Agenda

1. Executors and thread pools
   (this is how threads are coded in production systems)

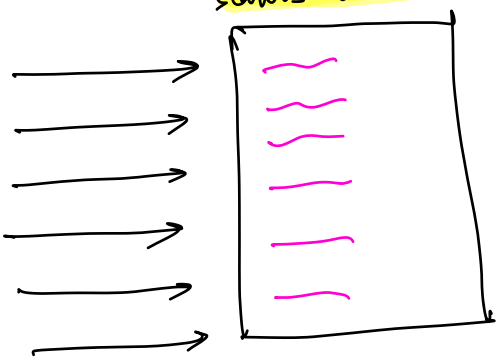
2. Callables
   (How threads can return data)
   → Implement Merge sort (Multithreaded environment)

Executors and Thread Pools

Client
- → Create a task
- → Decides when the task will run (multithreaded environment)

1 million threads

Scaler's Server Box



- → A lot of memory usage
- → Context Switching

Client

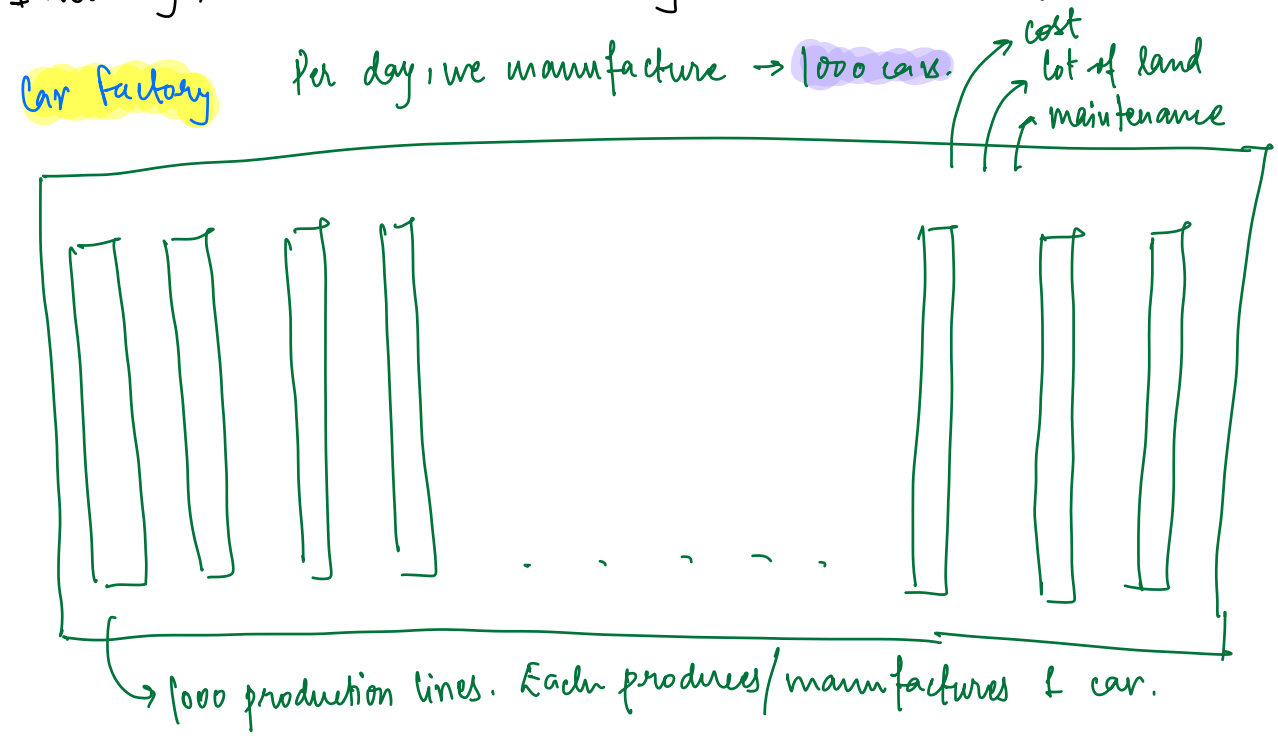
Knows best about what should run independently (Task)

Executor

Knows best about how to run task efficiently in a system.

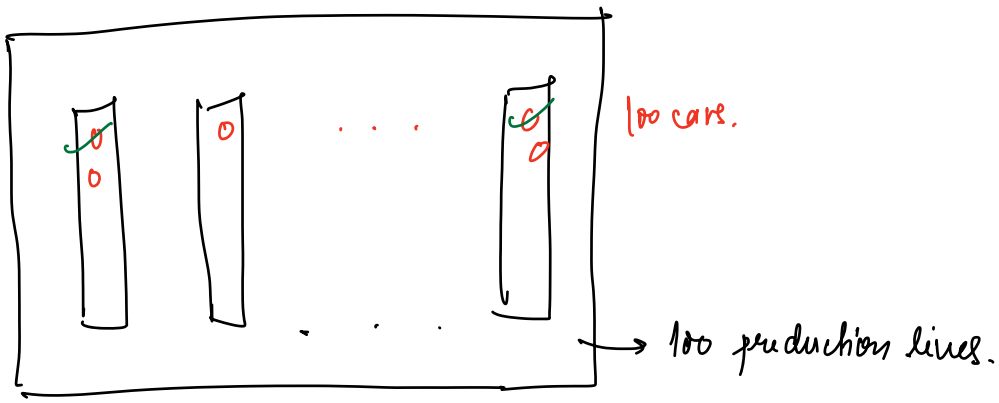
- → Division of responsibility
- → Efficiency in running the application.

Internally, executors use something called thread pools.



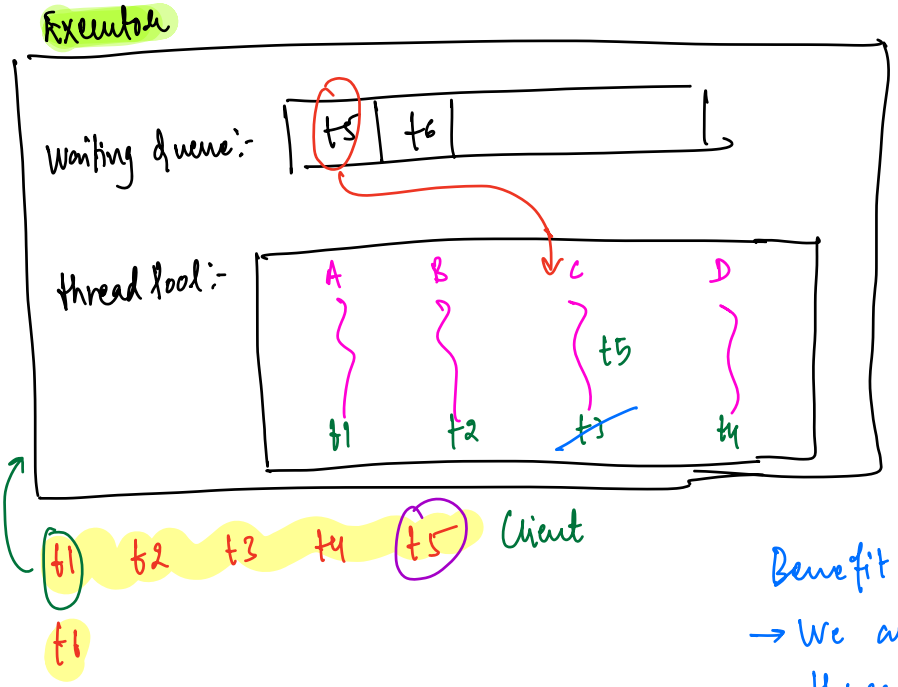
→ Not efficient

→ We are not reusing the production lines for future cases.



Thread Pools.

↳ Set of threads acting as prodn lines



t1 t2 t3 t4 t5 Client

t1

Benefit :-

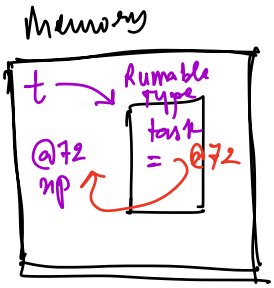
→ We are reusing the threads.

NumberPrinter np = new NumberPrinter(1)

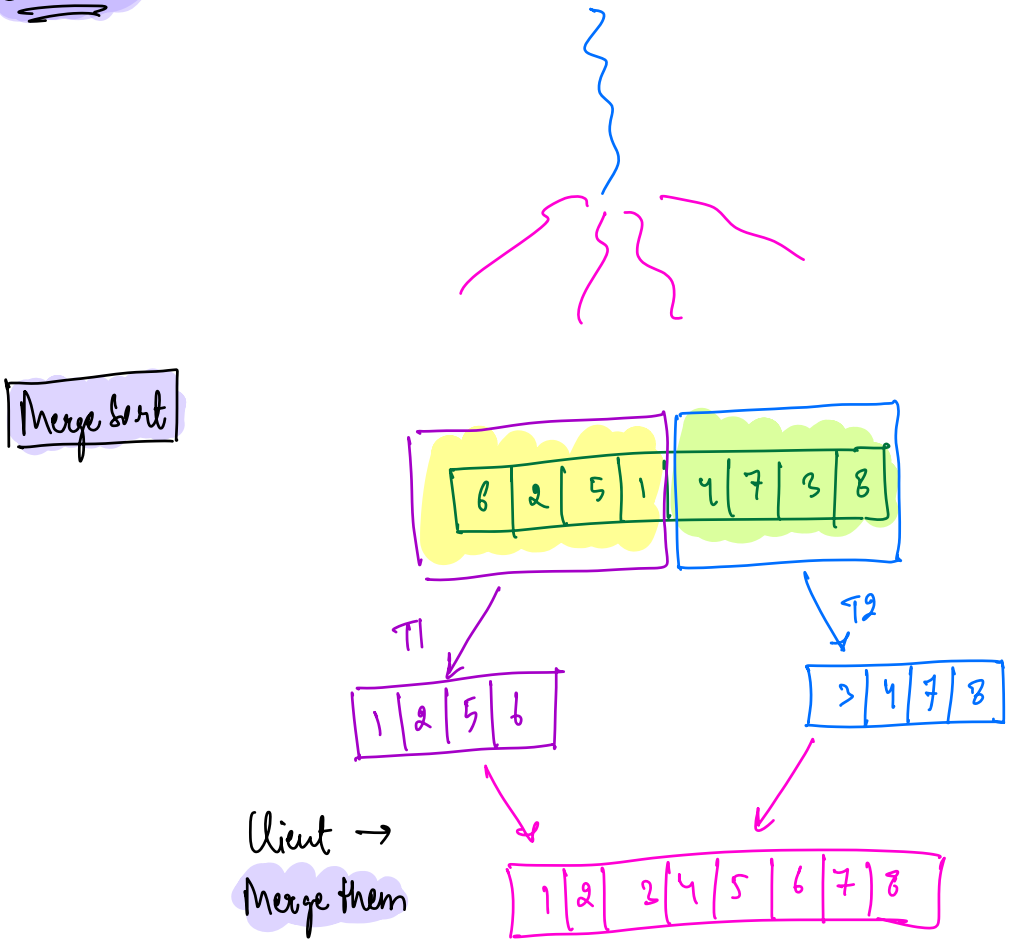
Runnable Type

Thread t = new Thread (NumberPrinter Object)

Student st = new Student(name, age, uni).



Callables



What are Callables?

→ Like Runnables, Callables are a way to define a task.

→ Unlike Runnable, Callables return something back to the Client.

Steps:-

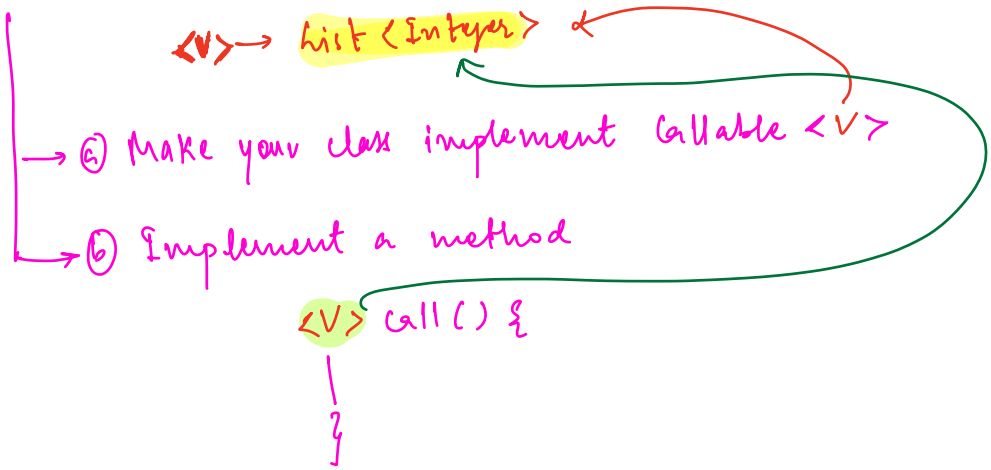
① Identify the task to be run in parallel thread.
Create a class for that task.
Name of Runnable / Callable → Noun

MergeSorter

class MergeSorter { }

H/W → Read about Generics in Java

② Identify the return type of data that the task will return



class MergeSorter implements Callable<List<Integer>> {

List<Integer> call(){

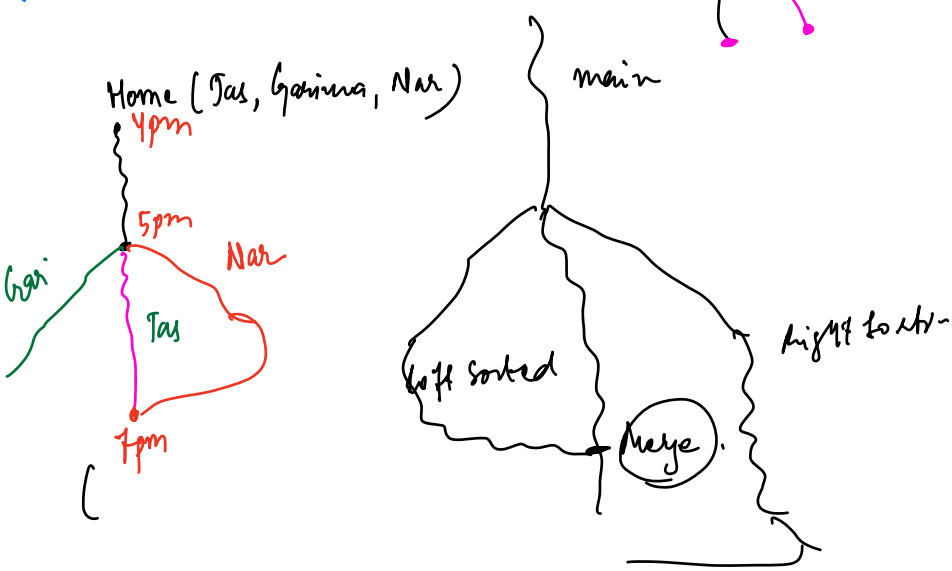
} logic of merge sort is here.

}.

futures

```
print (a)
executor. execute ( task )
print(b)
```


```
print(a)
Integer i = executor. submit ( callable task - - - )
print(b)
print(i)
```




```
print(a)
Future<Integer> f = executor. submit ( callable task - - - )
print(b)
print(i)
```

→ this future object is like a bucket (assurance to the caller) that I will put an Integer in the bucket later.

e.g.