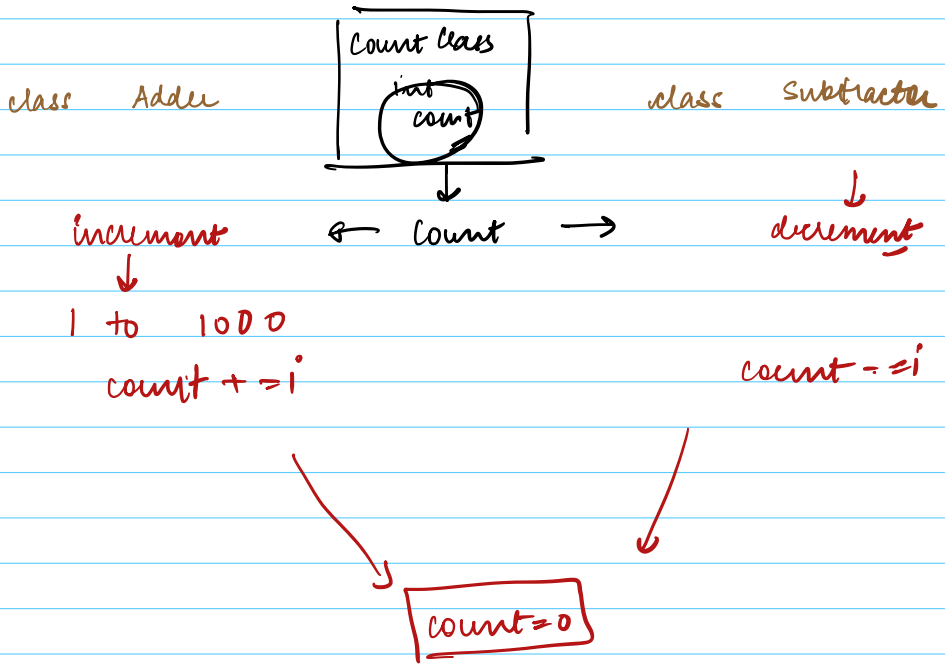
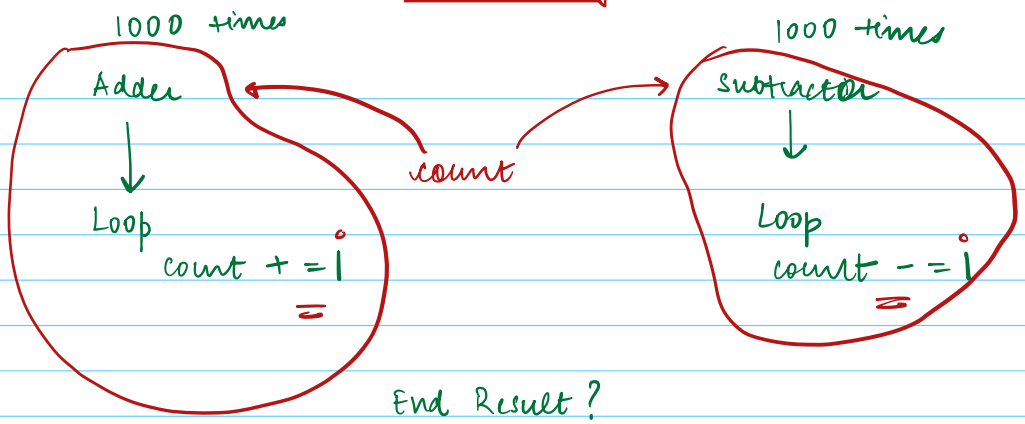
AGENDA

1) Problems with Multi-threading programs
   (i) Adder Subtractor problem
   (ii) Problem of Synchronization?
   (iii) conditions leading to problem

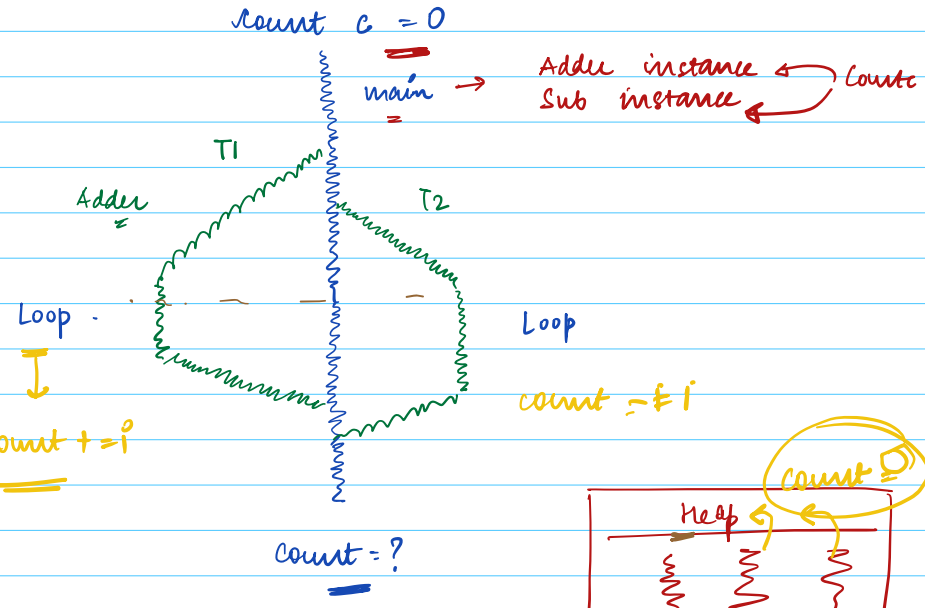
2) Solutions
   (i) Theoretical solution
   (ii) Practical solution
      a) Mutex
      b) Synchronized
      c) Atomic variables
      d) Semaphores.

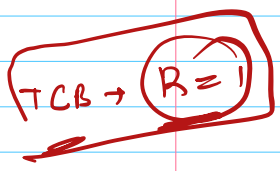


Data sharing

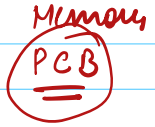


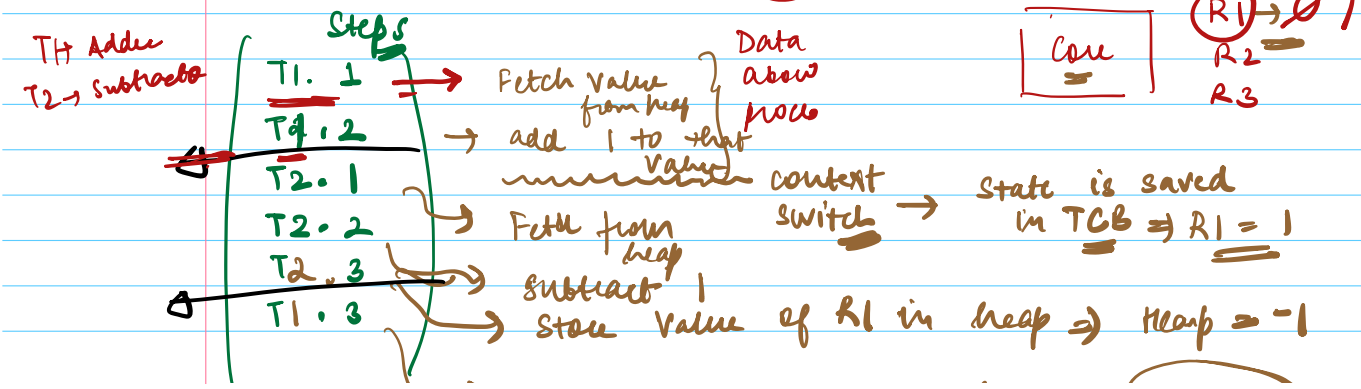
End Result ?





Practical





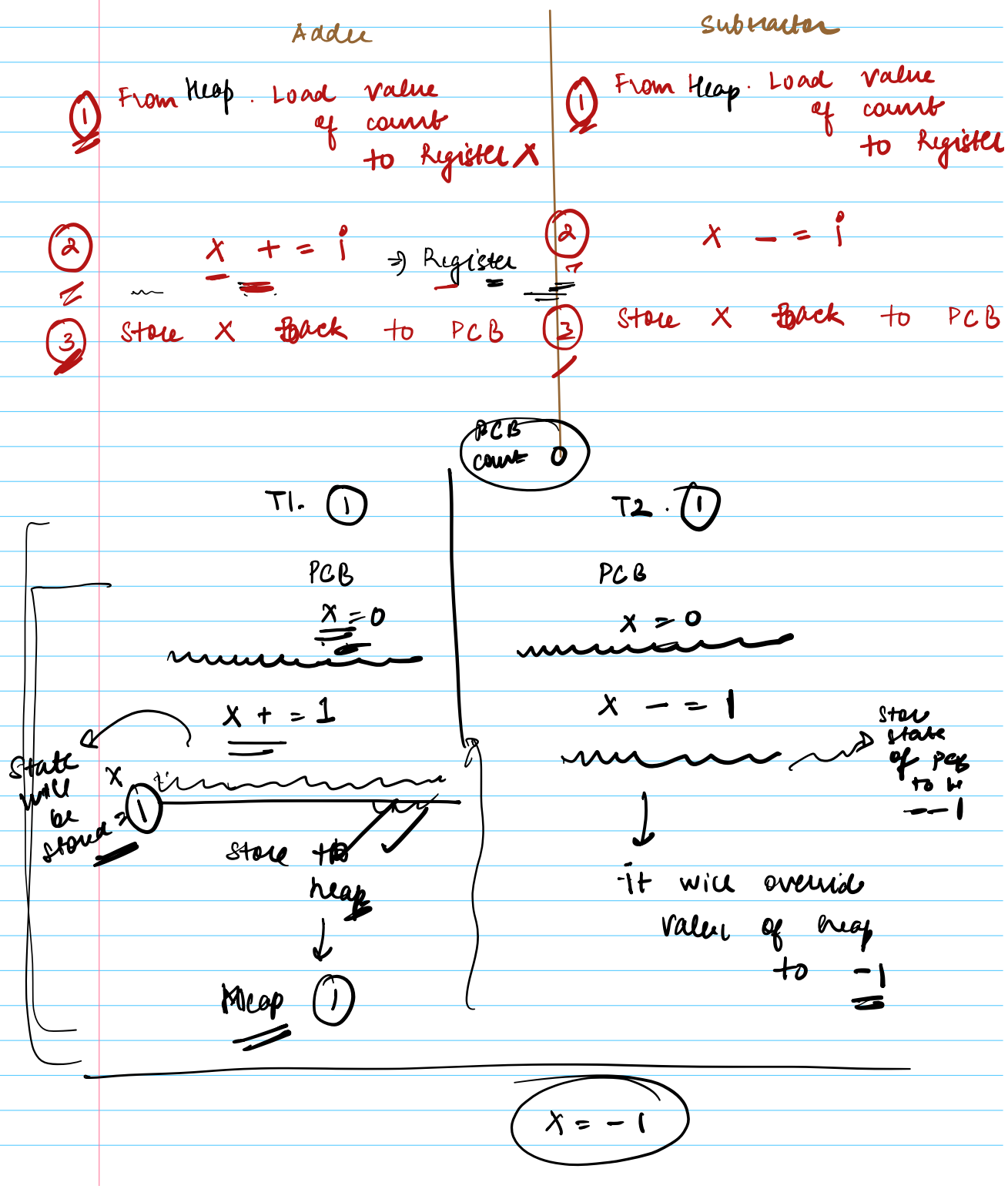
1

↳ Store value of R1 in heap ⇒ (heap = 1)

(1)

Expect = 0

Explanation:



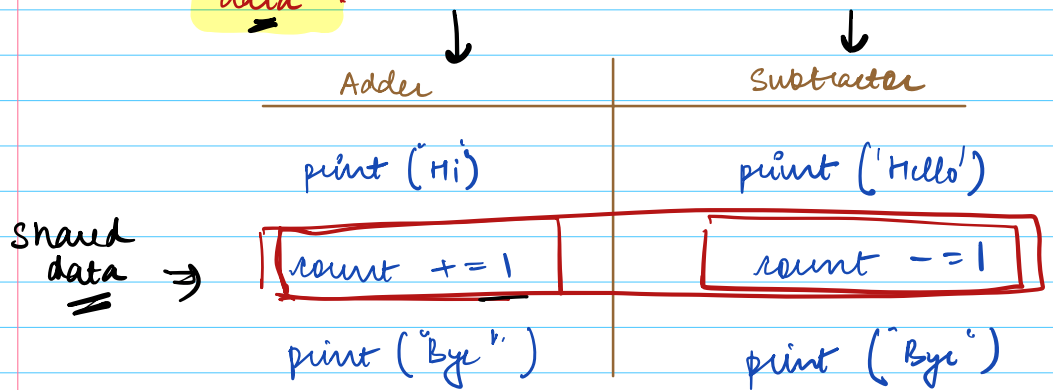
Problem of Synchronization

If multiple threads work simultaneously on shared data, we might get inconsistent results because of time sharing.

When will synchronisation problems happen? Or the conditions?

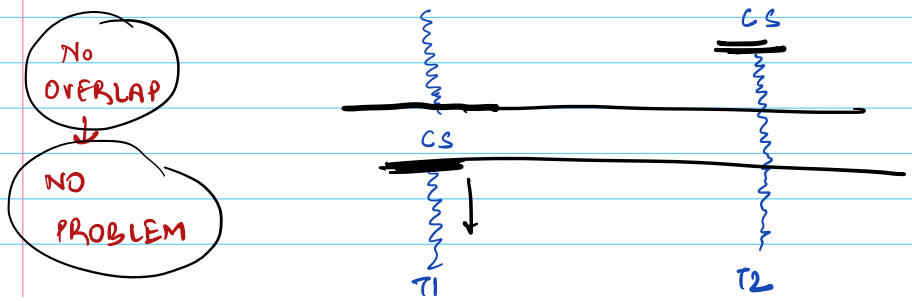
(1) Critical Section

part of code working on shared piece of data.



(2) Race condition

→ More than one thread trying to enter critical section at same time.



critical section is never paused between

preemptive scheduling algorithm

↓

OS can context switch before the thread completed its execution.

(3) Pre-emption :

Threads should be pre-empted while executing critical section

" context-switched "

SOLUTIONS

Theoretically what would you want to happen ?

(1) Mutual Exclusion :
Only 1 thread should execute critical section.

S2

(2) Progress -
Atleast 1 thread should be making progress
Entire app shouldn't stop.

(3) Bounded waiting : (starvation)
No thread should wait indefinitely.
to enter its critical area

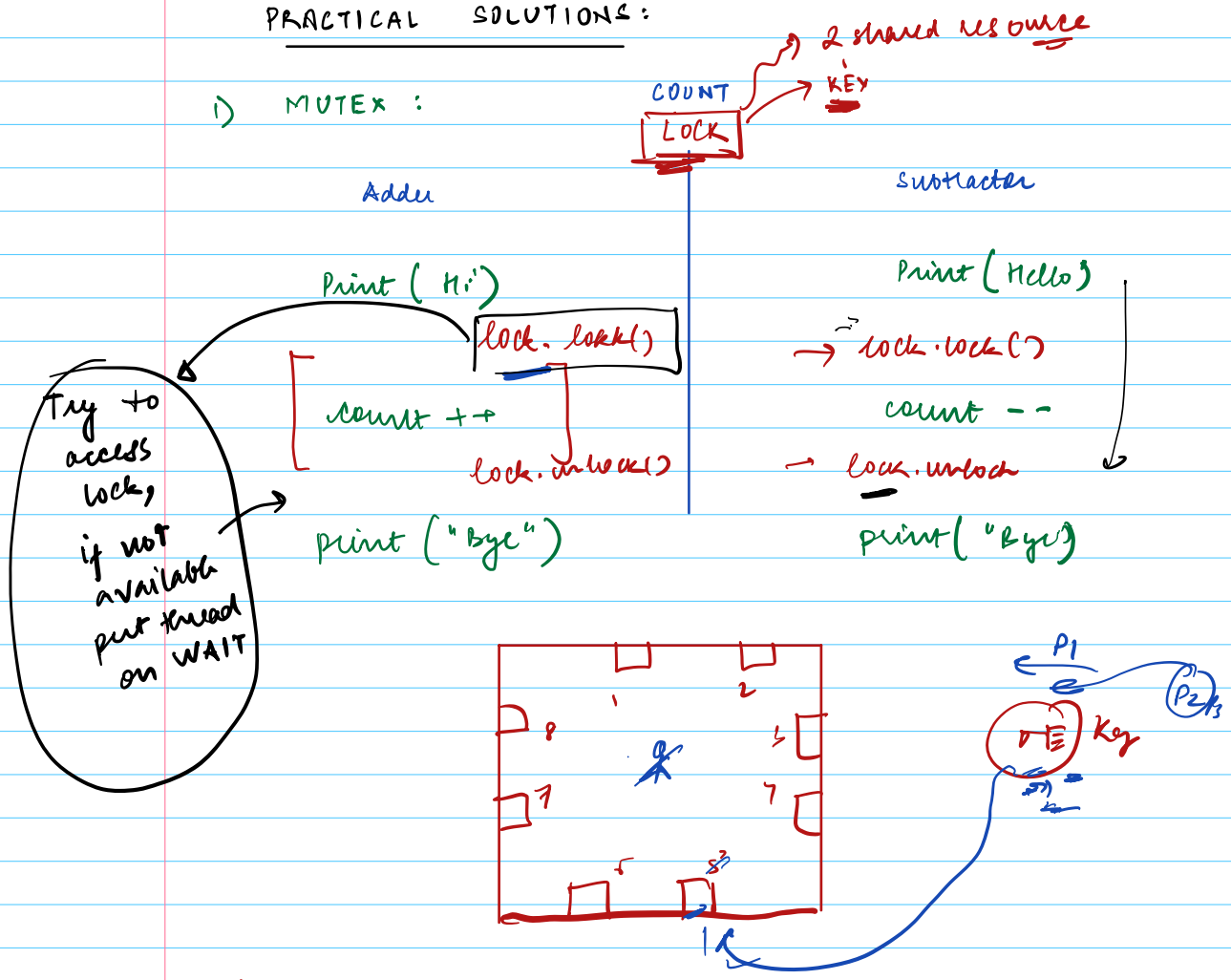
(4) No busy waiting

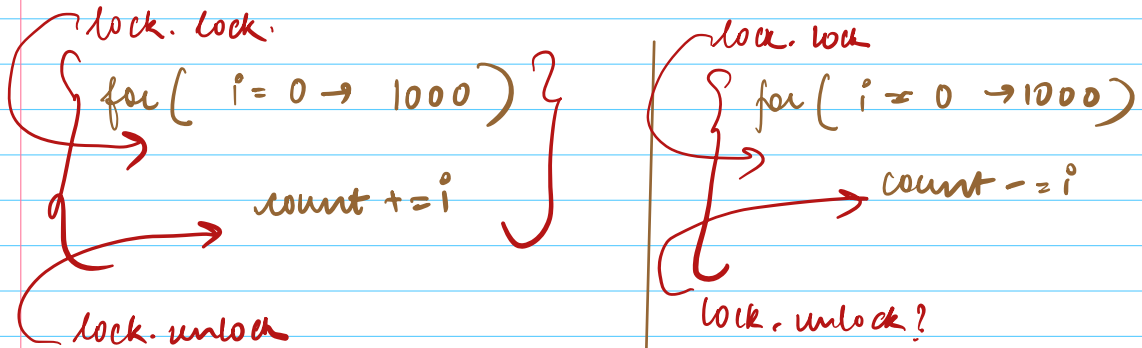
some kind of notification system.

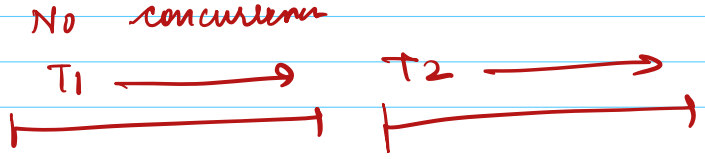
5 min → Mutex, Synchronized

PRACTICAL SOLUTIONS:

1) MUTEX :

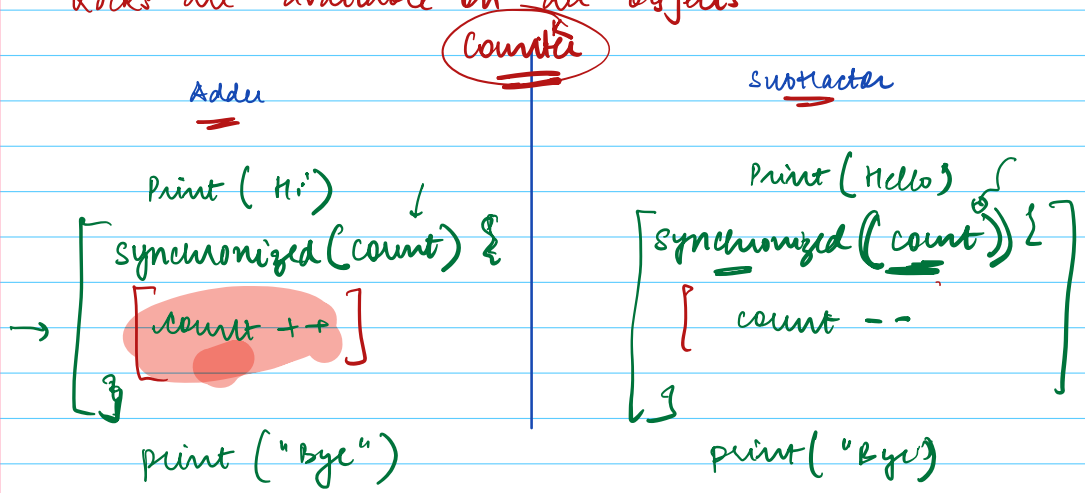






2) Synchronized keyword (Java)

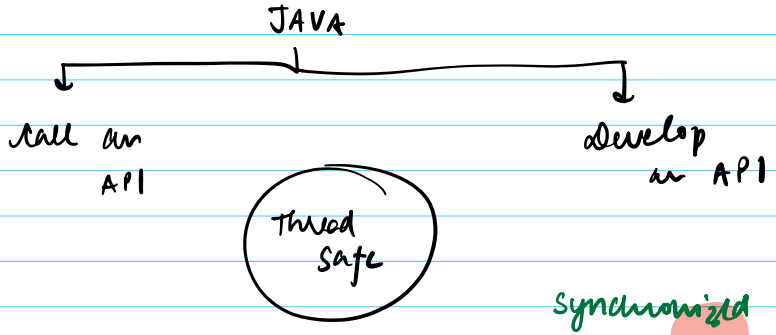
Locks are available on all objects



Mutex is an OS concept

Synchronized → Java specific

In java, locks are available on all objects



Counter

↓

not thread safe

Thread safe

↑

```
class Counter {

    private int count = 0

    public getCount ( ) {

        return count

    }
```

instance =

```
    Synchronized public void increment ( ) {

        count + = 1

    }
```

← only 1 thread will be able to access this method at a particular time

```
    Synchronized public void decrement ( ) {

        count -= 1

    }
```

① HashTable v/s HashMap.

② StringBuilder v/s StringBuffer.

Counter

static count = 0

sync static increment count() {

count++

}

decrement()

sync

count--

Adder()

increment()

class level

Subtracter

instance level