Agenda : (1) Write a multi-threaded Hello World

(2) Print nos. 1 to 100, each using a separate thread

(3) Executors & Thread pool
↳ print 1-100

(4) Callables ⇒ How threads can return data
↳ (a) Merge Sort multi-threaded
(b) Futures

Class Starts at 9:05

(1) Writing a Hello world in a multi-threaded appn.

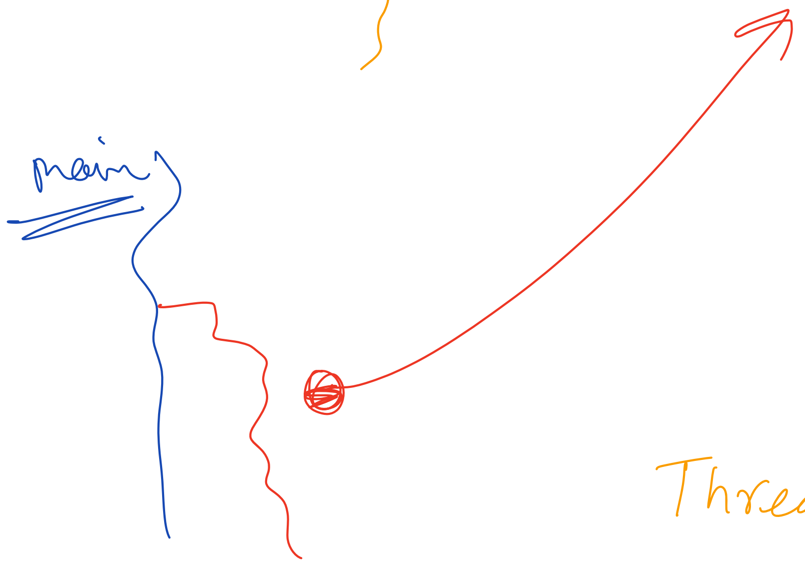
"Hello World" ⇒ using a separate thread

↳ guideline ⇒ HelloWorldPrinter
(noun)

Step 2 ⇒ class HelloWorldPrinter implements Runnable {
@overide

```
void run() {
    SOP("Hello World")
}
}
```

→ write code that needs to be executed in a ~~st~~ separate thread

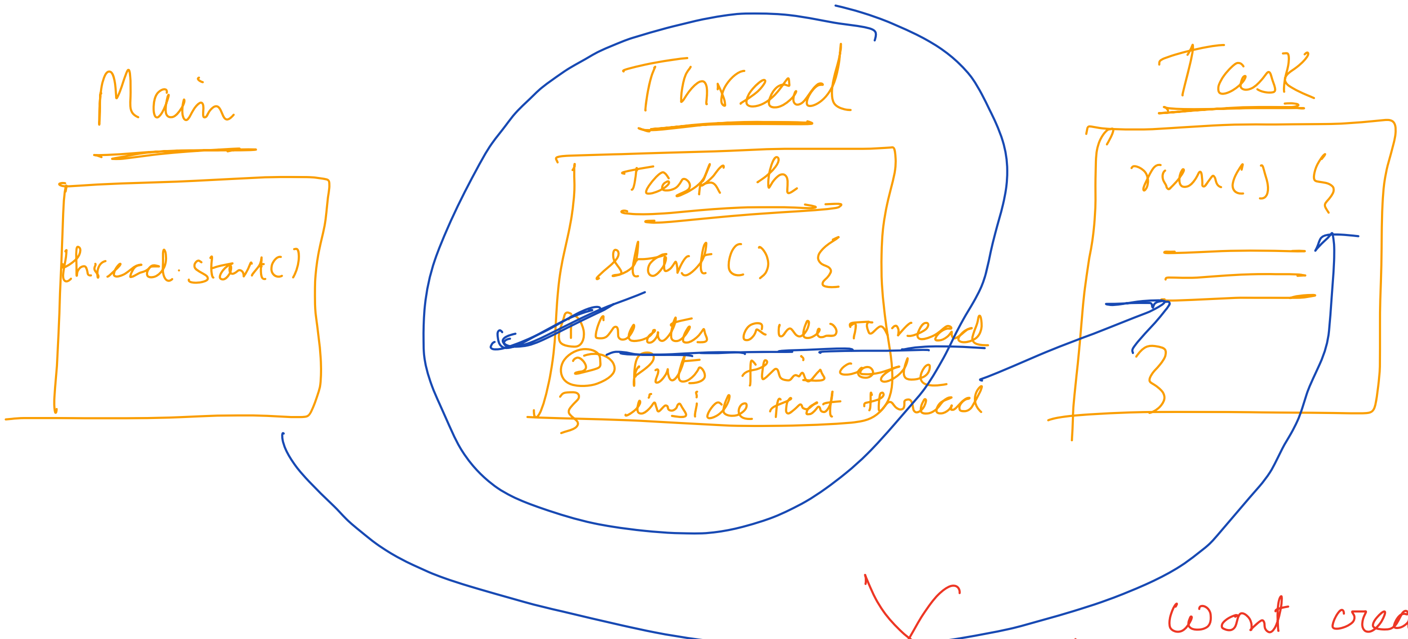


```
Thread thread = new Thread( );
```

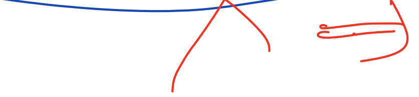
Runnable task

```
thread.start();
```

```
HelloWorldPrinter h = new HelloWorldPrinter();
h.run();
```

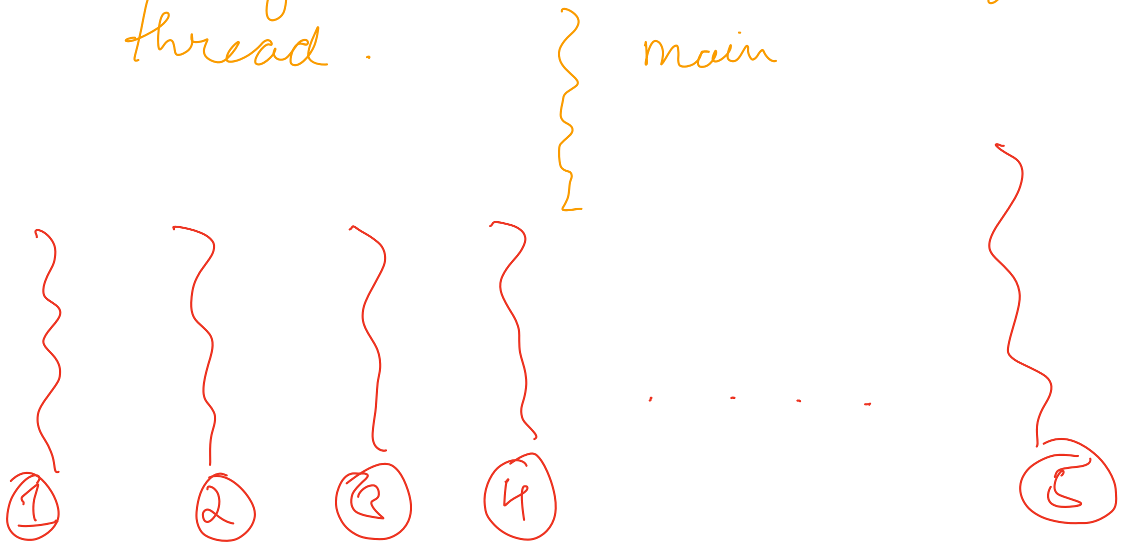


wont create



⟹ a new thread

② Printing 1-100, each using a separate thread.

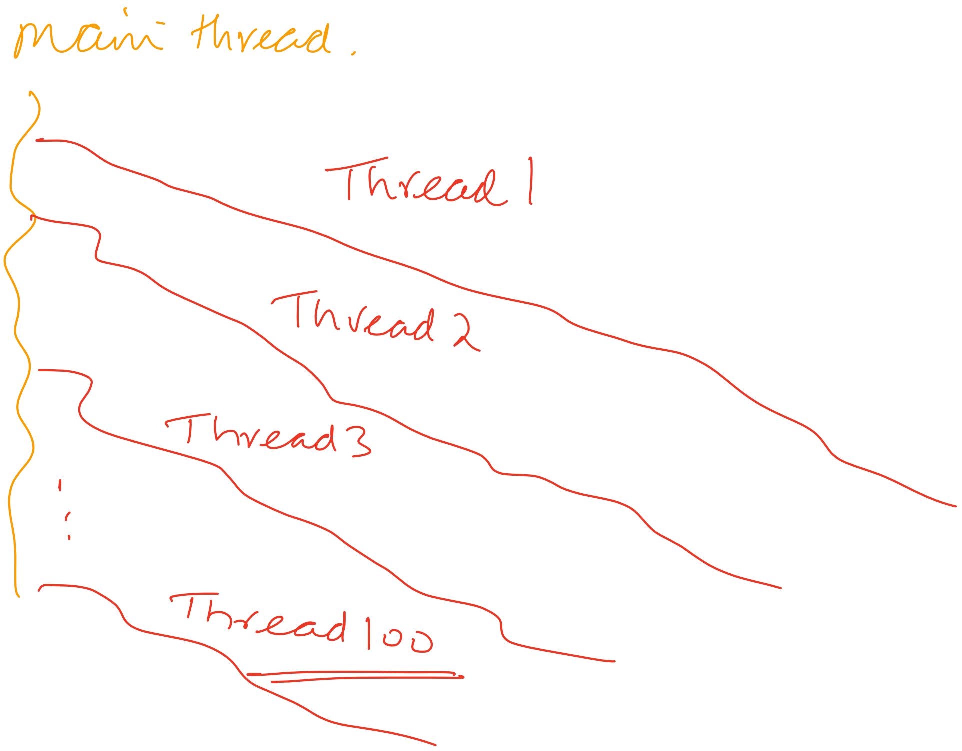


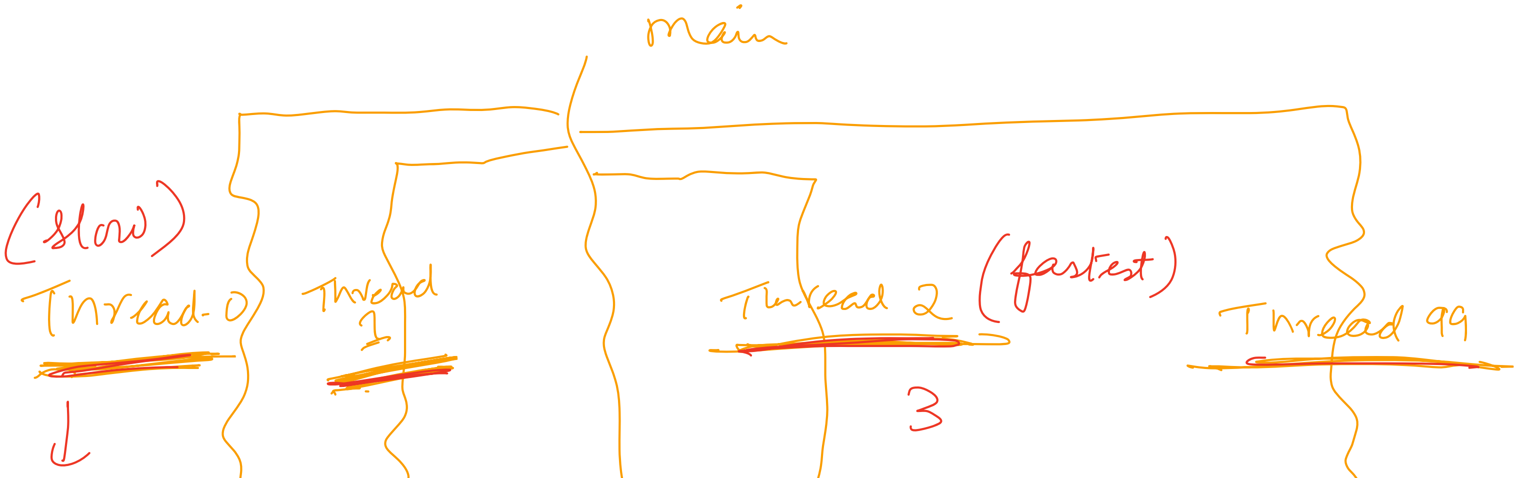
Step 1 ⟹ Task is printing a number.

class NumberPrinter

```
class NumberPrinter implements Runnable {
        int numberToPrint;
      public NumberPrinter(int i)
        numberToPrint = i;
    @Override
    void run() {
                             ⟹ print a number
    SOP(numberToPrint);

}}
```





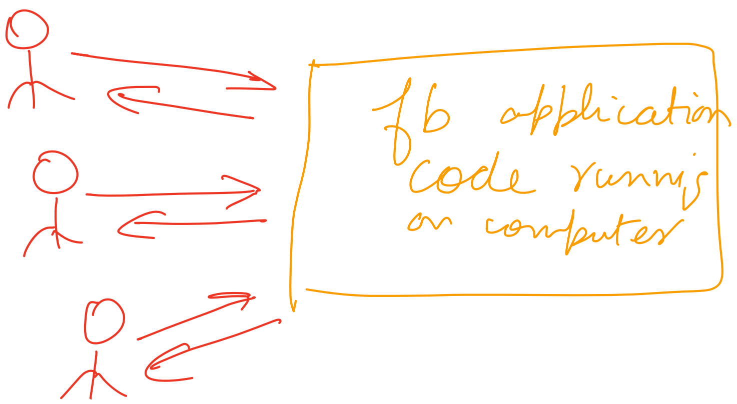
If thread-0 is slowest

↳ it could happen that 1 is the last no. to get printed

⇒ Printed nos. from 1-100

↳ printed using 100 threads

eg :

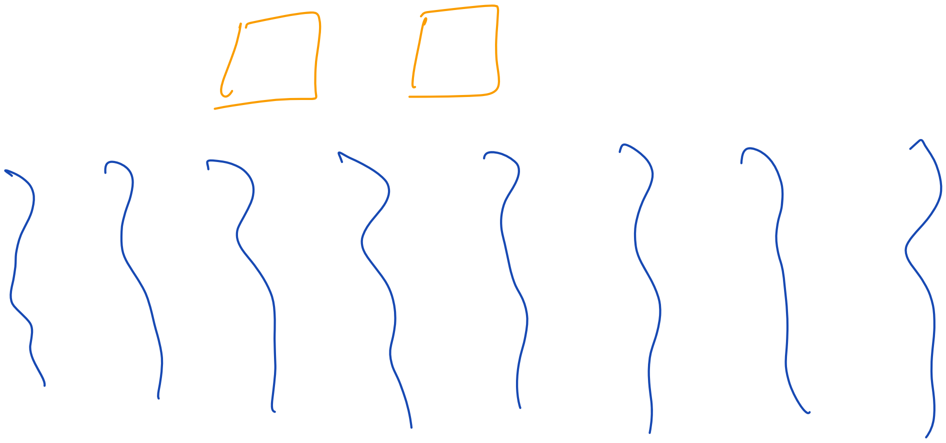


⇒ all of these requests are nothing but tasks, If I start creating a new thread for every task ⇒ 100000's of requests

↳ 10000s of threads

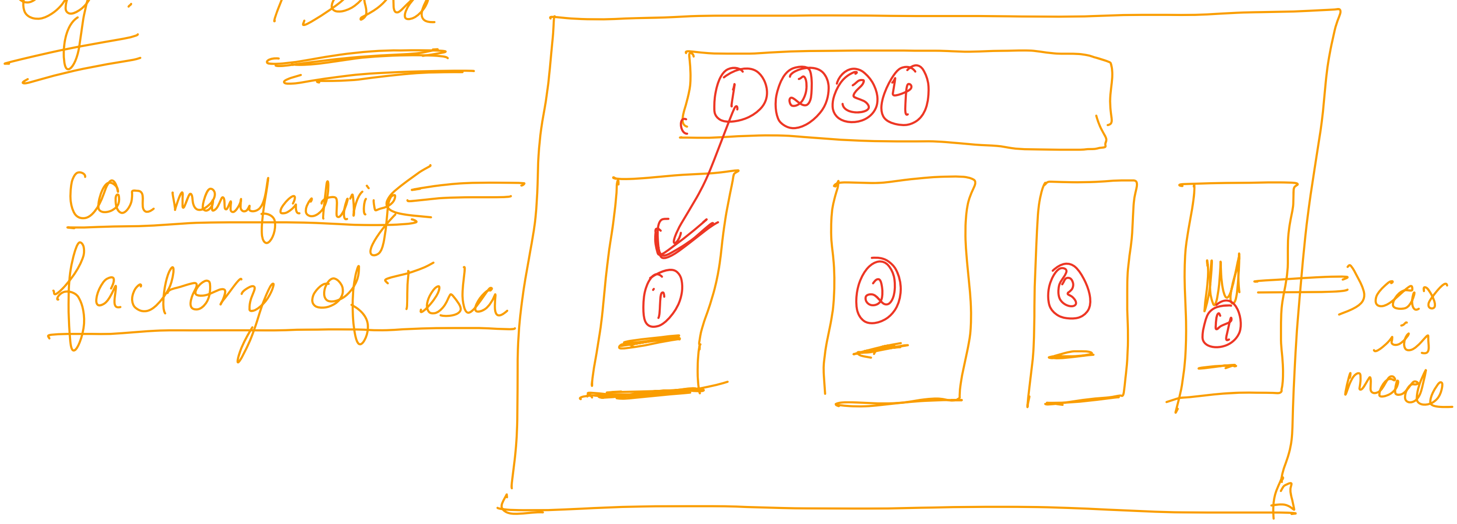
Ques What is the problem with this

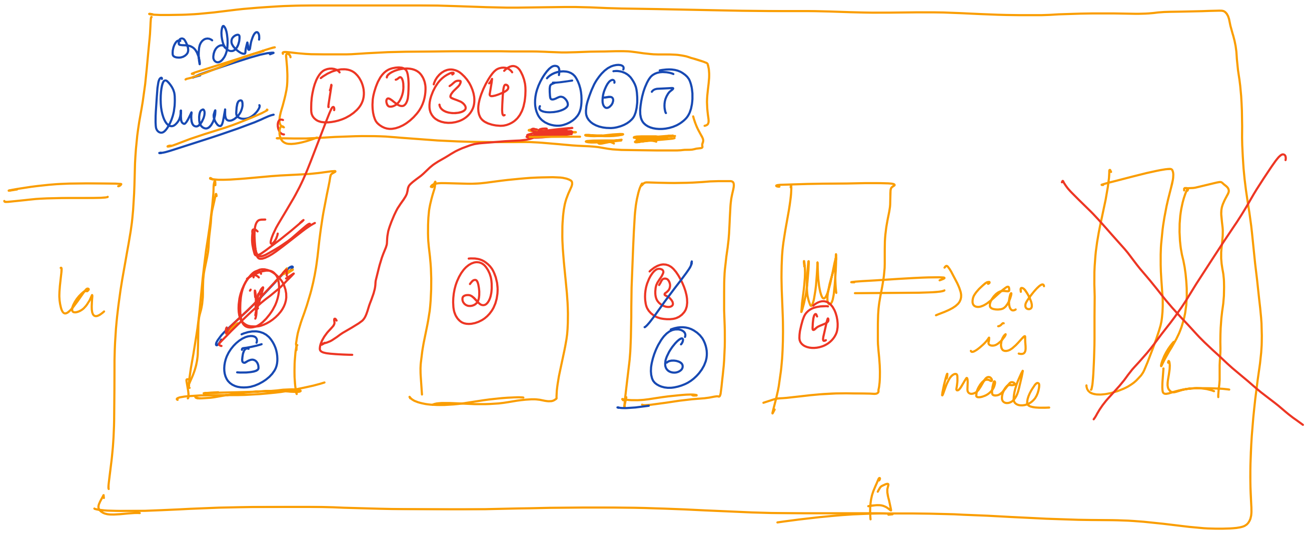
Ques Max no. of threads that can run in parallel at the same time ⇒ No. of cores



↳ a lot of content switching will happen for above case.

eg: Tesla





This exact concept in S/W ⇒ Thread Pools.
Computers

Thread Pools

Earlier what we have done is

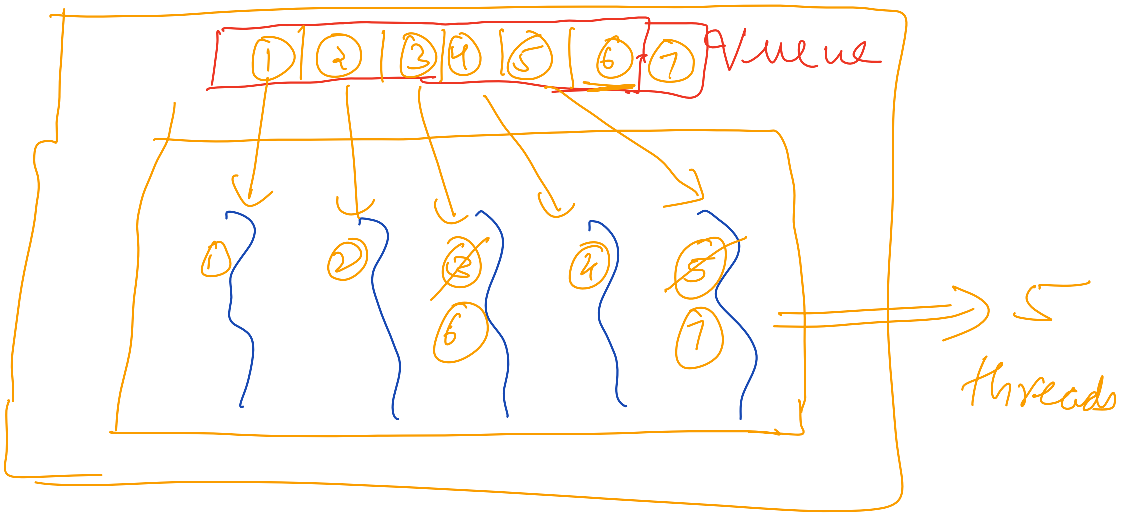
1. Created a Task
2. Create a thread and assign Task to this thread
3. Start a thread

Thread Pool says:

① Create a Task

② Handover the task to me & I'll take care of it.

Thread Pool

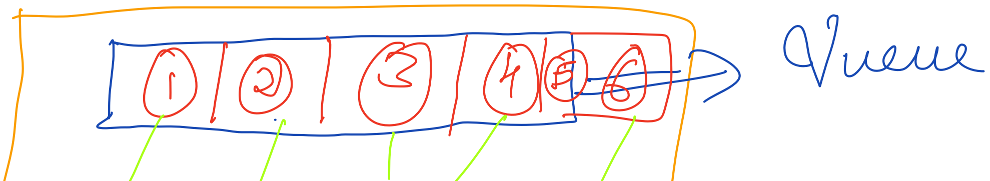


Q ⇒ Have we solved the problem of Context switching ⇒ Yes, basically by controlling the no. of threads.

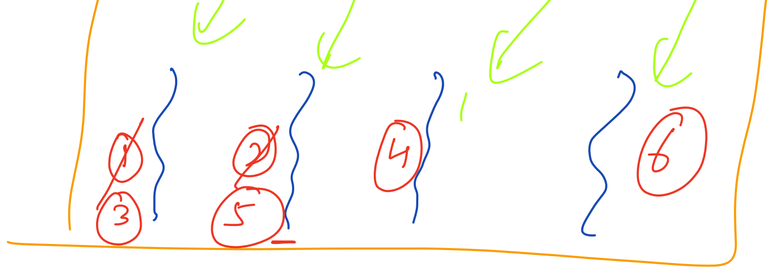
Class Starts at 10:35

Cached Thread Pool ⇒ Reuse the existing threads if possible, otherwise create a new thread, but dont put any task in the waiting queue.

Cached Thread



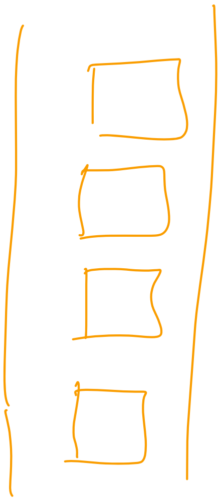
Thread Pool



Ques What is the number of threads that we should create?

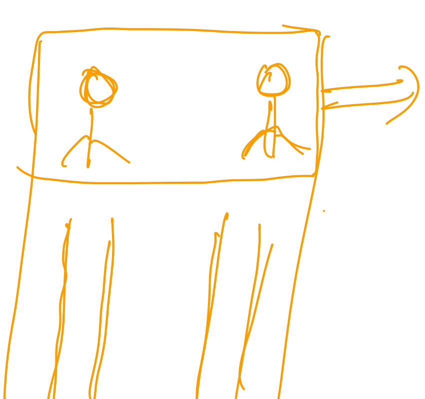
⇒ Purpose of multi-threading ⇒ parallel execution of tasks to reduce time

⇒ no. of tasks that can be executed parallelly at same time ⇒ no. of cores



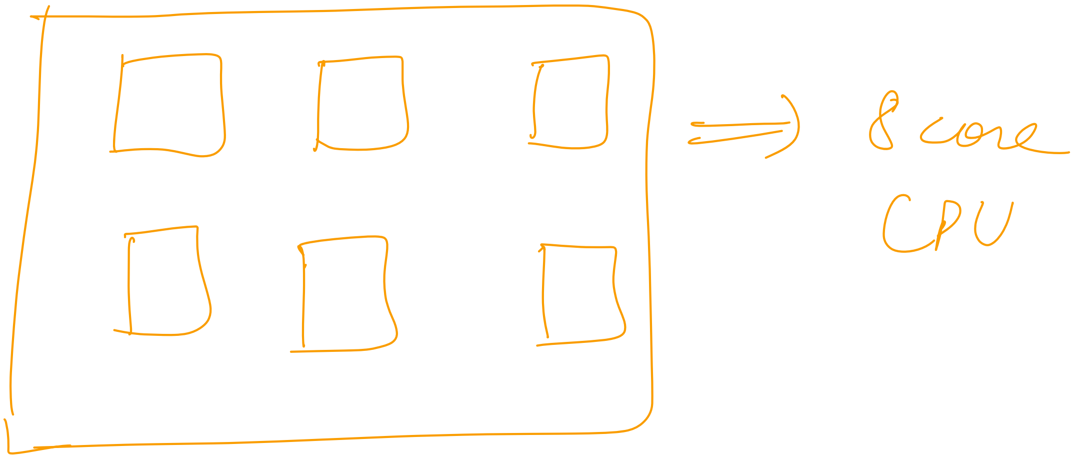
no. of threads that I should create ~~=~~ no. of cores

① Hyper-threading ⇒ allows a single core to execute 2 threads at the same time

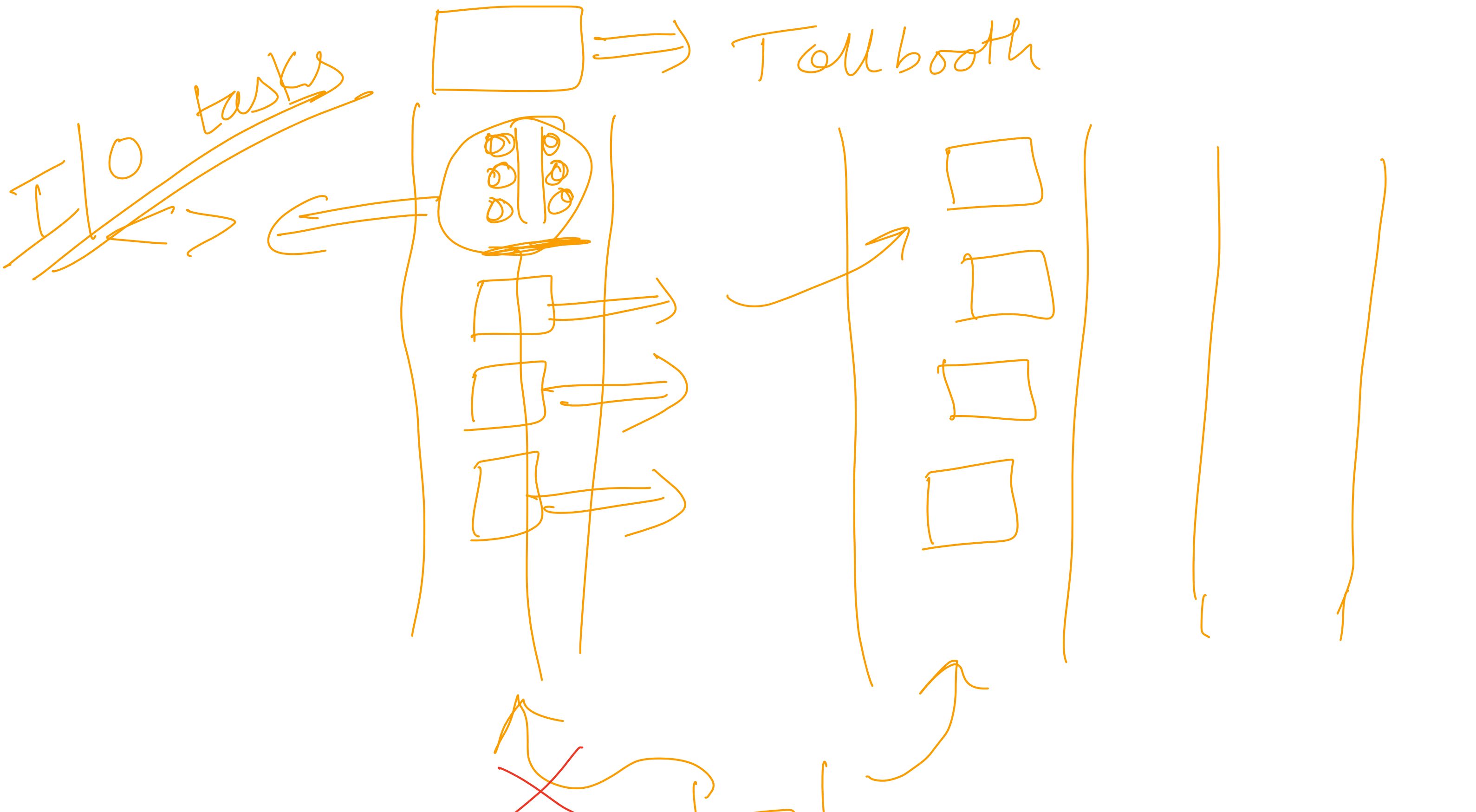


core is executing 2 threads at the same time no. of threads $\cancel{=}$ 2 * no. of cores

②



⇒ we create a multi-threaded application in a multi-cores system so that all cores are used ⇒ to have more cpu utilization.



⇒ There are certain tasks that take more time than others ⇒ I/O Tasks

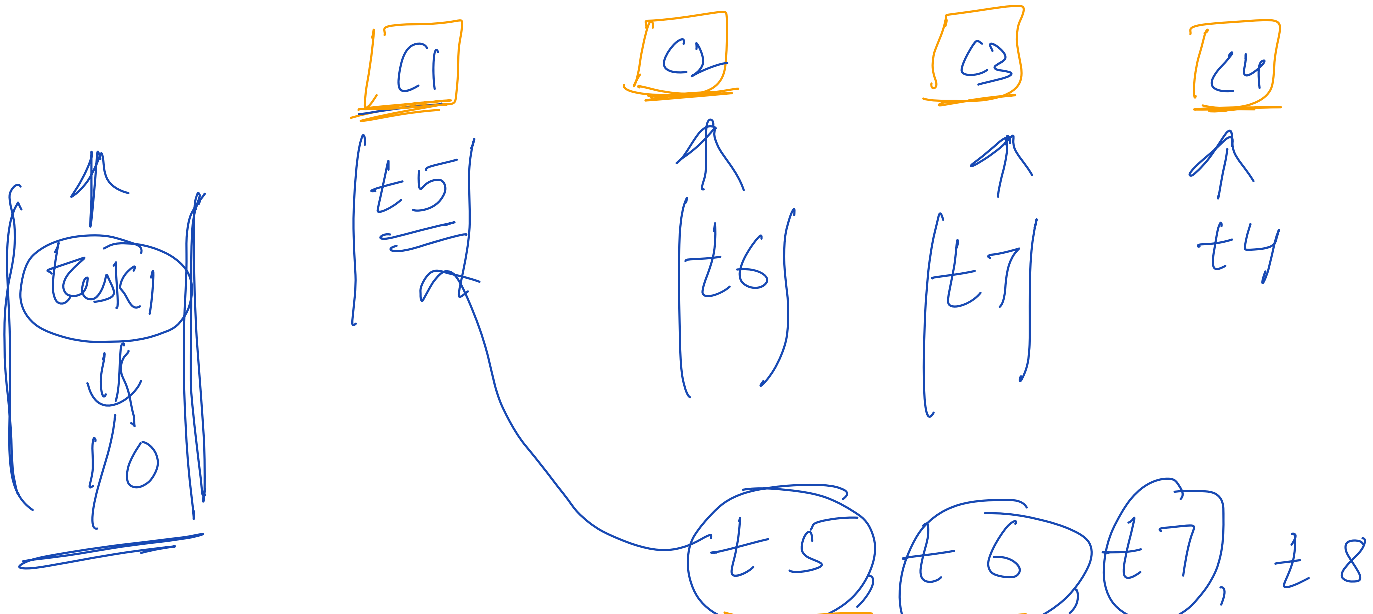
I/O ⇒ Input Output

① Printing to a command line

② Read/write from a disk

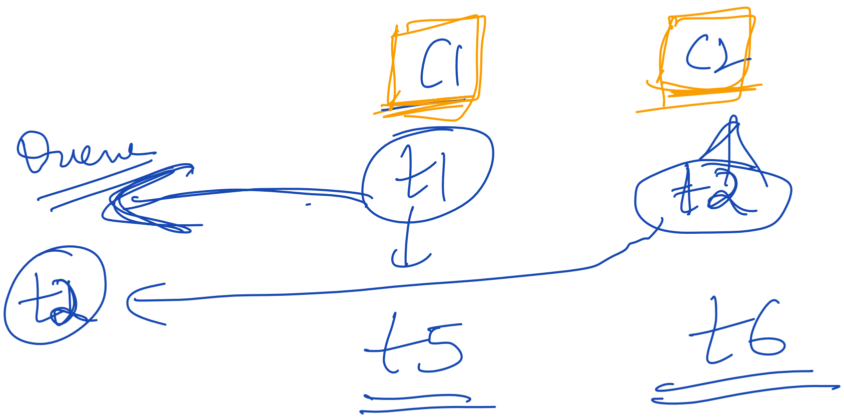
③ Network call

If tasks are I/O intensive, does it makes sense to have more threads



|t2| |t3|

t9, t10, t11



t5, t6, t7, t8, t9

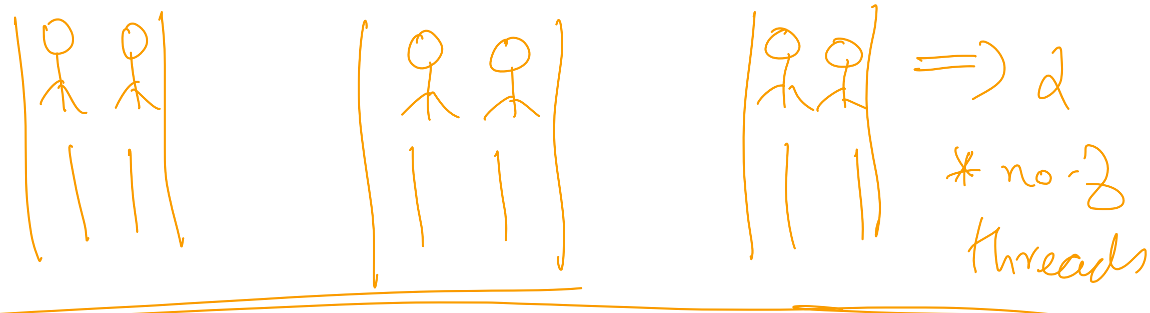
If tasks are I/O intensive, number of threads = 2* no. of cores

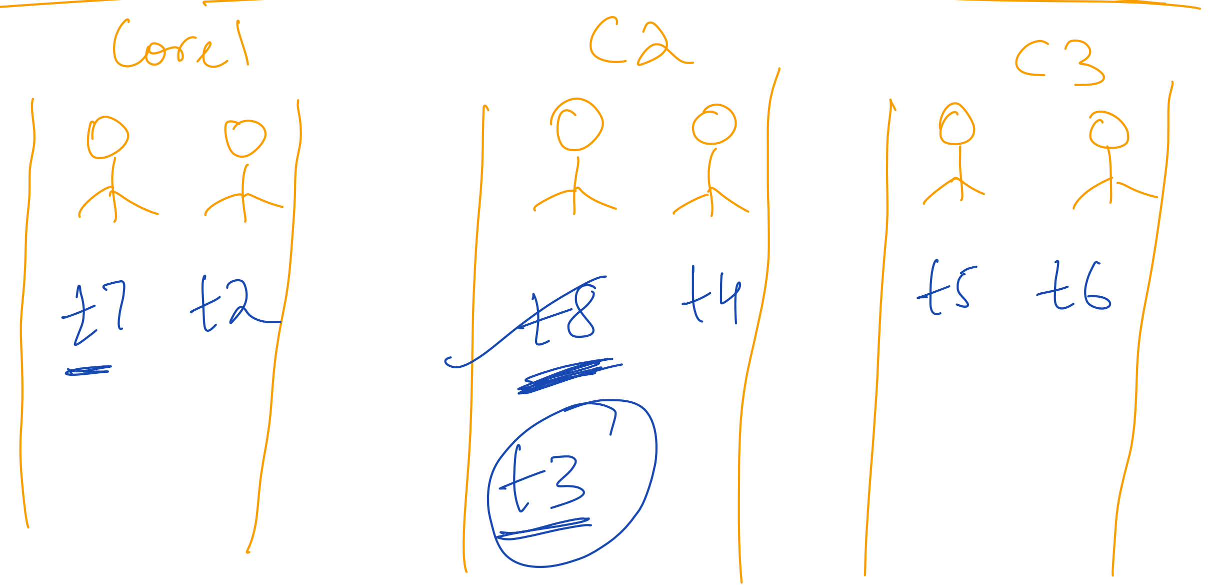
✓ > 2* no. of cores

< 2* no. of cores

If tasks are CPU intensive

no. of threads ✓ = 2* no. of cores





t1
⇓
needs to do some I/o work

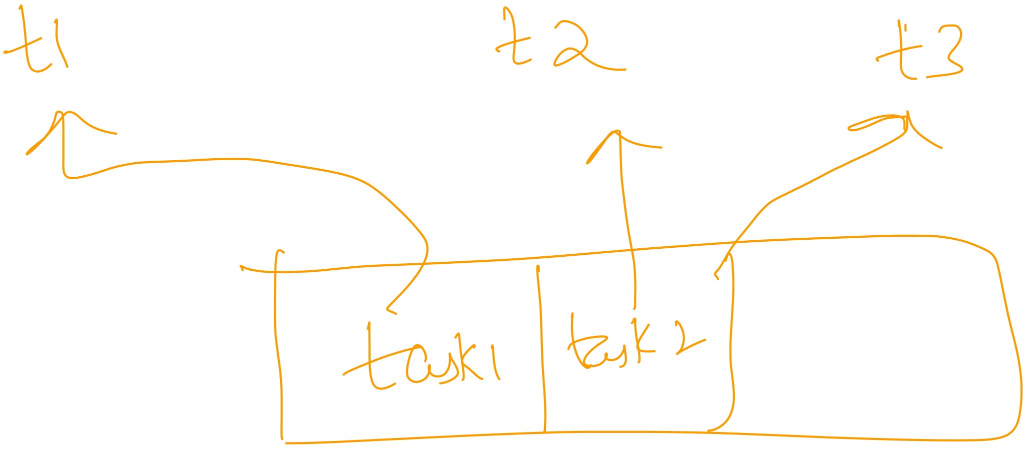
buses

no. of threads in I/O intensive > 2 * no. of cores

```
void run() {
    print( )
    int a=10;    }  ⟶  Core
    int b=20;
    ~~reading~~ data from a disk
    int c=a+b;   ⟶  again come back to core
}
```

Agenda : ~~①~~ Callables
② Adder-Subtractor

Synchronization


t1
t2
t3
task1
task2