OS - Threads and Synchronisation

① Executors ✓

- Thread Pools ✓

② Callables

- Futures

③ Merge Sort (Multi threaded)

---

④ Synchronisation

- Adder Subtracton

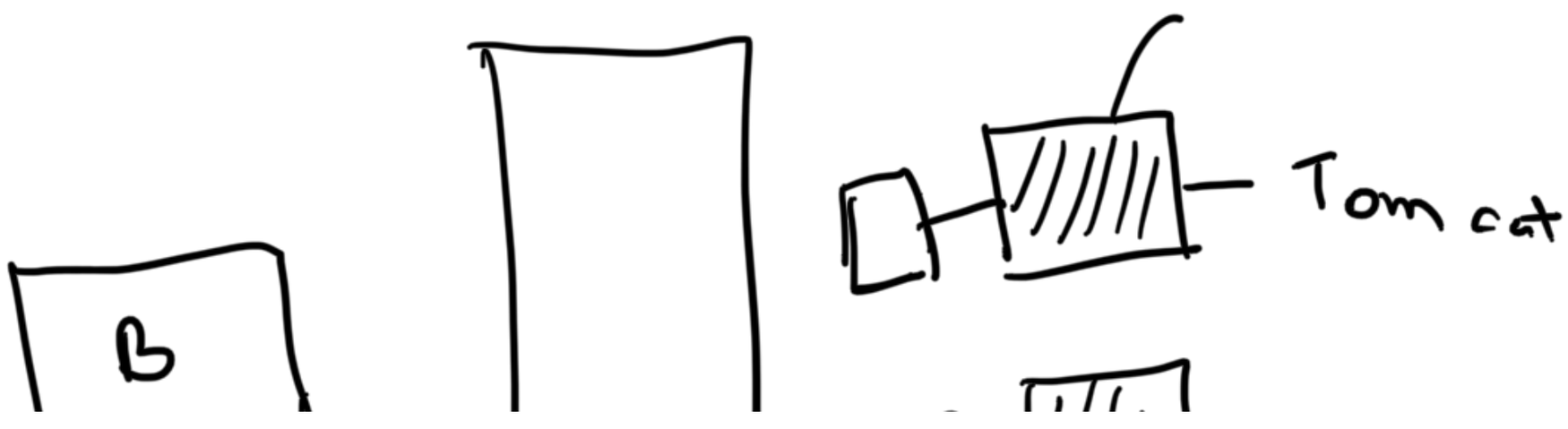
→ Build an application server

→ Apache Tomcat

→ Express

→ Jetty

→ Derby



nginx ⟶ gunicorn

Apache ⟶ Tomcat

① Recieve requests

② Send responses

- Out of memory
- context switching

Executors

① Runnable — Request Handler

② Create a new thread

— new thread (r)

① Logic

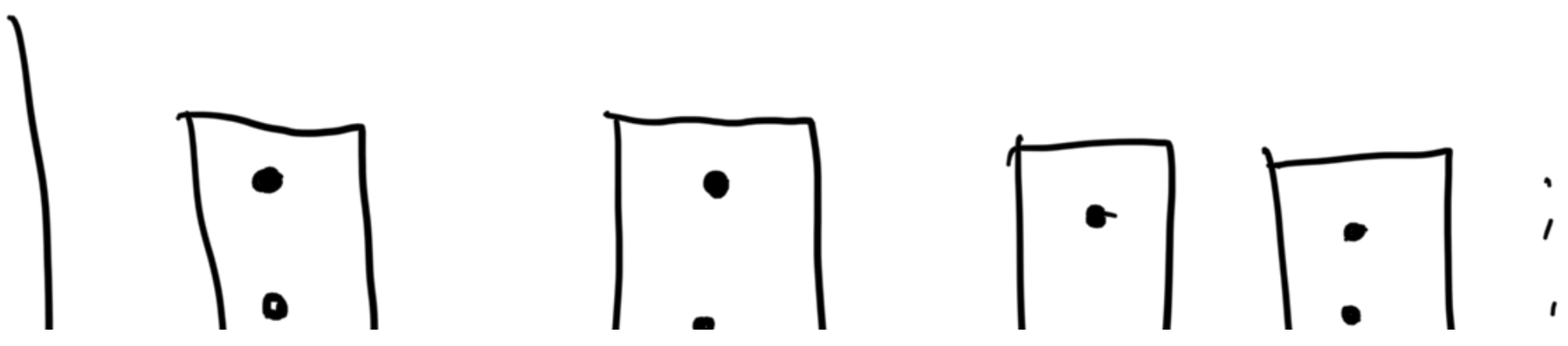
② Running

Executor

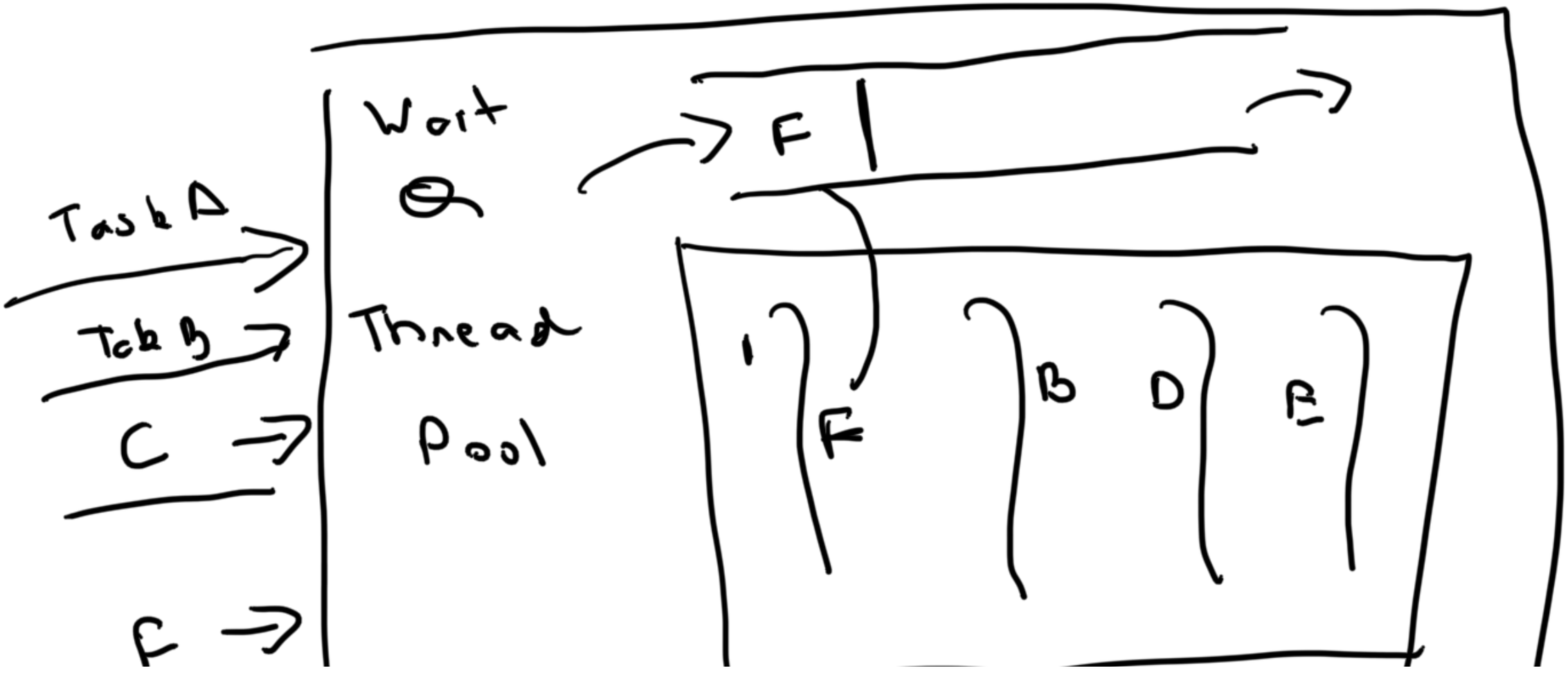
- execute (Runnable)

- Optimising

Can Factory 10000



Thread Pool



→ C3P0

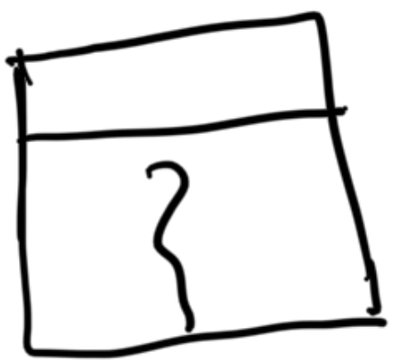
→ Execution - I

- Fixed size thread
- Cached thread Pool

200

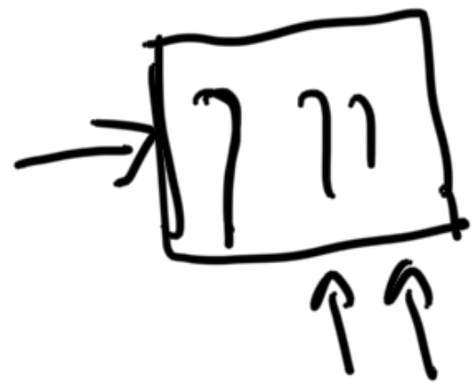
Application Server

- handle requests



- Process
  - huge mem. waste
  - context switching

- Thread



- JDK
  - Thread Pools
    - reusable collection of threads

- Executors

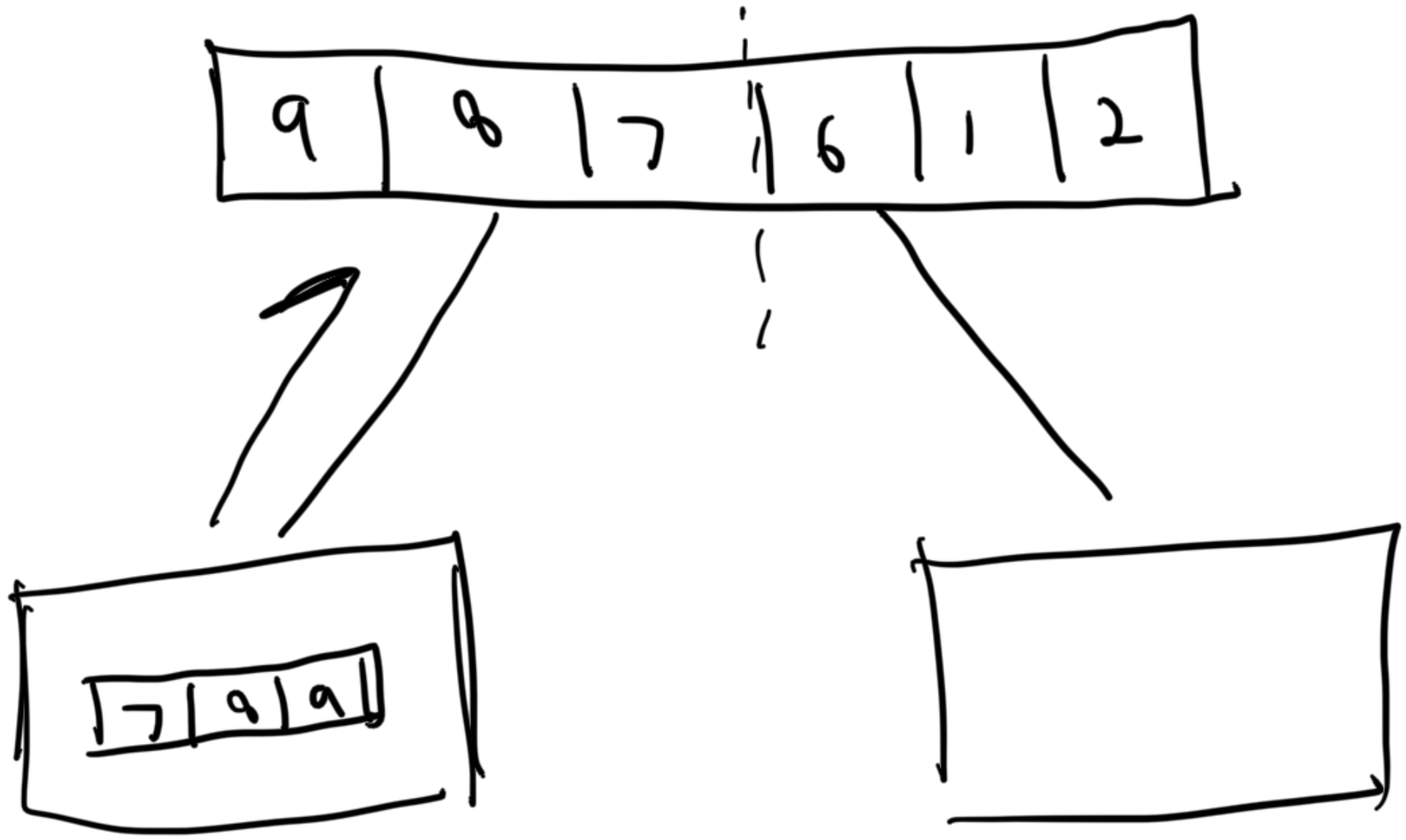
SRP

- Logic
- Executor
  - execute(Runnable)

---

① Runnable
  - run() → void

* does not return anything



* Callables

→ Interface

○ call( )

→ Generics

Callables

→ Like runnable

→ Define a task

→ returns a value

Merge sort

```
class Sorter implement Callble
                        < List <Int> >

    public Lis t< Integer> call ( ) {

}           }
```

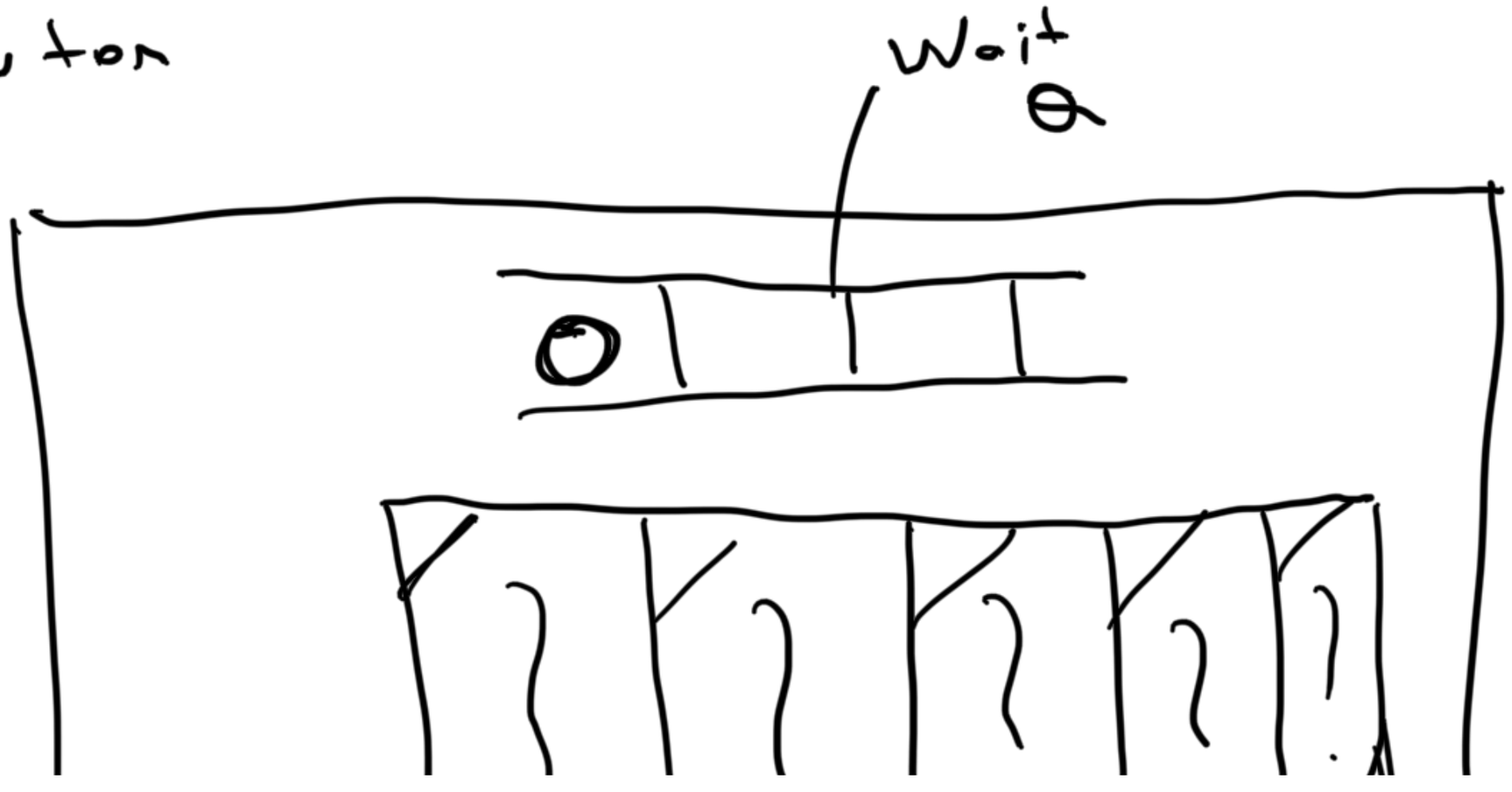
Executors. execute (Runnable)

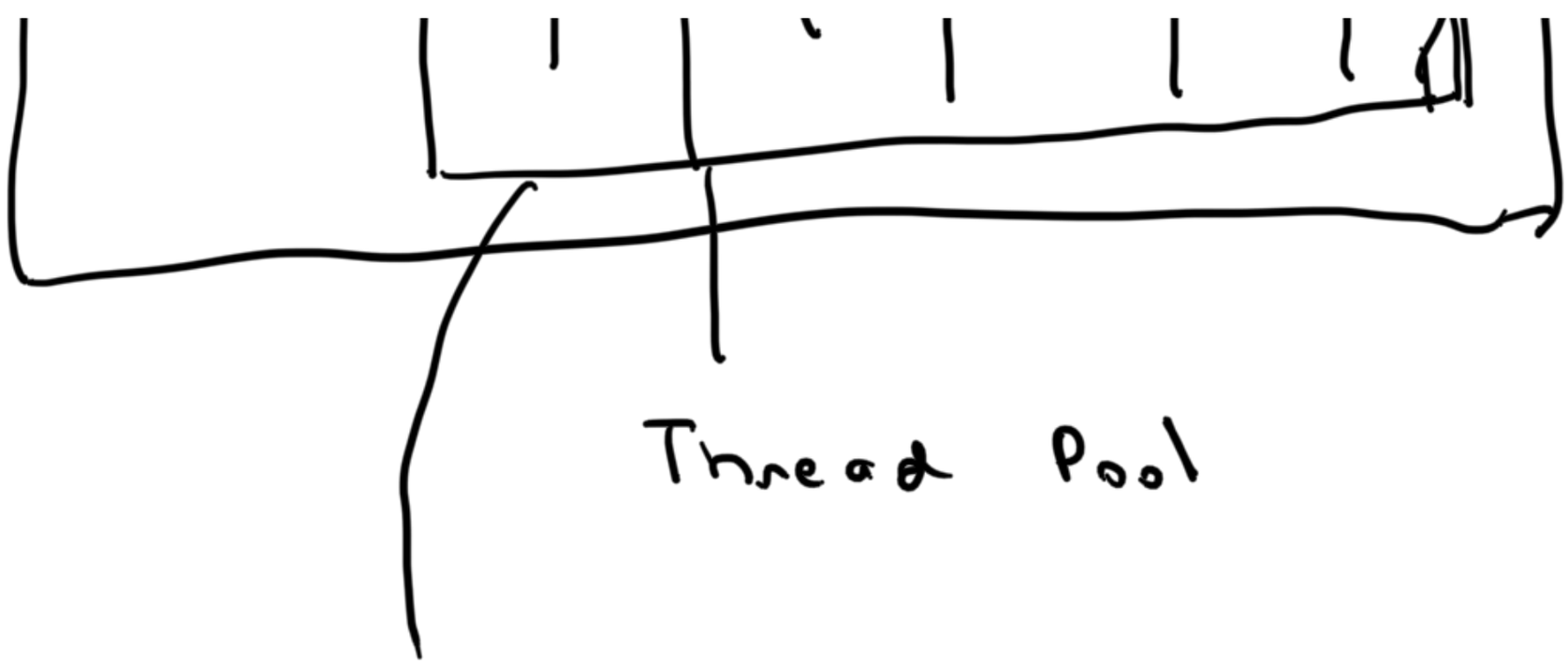
- submit (Callable)

6:17 6:22

10:52

Executon





Thread Pool

Sorten (Wanben)

Functional

- Single method I (SMI)

↓

abstract

S Am

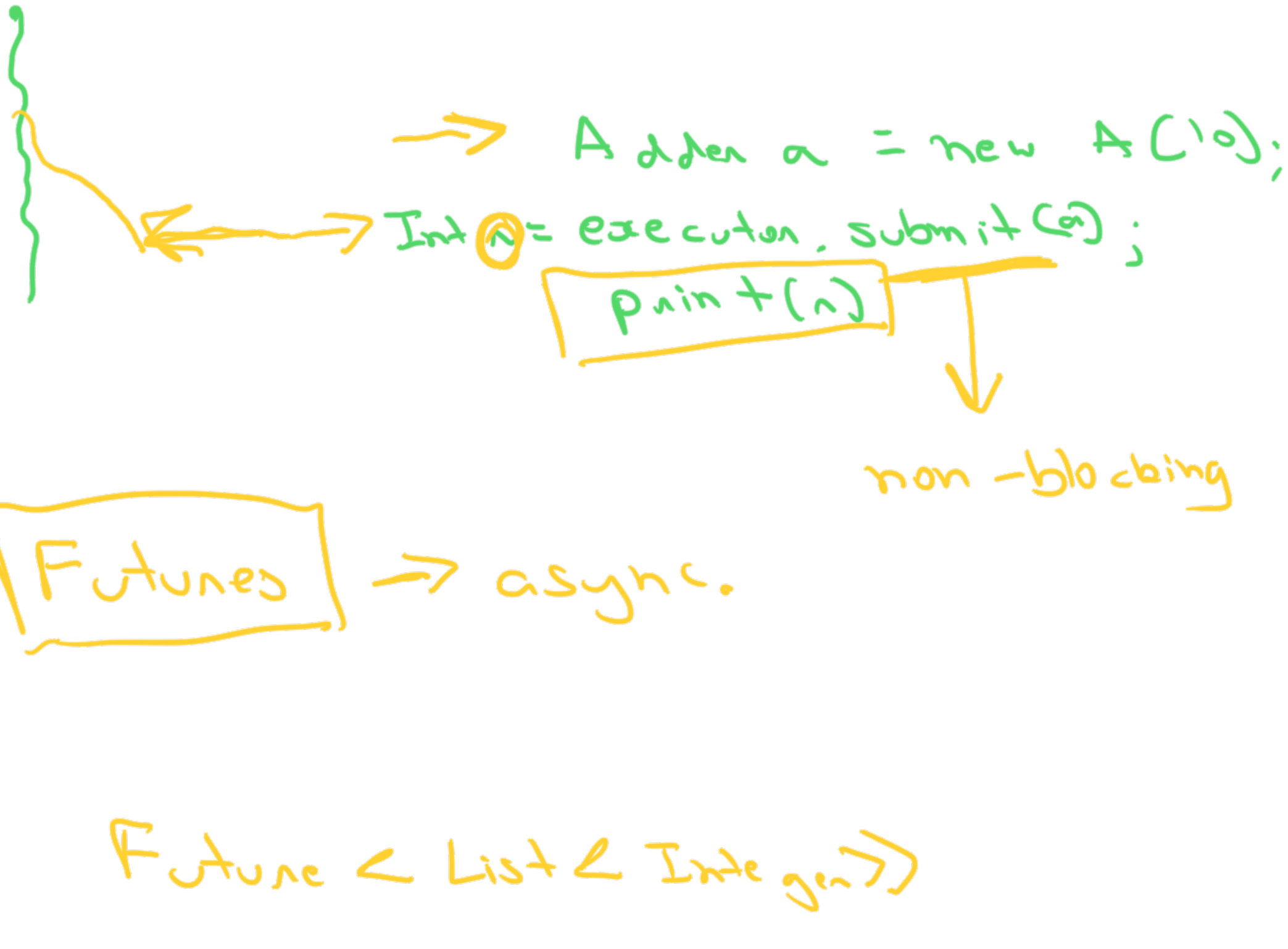
lambda

executor.submit(() => 1+2)

Java Futures

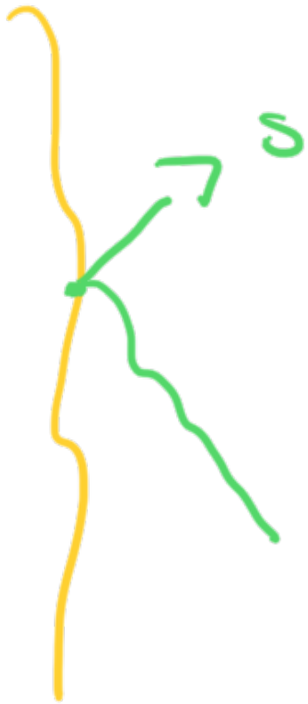
Adder(

count++



Futures → async.

Future < List < Integer >>



Future<Int>

sendEmail

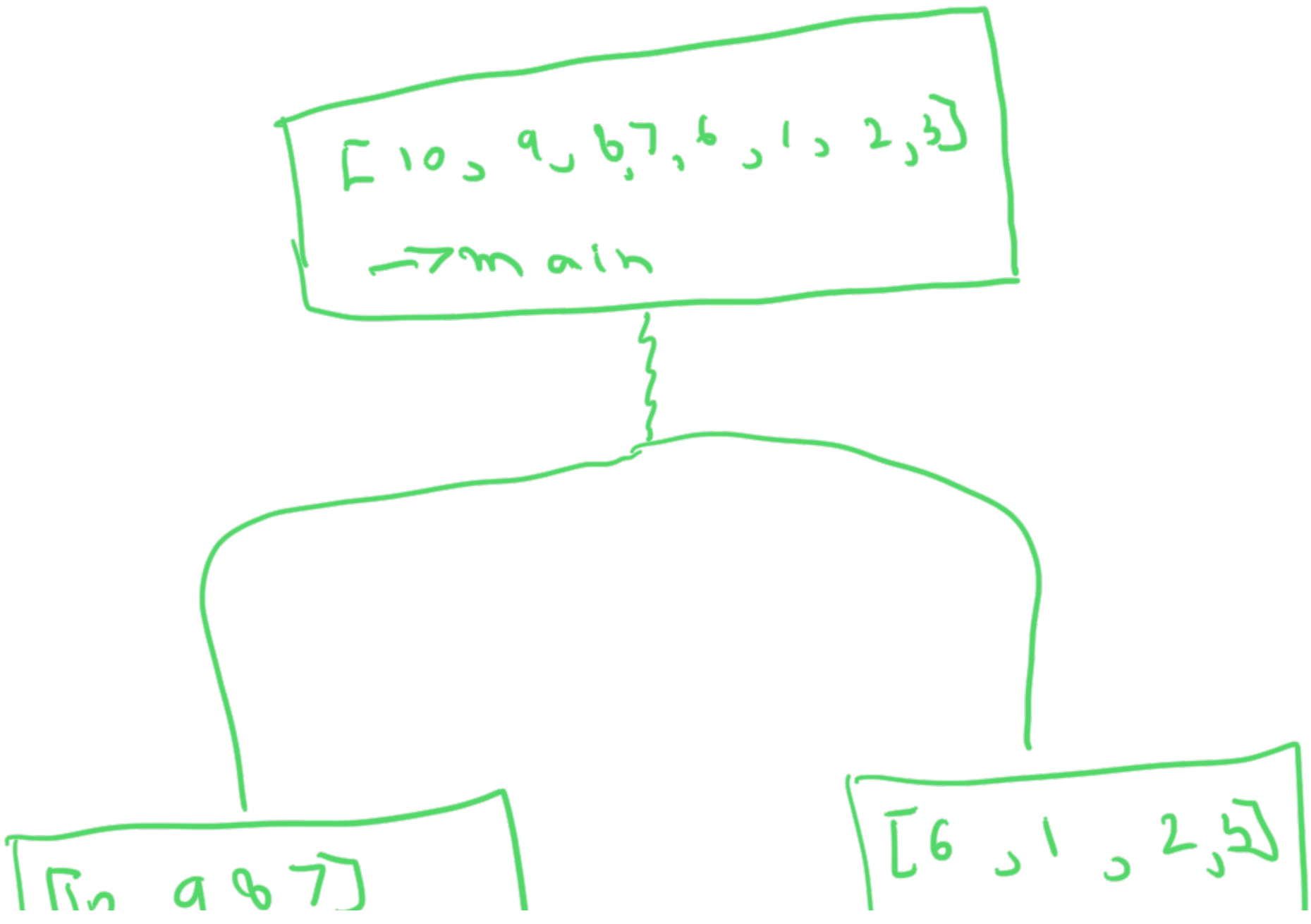
a + b

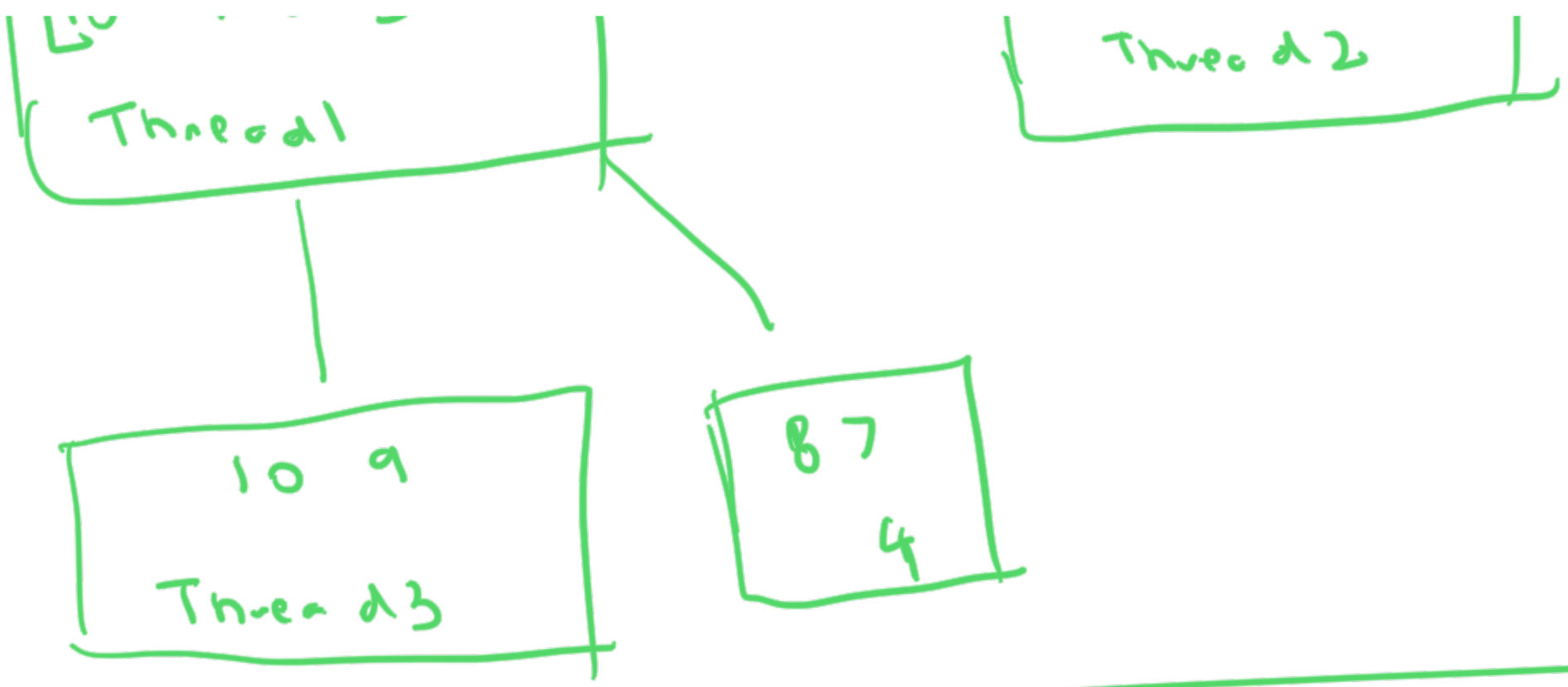
blocking

Int value = a.get()

Callables

- submit
- Future
- get()





Assn

- Quick Sort - multithreaded

subList()

- Immutable
  - addAll
  - removeAll

} USOE

Synchronisation