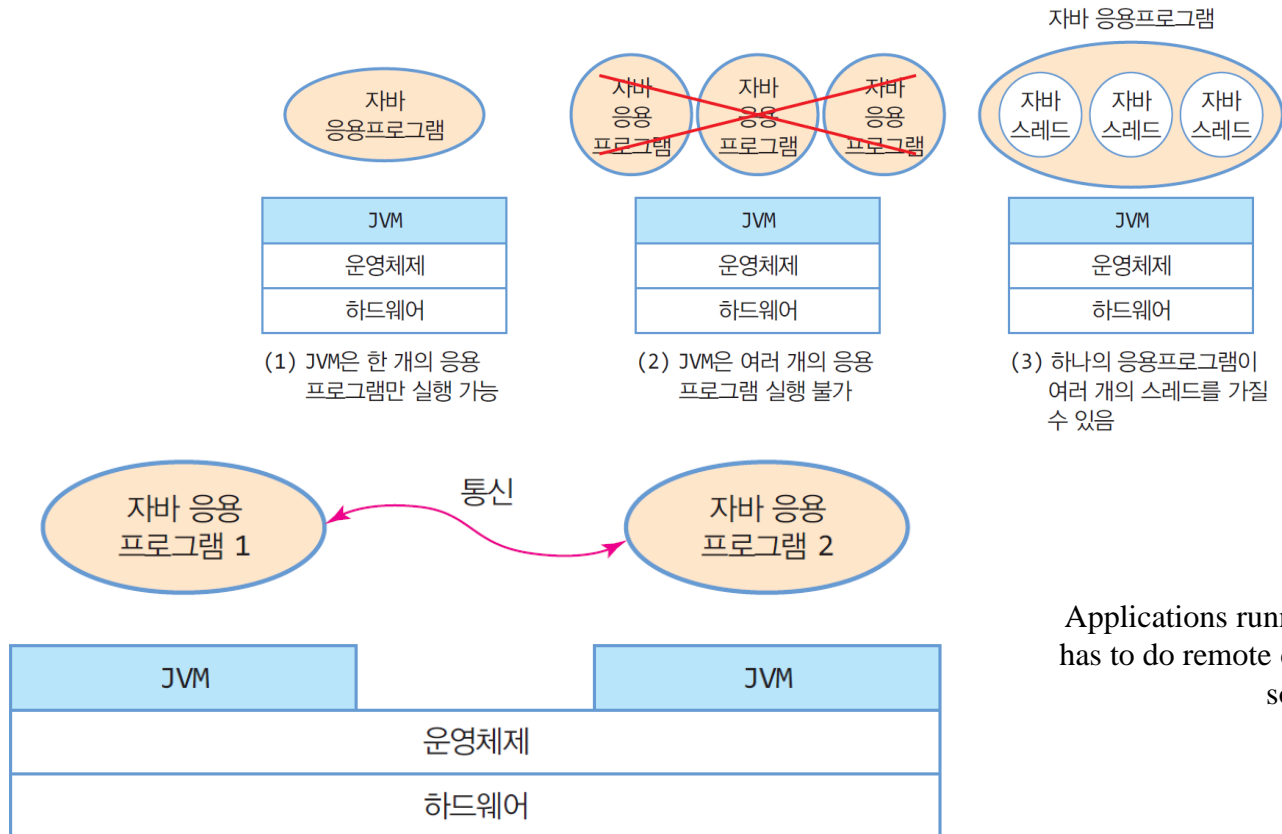


Thread Example



Java Thread

- An execution unit scheduled by Java Virtual Machine (JVM)
- One JVM runs one Java application.
- One Java application could have one or more thread(s).




Applications running in different JVM has to do remote communications (e.g., socket)

How to create Java Thread

1. Create Thread
 - By extending `java.lang.Thread`
 - By implementing `java.lang.Runnable`
2. Request to run Thread

Create Thread: by extending Thread class


- Create your class extending Thread class



```
class TimerThread extends Thread {  
    .....  
    @Override  
    public void run() { // run() overriding  
        .....  
    }  
}
```

- Overriding run() method

- Instantiate a thread class



```
TimerThread th = new TimerThread();
```

- And start() (not run())

- JVM will schedule the thread



```
th.start();
```

Practice 1

- Hello World Thread
 - Note: After run() is done, thread will be terminated
 - Infinite loop? → use while(true) inside run()
 - Cannot be re-executed
 - Can be terminated by other threads

```
public class HelloWorldThread extends Thread {  
    @Override  
    public void run() {  
        System.out.println("Hello");  
    }  
}
```

```
public class App {  
    public static void main(String[] args) {  
        HelloWorldThread t1 = new HelloWorldThread();  
        t1.start();  
    }  
}
```

Practice 2

- Feel concurrency
- Multiple threads have its own message and print out 10000 times
 - App
 - Thread1 → print “A” infinitely
 - Thread2 → print “\tB” infinitely

A
A
A
A
A


B
B
B
B
B

A
A
A
A
A

B
B
B
B
B

Create Thread by implementing Runnable interface

- Create your class implementing Runnable interface



```
class TimerRunnable implements Runnable {  
    .....  
    @Override  
    public void run() { // run() 메소드 구현  
        .....  
    }  
}
```

- Overriding run() method
 - Your own logic

- Create a thread using your runnable



```
Thread th = new Thread(new TimerRunnable());
```

- Start the thread with start()



```
th.start();
```

Practice 3

- Practice 2 with Runnable

A
A
A
A
A

B
B
B
B
B

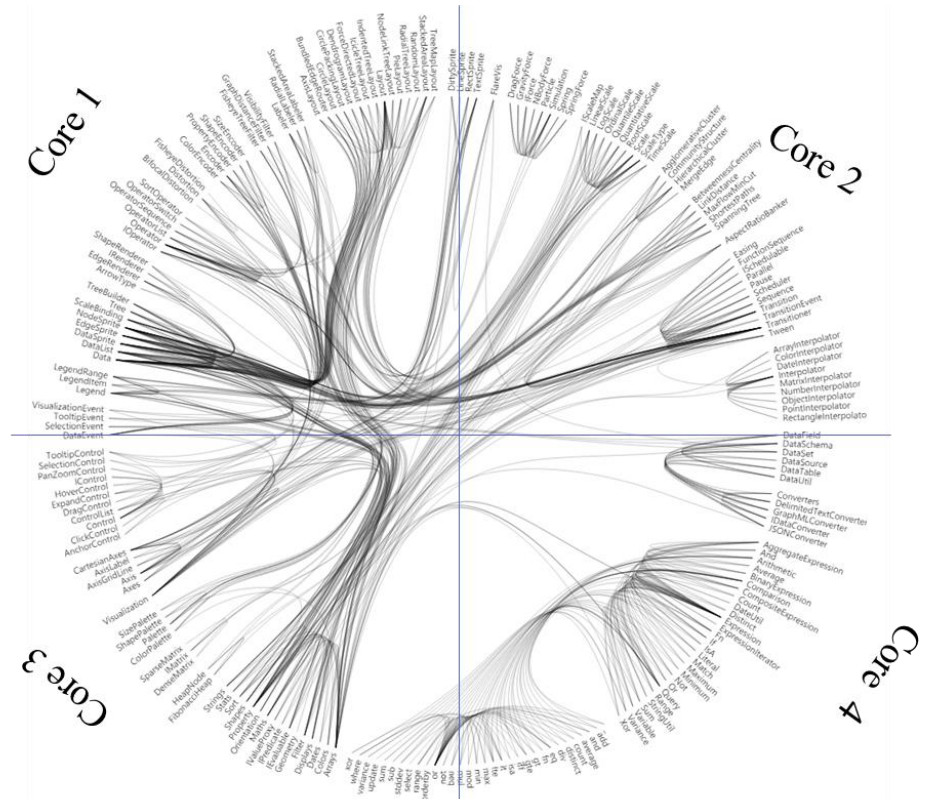
A
A
A
A
A

B
B
B
B
B

Practice 4

How about processing it
In parallel?

- Compute the number of friend of friend of friend?
 - `HashMap<Integer, HashSet<Integer>>`
 - Each thread process quarter of whole data
 - (0 to `data.length/4`) ...



Thread Information

필드	타입	내용
스레드 이름	스트링	스레드의 이름으로서 사용자가 지정
스레드 ID	정수	스레드 고유의 식별자 번호
스레드의 PC(Program Count)	정수	현재 실행 중인 스레드 코드의 주소
스레드 상태	정수	NEW, RUNNABLE, WAITING, TIMED_WAITING, BLOCK, TERMINATED 등 6개 상태 중 하나
스레드 우선순위	정수	스레드 스케줄링 시 사용되는 우선순위 값으로서 1~10 사이의 값이며 10이 최상위 우선순위
스레드 그룹	정수	여러 개의 자바 스레드가 하나의 그룹을 형성할 수 있으며 이 경우 스레드가 속한 그룹
스레드 레지스터 스택	메모리 블록	스레드가 실행되는 동안 레지스터들의 값

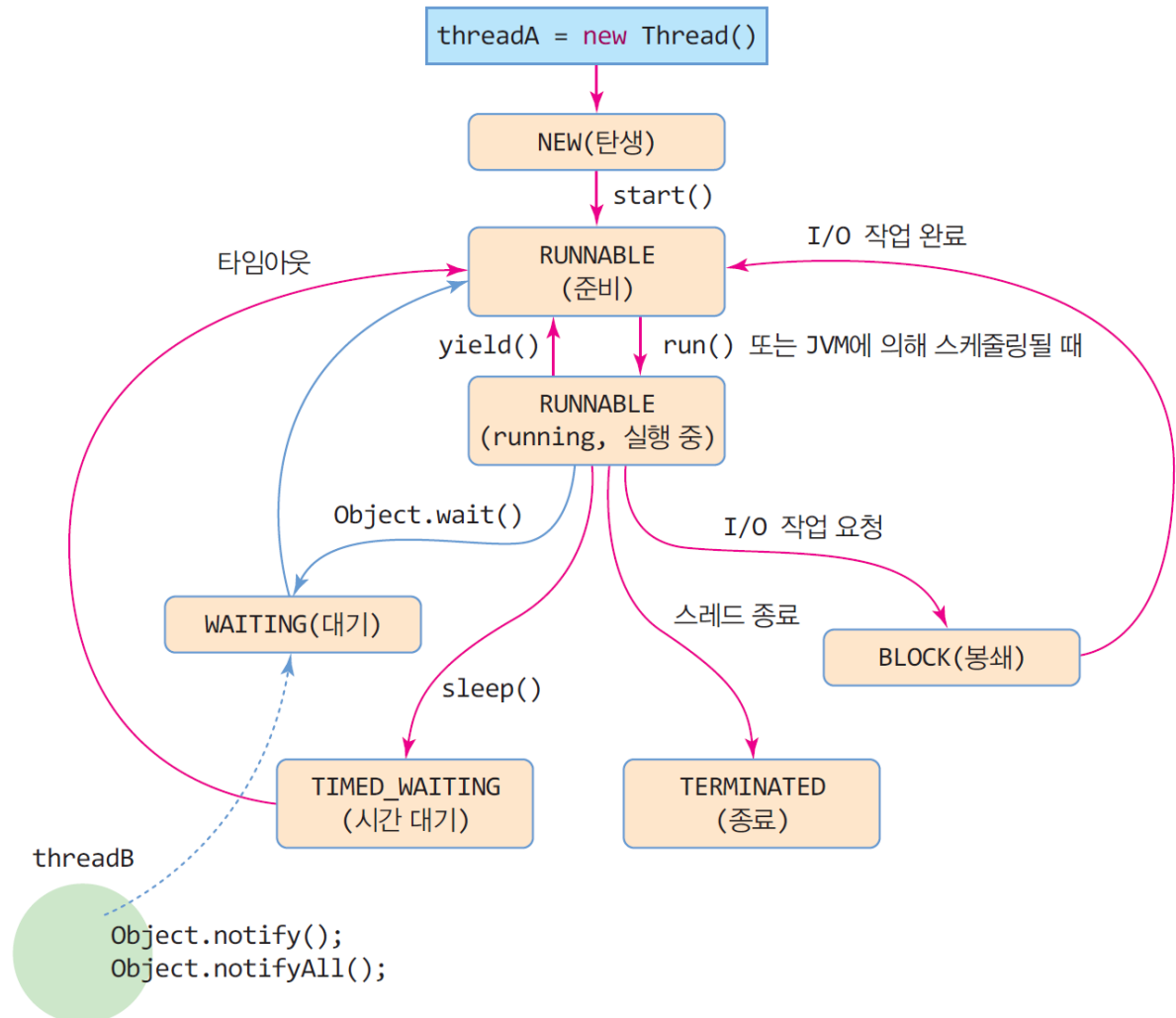
스레드 상태

- 스레드 상태 6 가지
 - NEW
 - 스레드가 생성되었지만 스레드가 아직 실행할 준비가 되지 않았음
 - RUNNABLE
 - 스레드가 현재 실행되고 있거나
 - 실행 준비되어 스케줄링을 기다리는 상태
 - WAITING
 - wait()를 호출한 상태
 - 다른 스레드가 notify()나 notifyAll()을 불러주기를 기다리고 있는 상태
 - 스레드 동기화를 위해 사용
 - TIMED_WAITING
 - sleep(n)을 호출하여 n 밀리초 동안 잠을 자고 있는 상태
 - BLOCK
 - 스레드가 I/O 작업을 요청하면 JVM이 자동으로 BLOCK 상태로 만듦
 - TERMINATED
 - 스레드가 종료한 상태
- 스레드 상태는 JVM에 의해 기록 관리됨

스레드 상태와 생명 주기

스레드 상태 6 가지

- NEW
- RUNNABLE
- WAITING
- TIMED_WAITING
- BLOCK
- TERMINATED



** wait(), notify(), notifyAll()은 Thread의 메소드가 아니며 Object의 메소드임

스레드 우선순위와 스케줄링

- 스레드의 우선순위
 - 최대값 = 10(MAX_PRIORITY)
 - 최소값 = 1(MIN_PRIORITY)
 - 보통값 = 5(NORMAL_PRIORITY)
- 스레드 우선순위는 응용프로그램에서 변경 가능
 - `void setPriority(int priority)`
 - `int getPriority()`
- `main()` 스레드의 우선순위 값은 초기에 5
- 스레드는 부모 스레드와 동일한 우선순위 값을 가지고 탄생
- JVM의 스케줄링 정책
 - 철저한 우선순위 기반
 - 가장 높은 우선순위의 스레드가 우선적으로 스케줄링
 - 동일한 우선순위의 스레드는 돌아가면서 스케줄링(라운드 로빈)

Practice 4

- Practice how to set name, get id, priority, see how a state changes
- Make a HelloWorldThread printing a message 5 times
- Print out the followings

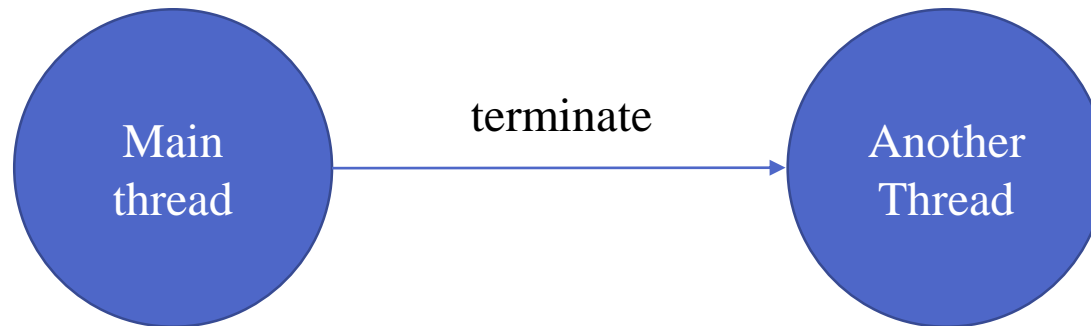
```
Thread name: T1
Thread id: 19
Thread state: NEW
Thread priority: 3
Thread state: RUNNABLE
Thread state: RUNNABLE
A
A
A
A
A
Thread state: TERMINATED
```

A thread for main()

- JVM creates a thread for main()
- The thread executes the main() method

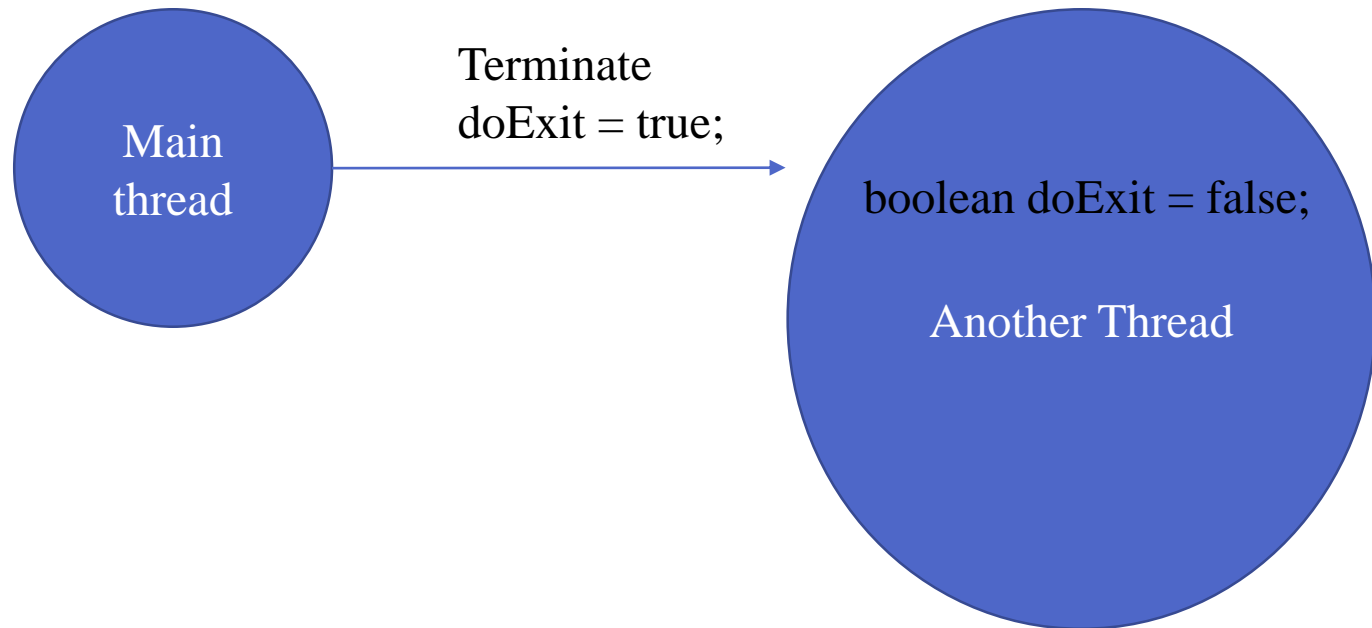
Interaction between threads

- Termination?



Interaction between threads

- Termination?
 - 1. Flag
 - Have a variable in a class
 - According to the value of the variable, stop the execution



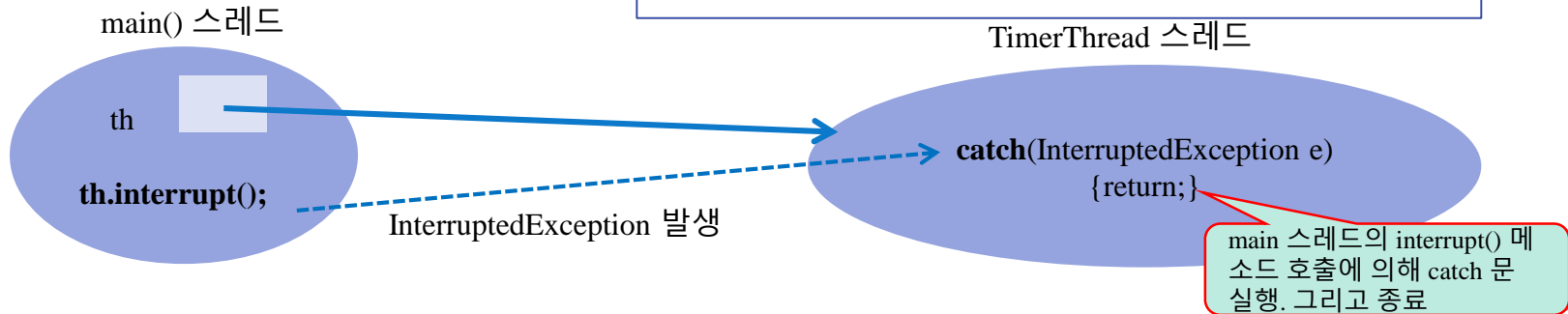
Interaction between threads

- Termination?
 - Invoke `interrupt()` of 'A' thread
 - Then, if A is in sleep, the interrupted exception occurs.

```
public static void main(String [] args) {  
    TimerThread th = new TimerThread();  
    th.start();  
  
    th.interrupt(); // TimerThread 강제 종료  
}
```

```
class TimerThread extends Thread {  
    private int n = 0;  
    @Override  
    public void run() {  
        while(true) {  
            System.out.println(n); // 화면에 카운트 값 출력  
            n++;  
            try {  
                sleep(1000);  
            }  
            catch (InterruptedException e) {  
                return; // 예외를 받고 스스로 리턴하여 종료  
            }  
        }  
    }  
}
```

만일 return 하지 않으면
스레드는 종료하지 않음



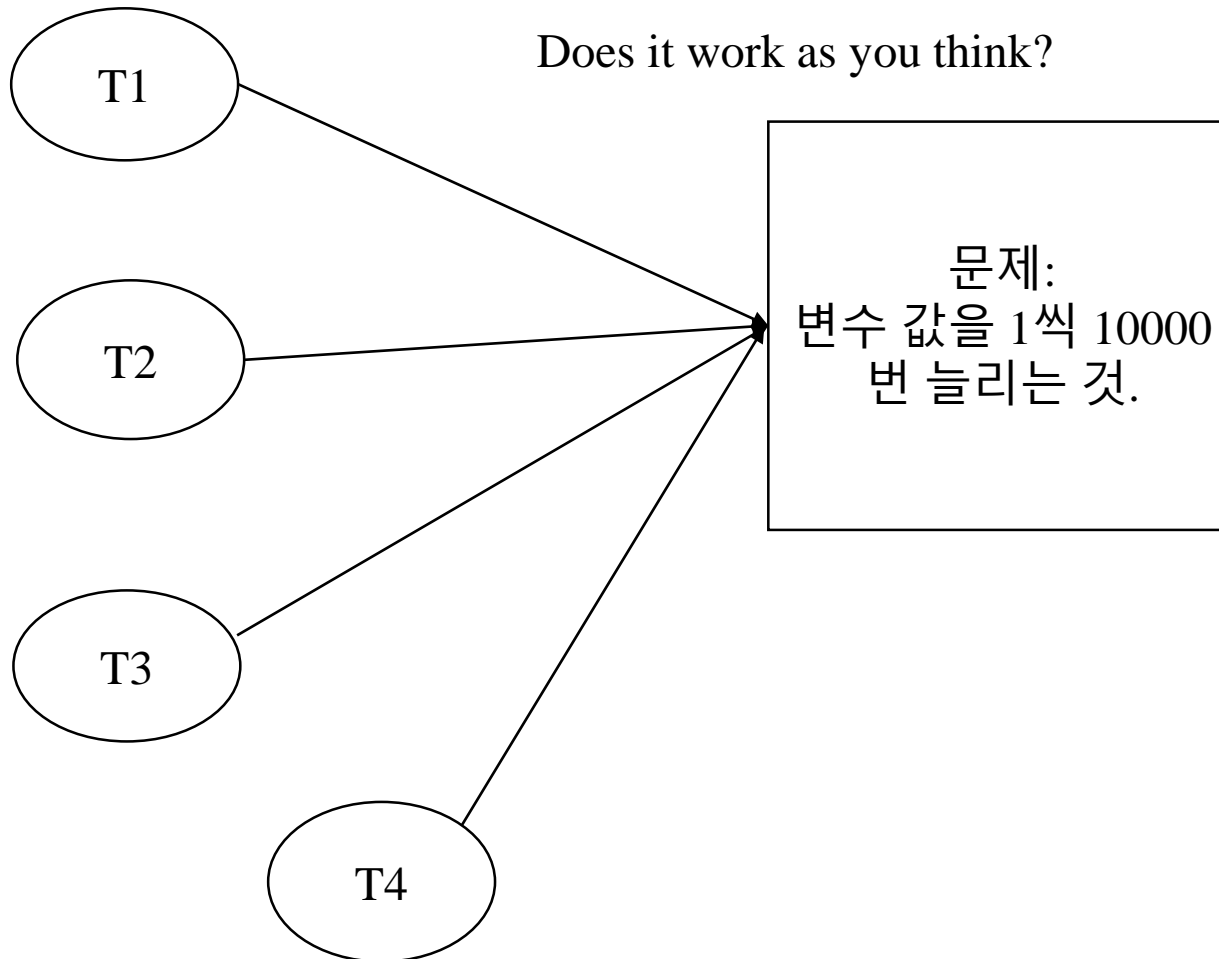
Practice 6

- Using try/catch statement
- Feel how a thread is interrupted
- Make HelloWorldThread
- Inside run(), using Thread.interrupted() → break;
- Main thread interrupts the thread

Thread.interrupted(): Tests whether the current thread has been interrupted. The *interrupted status* of the thread is cleared by this method. In other words, if this method were to be called twice in succession, the second call would return false (unless the current thread were interrupted again, after the first call had cleared its interrupted status and before the second call had examined it).

Synchronization Problem

- `int val = 0;`
- Four threads increases the val by one 10000 times.



Summary

- Motivating example
- Thread programming
- Synchronization problem