Concurrency -2

Agenda:

0) Additional Examples , join( ) Method ✓ ✓

1) Executors & Thread Pools

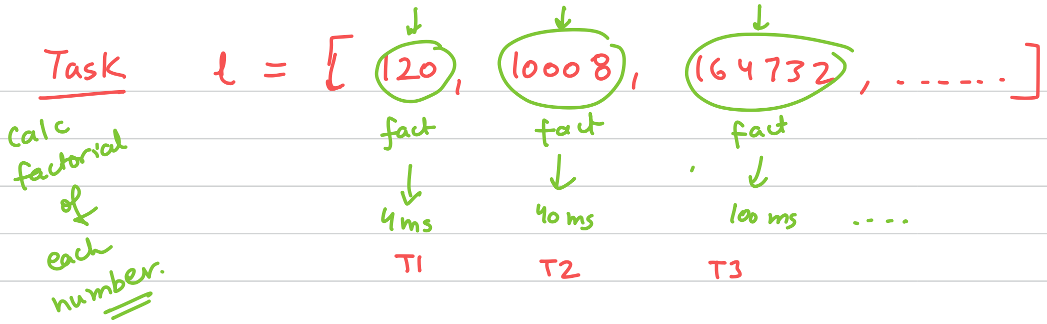
2) Callables
   - ↳ Return data from a thread
   - ↳ Merge Sort ⟹ Cached Thread Pool.

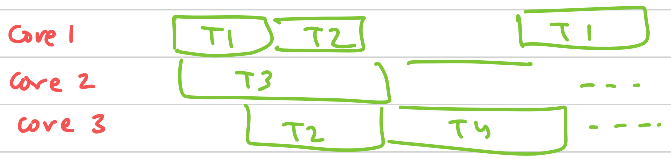
3) Intro to Synchronisation

Quick Recap on Threads!

Q) Print numbers from 1 to 100.

Task: To print a number
Multi-threaded program:
- 100 threads
- Each thread prints one number.


Task
l = [120, 10008, 164732, ...]
fact
fact
fact
4ms
40 ms
100 ms
....
T1
T2
T3
calc factorial of each number.



① implementing "Runnable" interface

② extend Thread, overide RUN() method.

Main

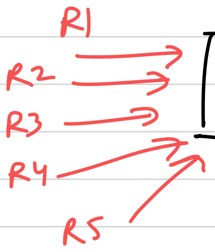
When we invoke the join() method on a thread, the calling thread goes into a waiting state. It remains in a waiting state until the referenced thread terminates.

```
Main(){

  ⇒ t.join()      Main goes into waiting state
                  until t terminates.

}
```

```
main(){

    t.join(2000)  → 2s is the MAX (at max) time Main
                    will wait.
    ≡
}
```

Executors

Motivation:



1000 req.

1 M

↓

1 M thread.

$\boxed{\text{16 Core CPU}}$

10 Threads

↑

⇒ Creating & Deleting Threads take time.

⇒ Context Switching, Thrashing.

⇒ wastage of Memory & Resources

Client

↓

what tasks to execute

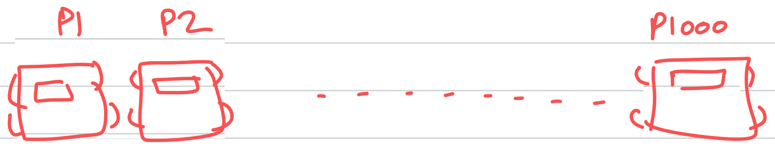
Executor

↓

Best about how to run tasks efficient in a system.

Executor Framework

1000 cars

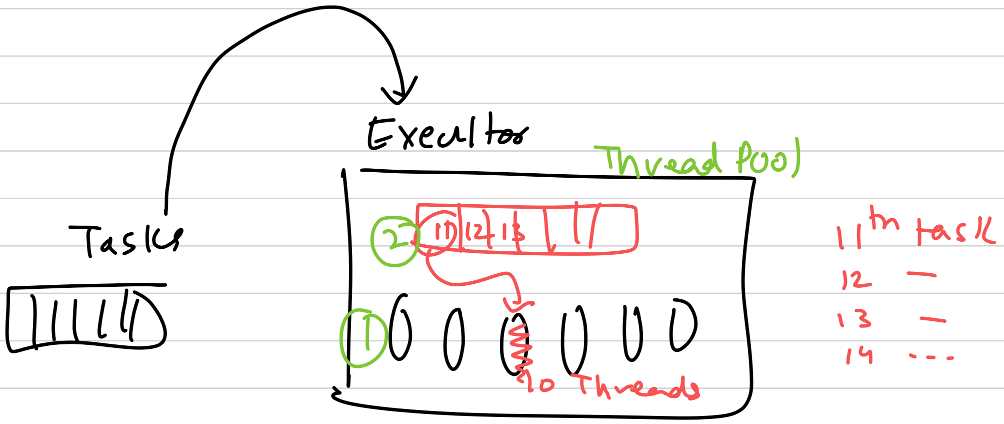


P1 P2 . . . . . P10

Reuse

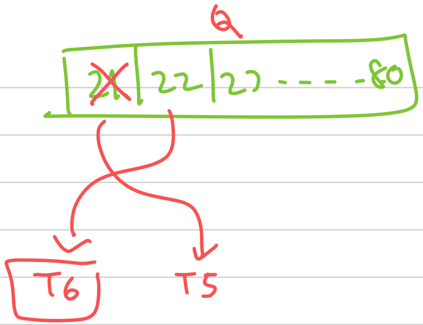
Java : Executor Service which is an interface, Java Provides several implementation.
Simplify Thread Management.

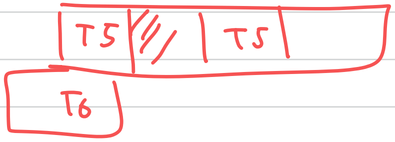
- Uses Thread Pools, and reduces the cost of creating new thread threads
- Efficient Scaling by utilizing multiple processor cores
- Built in synchronisation mechanisms, reducing concurrency related issues.



A thread pool consists of 3 components:
1) Worker Threads: are available in the pool to execute tasks. They are pre-created, and kept alive throughout the lifetime of application.

2) Submitted Tasks are placed in FIFO queue. Threads pop tasks from the queue and execute them so they are executed in the order they are submitted.





Thread Pool Manager: allocates tasks to threads and ensures proper thread synchronisation.

Java has 5 variations of Thread Pool

1) **FixedThreadPool :** fixed no of threads
2) **CachedThreadPool:** creates new threads as needed, so it is variable sized pool.
Can problems if large number of threads are created.

3) ScheduledThreadPool,
4) WorkStealingPool
5) ForkJoinPool (LargeTask -> subtasks)

Callables: Like Runnables, Callable are a way to define a task but unlike Runnable callable Can return some data back to the client.

1) Identify the task that you want to run in a different thread. Create a class for that task.

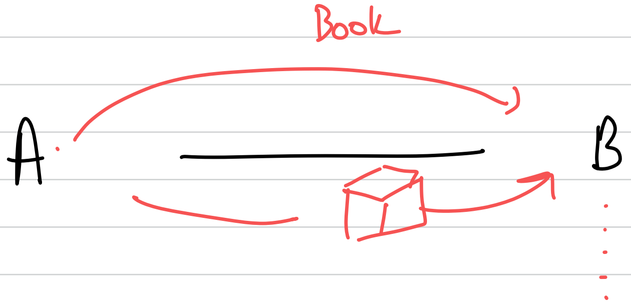
```
class Sorter implements Callable<T> {
    ≡
    T call() {
        ≡
    }
}
```

```
void run()
```

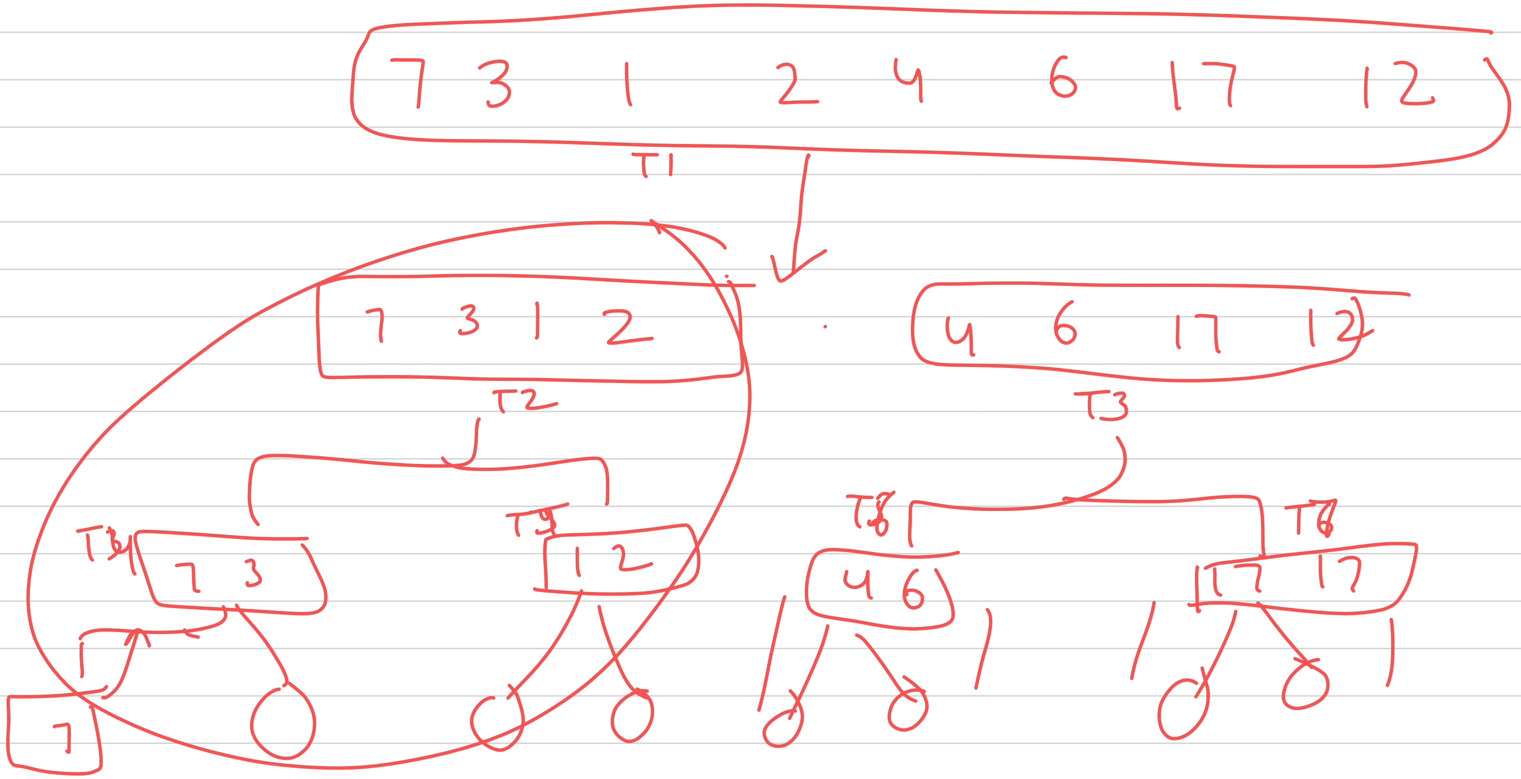
2) Identify the return type of the data. T

3) Implement 'call()' method in the class

4) Implement the logic inside 'call()' method.


Book
A
B

7,3,1,2,4,6,17,12)


7 3 1 2 4 6 17 12
T1
7 3 1 2
T2
4 6 17 12
T3
T4
7 3
T5
1 2
T6
4 6
T7
17 17
7