THREADS - 2

1. Code Threads ⇐ extend Thread / implement Runnables
2. Executors ⇒ Executor Frameworks
3. Thread Pools ⇐ How real web servers work
4. Callables ⇐ and Future
   - ⇐ Implement Multi Threaded Merge Sort
5. Intro to Synchronization Problems

Next Class : Solving Sync
- → Locks
- → Sync keyword
- → Atomic Data Types

```
Main() {
    doSomething() {
        sout("Hi")
    }
    psvm() {
        Sout("Hello World");
        doSomethingElse()
    }
}
```

→ Print (Hello World) ← from a separate Thread

⇒ Task ⇒ Print Hello World

SOP:

(1) Create a class for that task

→ class Print Hello World {

}

(2) Implement Runnable interface

OR

extend Thread class

↓

class Print Hello World implements Runnable {

}

extends Thread {

}

③ Create a run() method in the class

```
class Print Hello World implements Runnable {
    void run() {
        print(Hello World);
    }
}
```

④ At the place where you want to start a new thread
a.) creat an object of above class
b.) create an object of Thread class and pass a to it

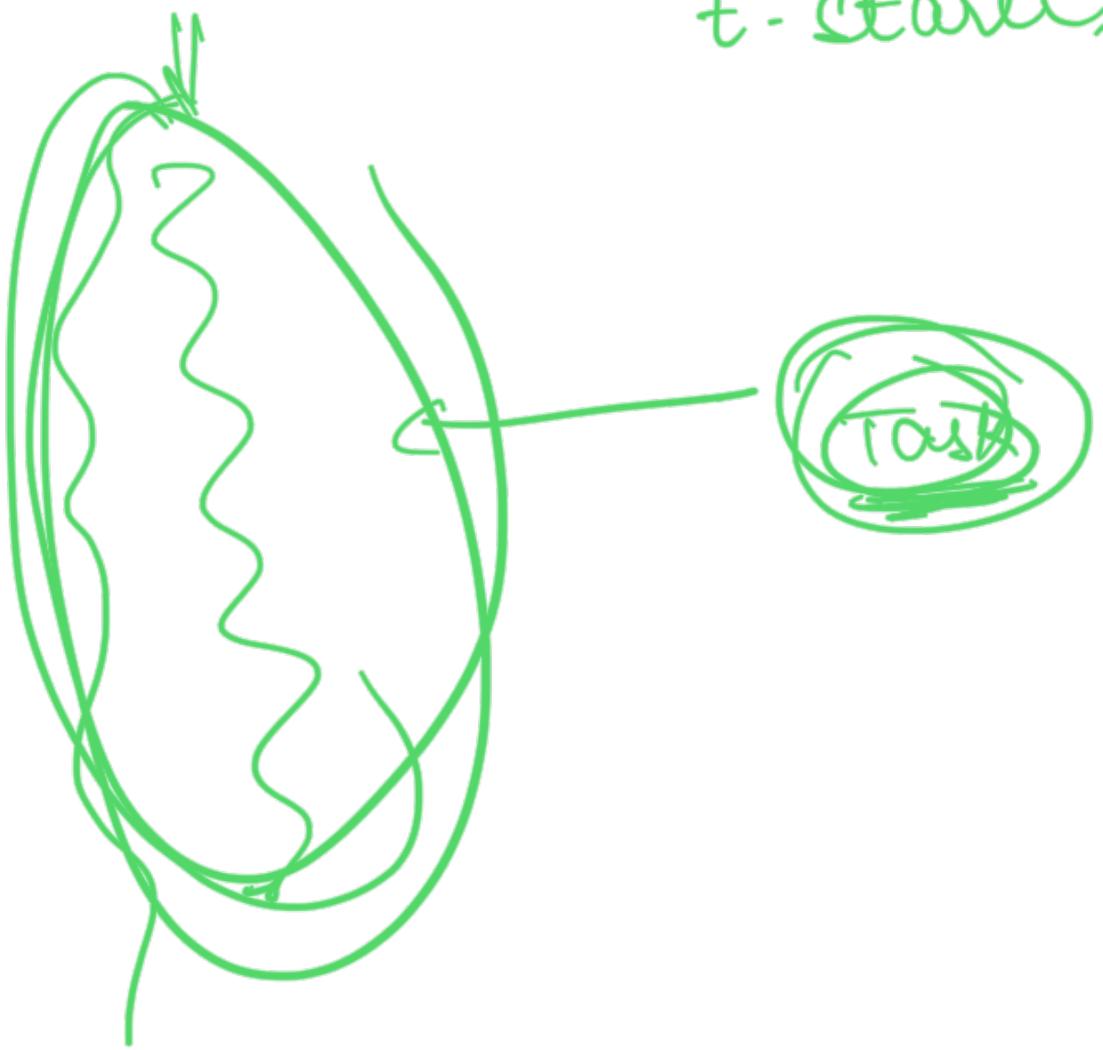
```
Print Hello World obj = new Print Hello World();
Thread t = new Thread(obj);
t.start();
```

Print Hello

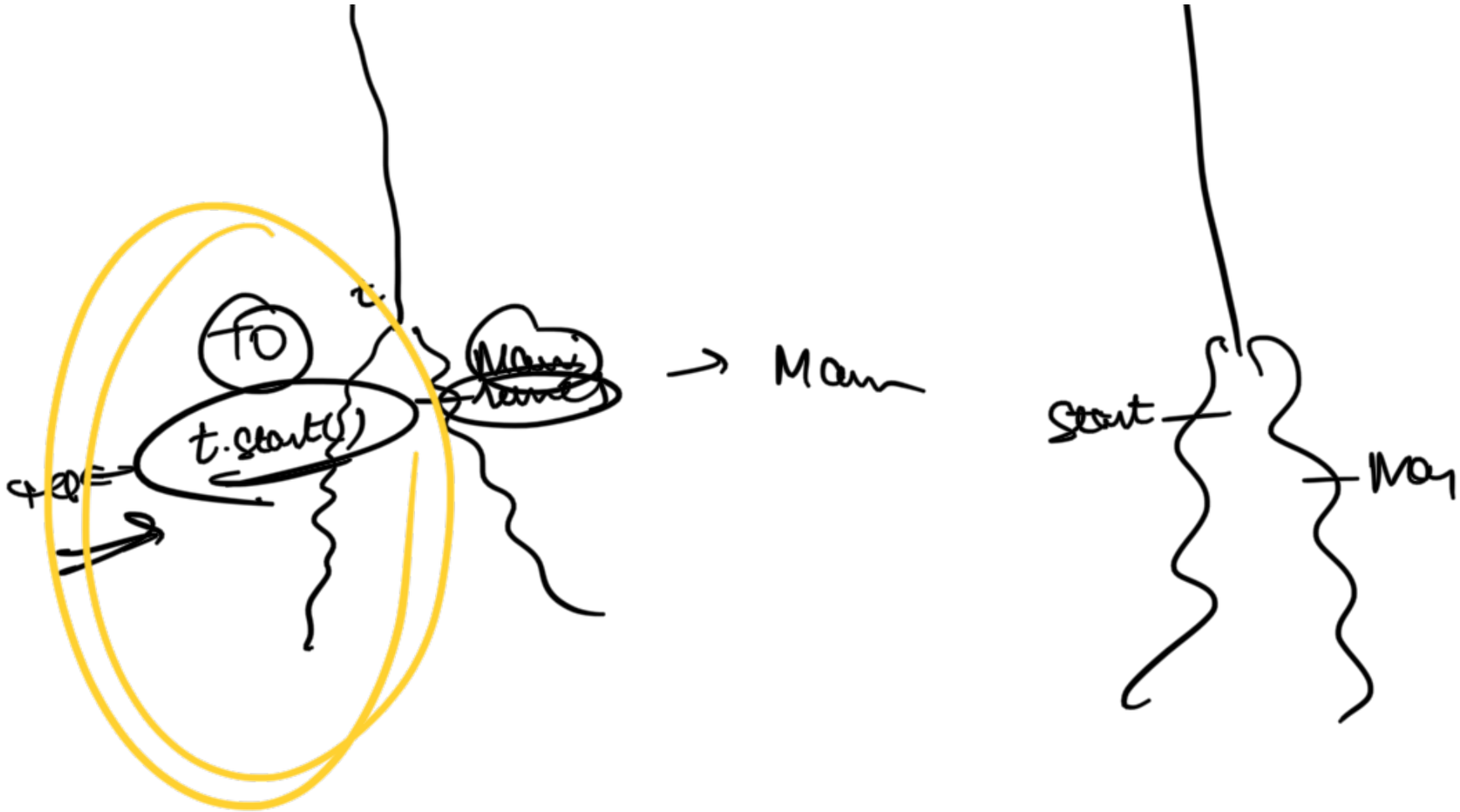
phw.run()

✗ NOT A NEW THREAD

```
Thread t = new Thread(phw);
t.start()
```



Main



Q^n Print Numbers 1 to 100 each via a Separate thread

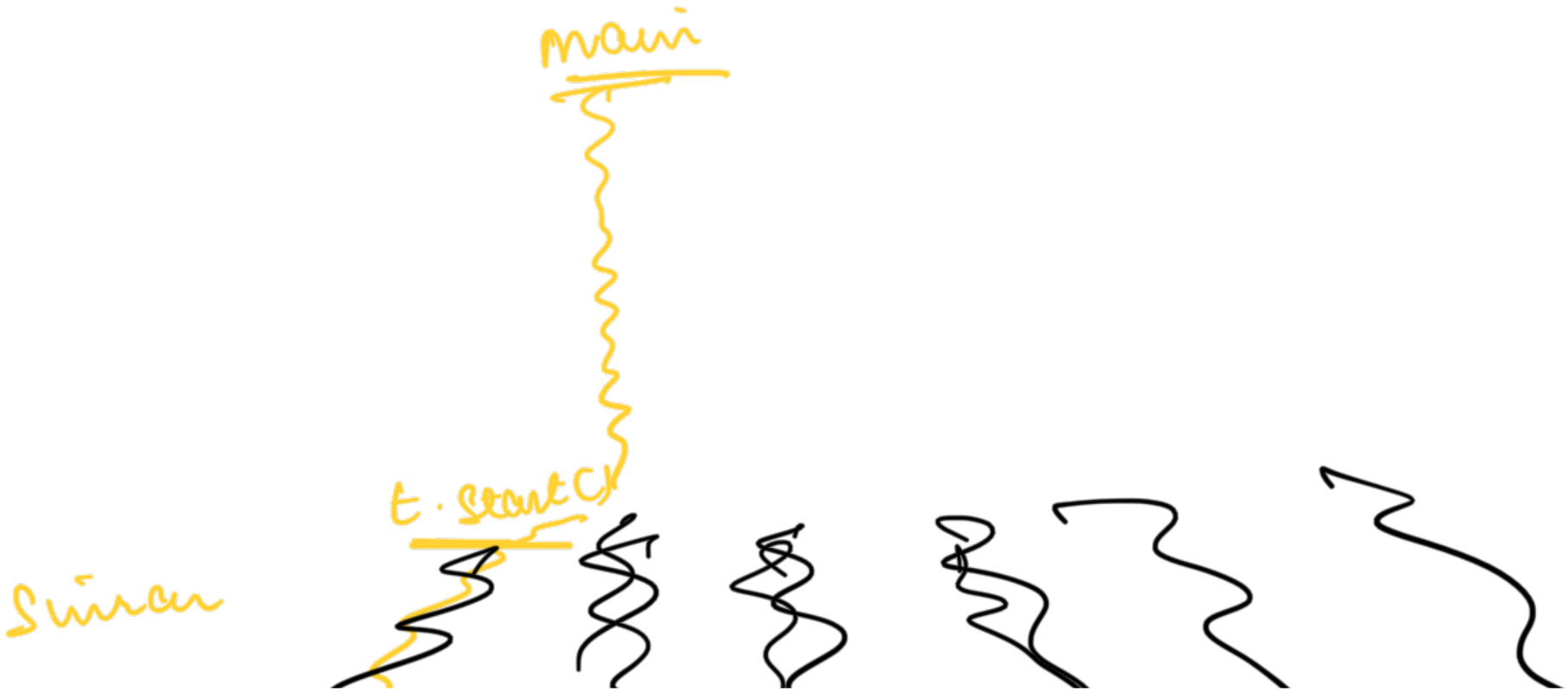
100

Task: Print Number

Print Number implements Runnable {

class

```
int Number;
Print Number (number)
this . Number = Number
}
run () {
Sout (this . Number)
}
```



Main Program

{ ① Creating a task

② Creating a thread and deciding when should thread start

Rather than doing both of these our Main prog should be concerned with just creating Task.

① Creating a Task

(1) Create a

(2) Give this task to an executors

→ Knows best on when to create a thread

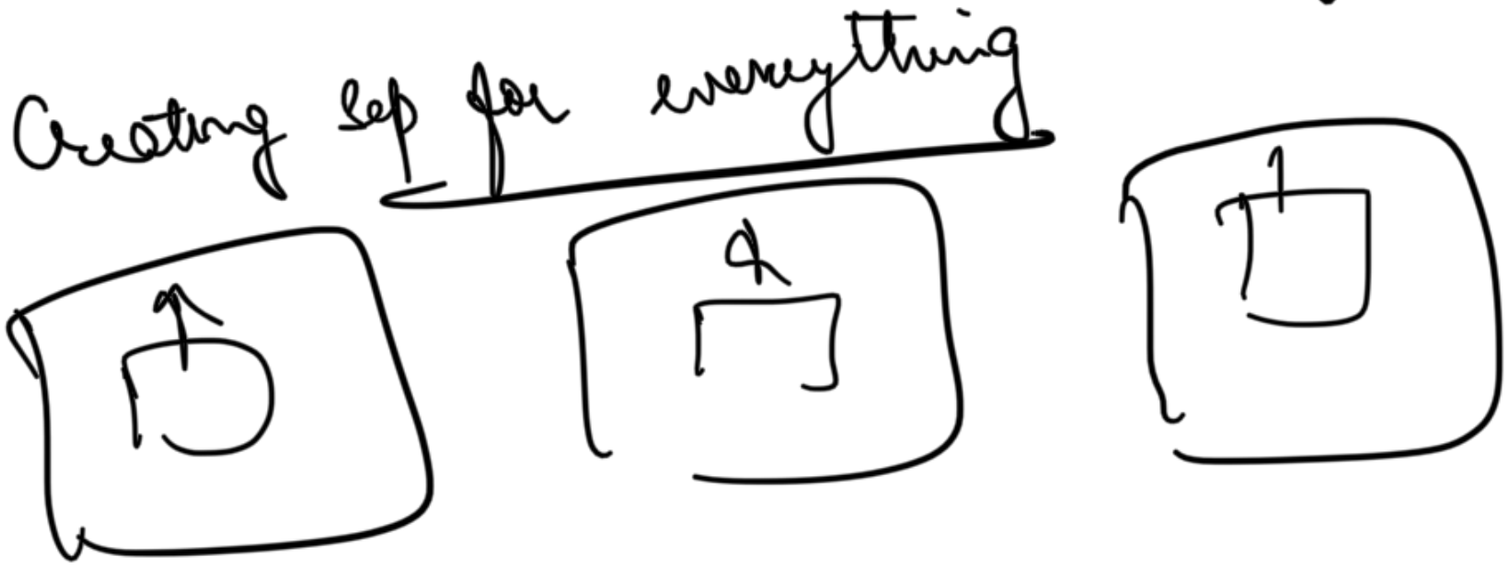
Executors

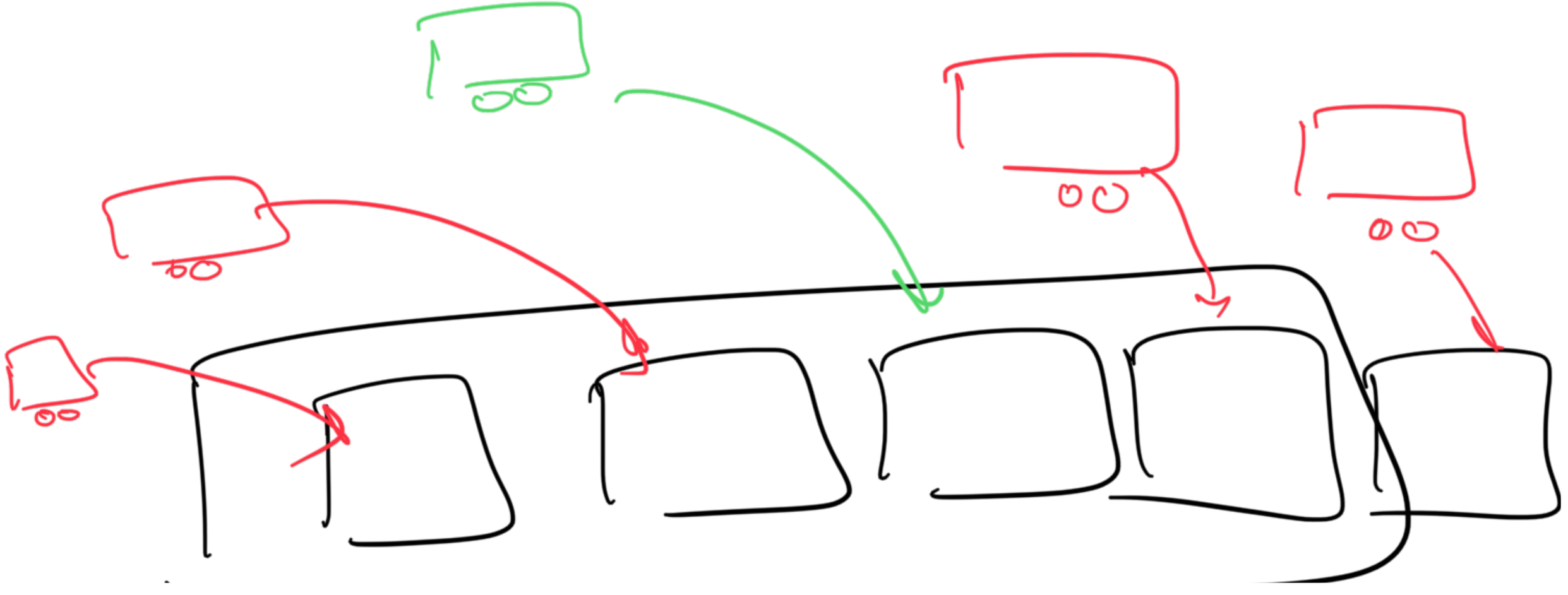
(1) Divided the responsibility

(2) Utilization of resources

Thread Pool

Honda has to create 1000 cars in one day





4 parallel lanes

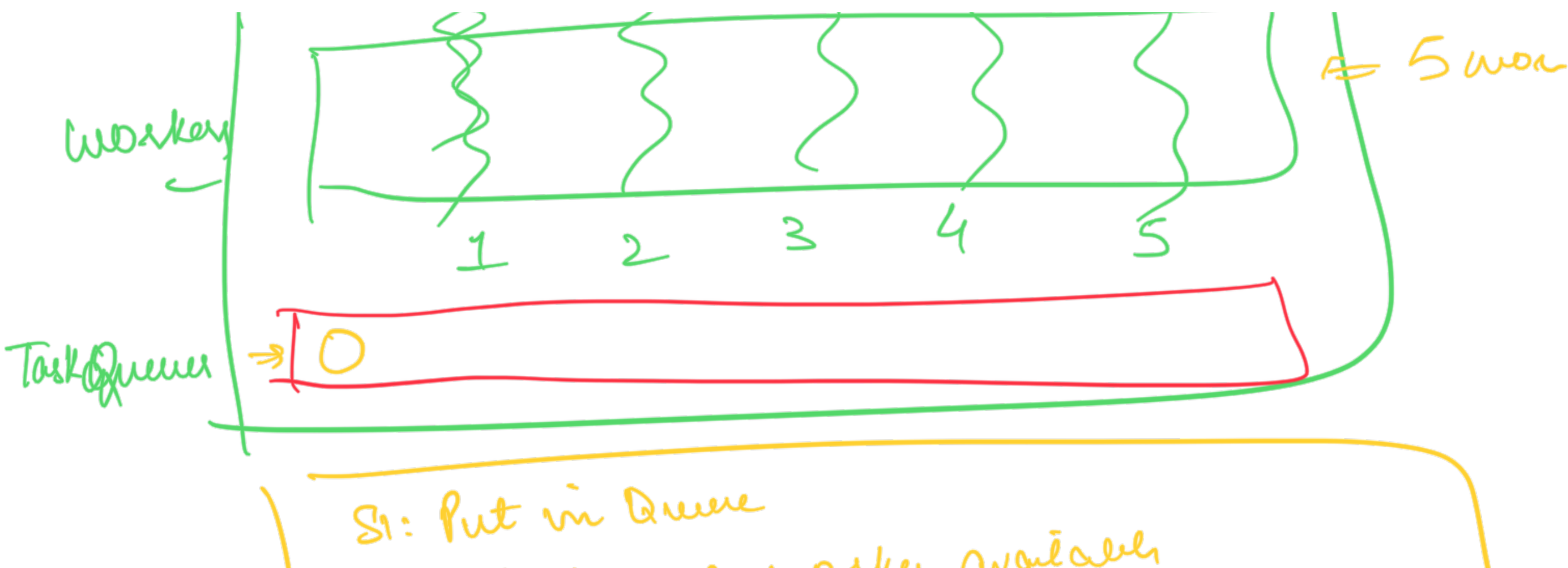
Thread Pool

① Worker Threads → Production lanes

② Waiting Queue → Pending Requests

Car Factory

Executor



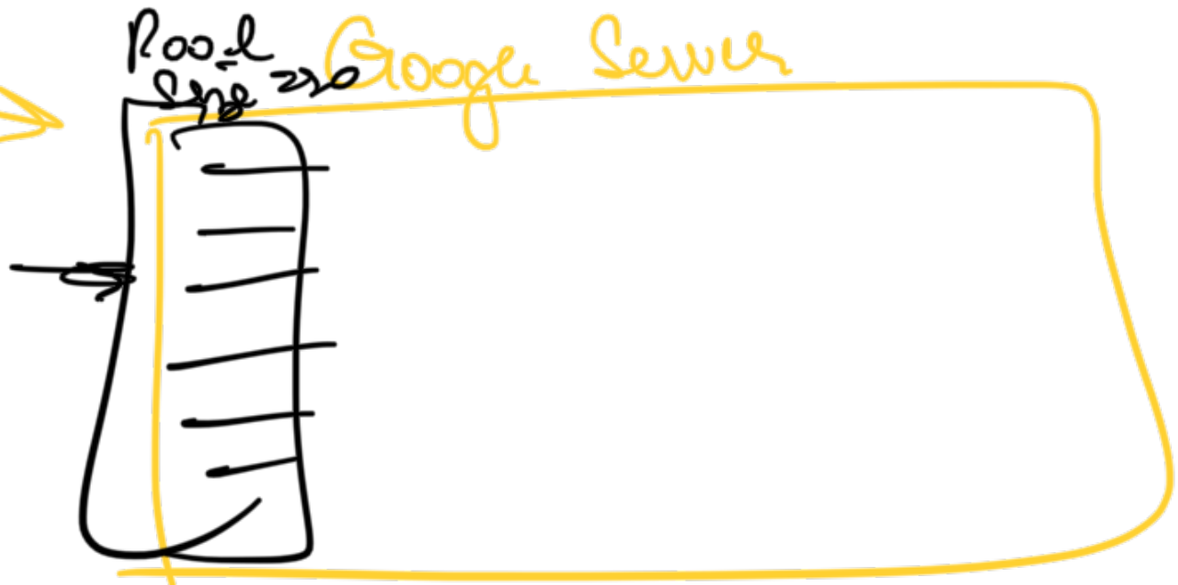
S1: Put in Queue

if yes: assign to the worker thre

else: wait

100 Requests

maxthreads = 20

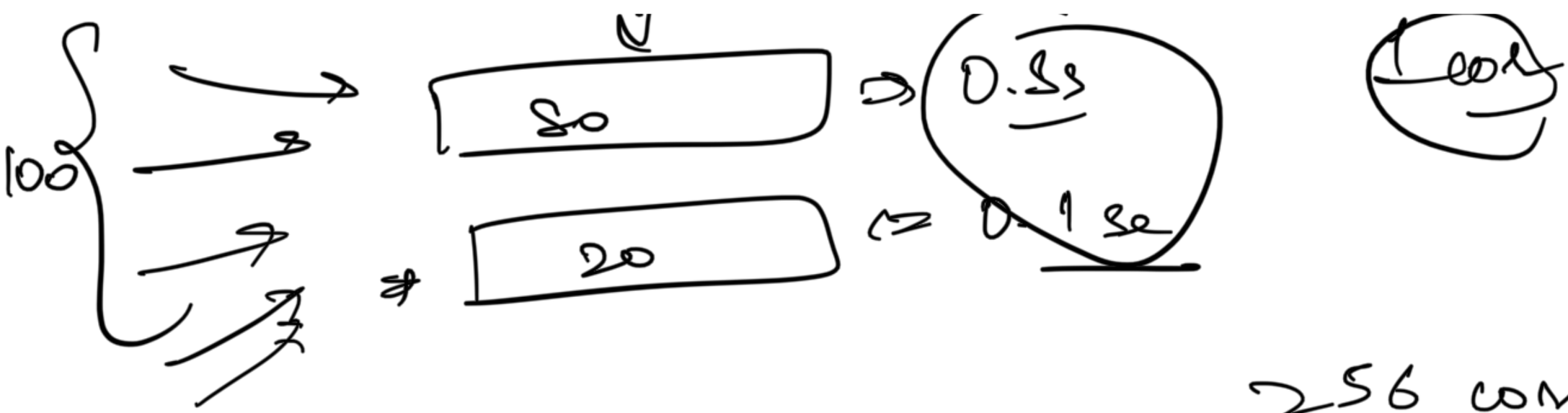


→ Every req is handled in a diff thread

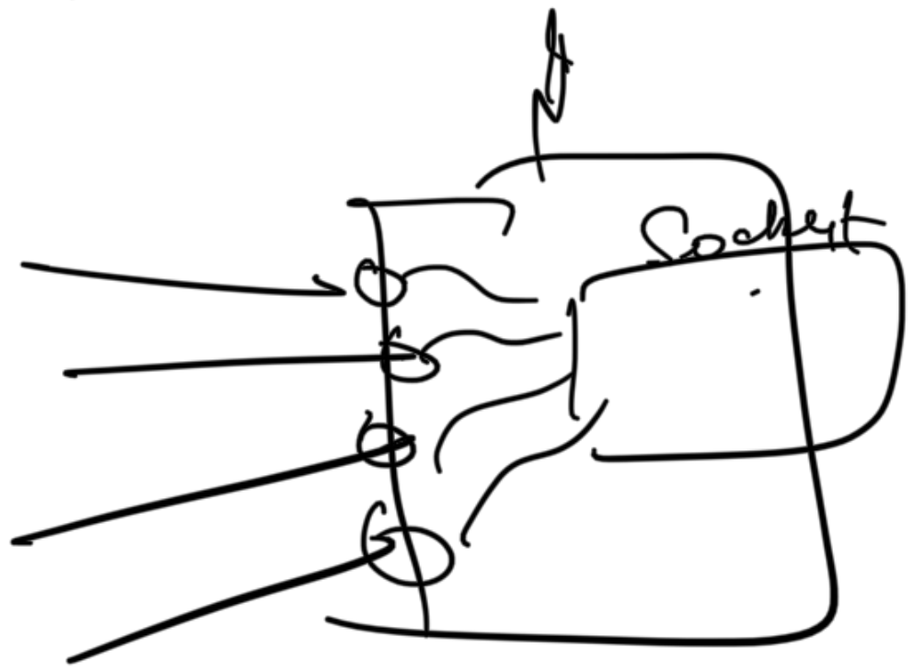
12 - 20

BMS

32 & core
256 GB RAM
4TB S.

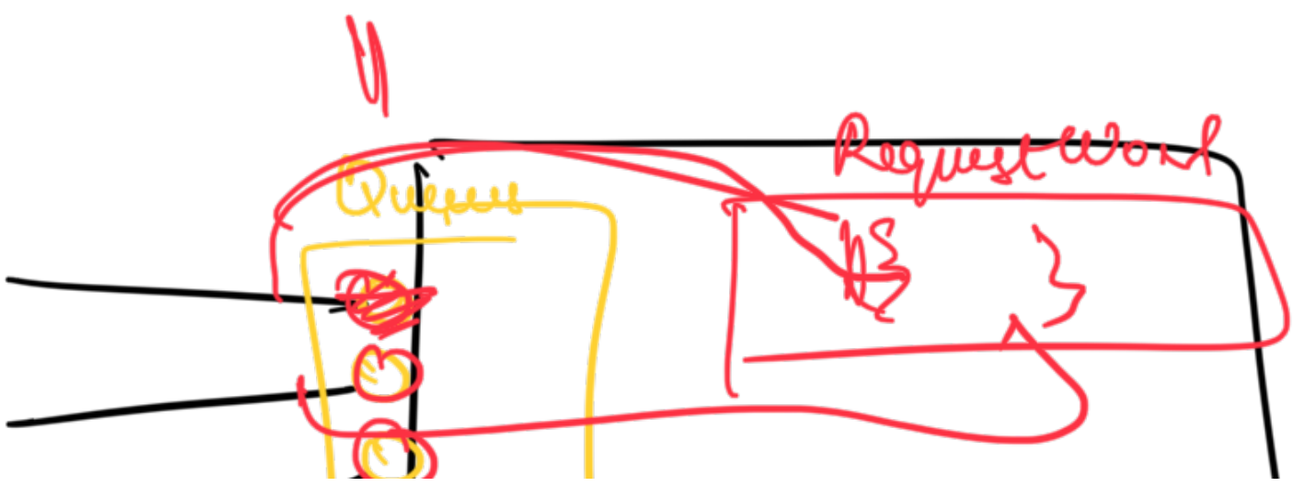


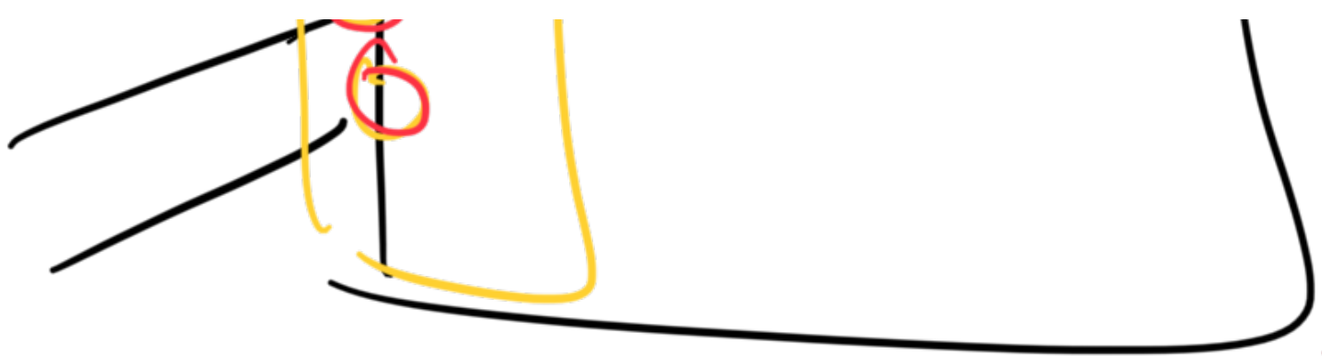
256 com



≈ 1000

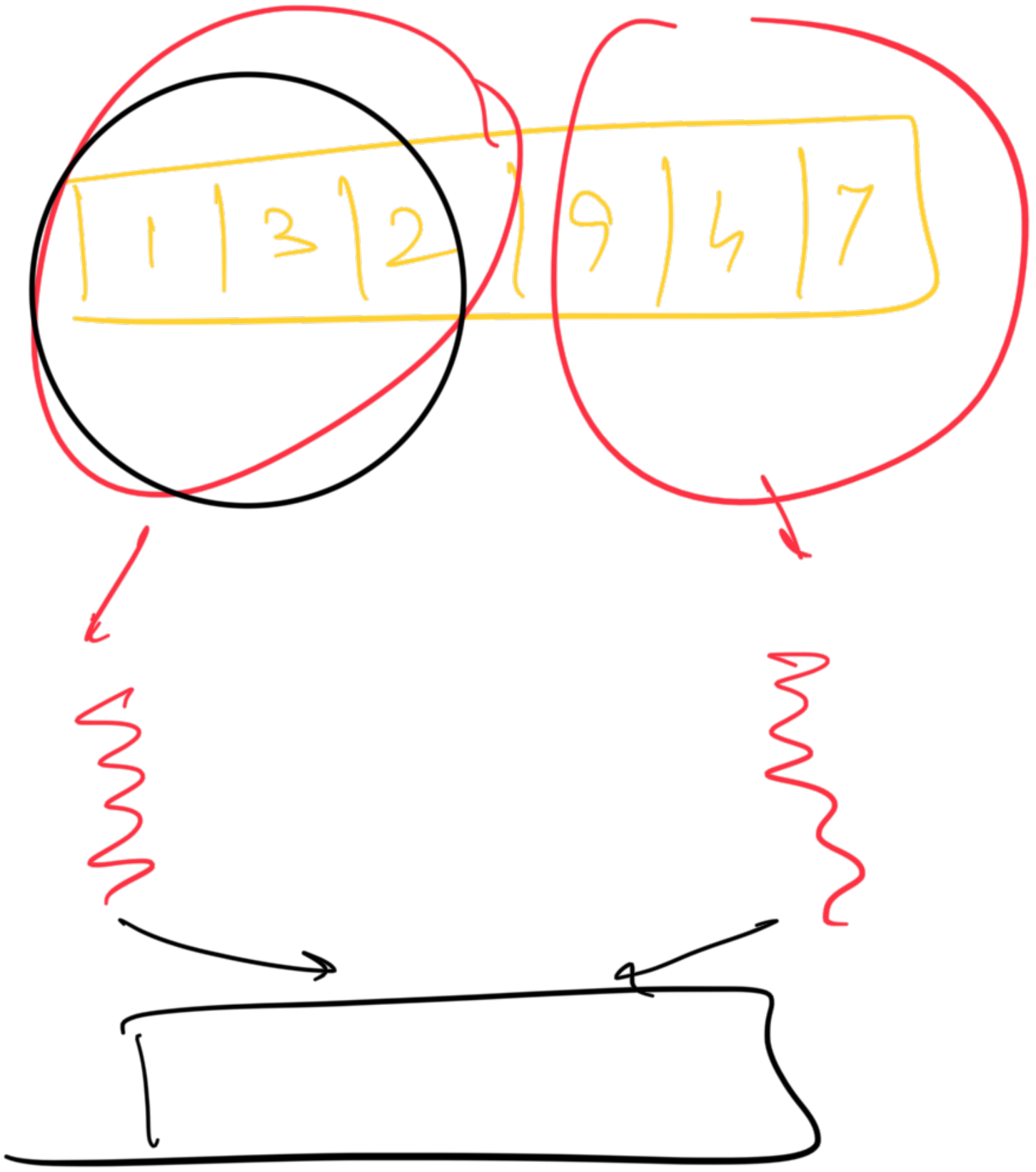
PS-120





⇒ we want threads to return some data to us back

Sort



Q^n : Can I get a return value from a thread

void runs

⇒ interface Callable {

String call () {

// do something
return 1

}

}

① Create a class for the task.

class Sorter {

}

② Implement Callable

class Sorter implements Callable<T> {

T call() {

}

}

③ Replace T with return type of the task

Generics
Template

class Sorter implements Callable<List<Integer>> {

List<Integer> call() {

→ // do the work

}

}

3

vector<int>

vector<string>

vector<animal>

Templating → Generics

vector<t>

void add(t data) {

}

Main


Sent comp
area
SL
SR
SL