LLD - Design a cache

① RAW Queries + Url

→ not manageable

→

② DB SDK

→ limited

→ RAW

→ Tight Coupling

$\boxed{\text{auth0.com|authorize}}$

SDKs

③ ORM – Object Relational Mapper

Hibernate / JPA

Domain Language

ORM

- Hibernate ↓
- Jboss
- Mongoose

[illegible]

Mini-ORM

- $\boxed{\text{JPA}}$ ↑
- CRUD

.....

→ Alchemy | Alembic

→ TypeORM

→ Active Record

→ CRUD

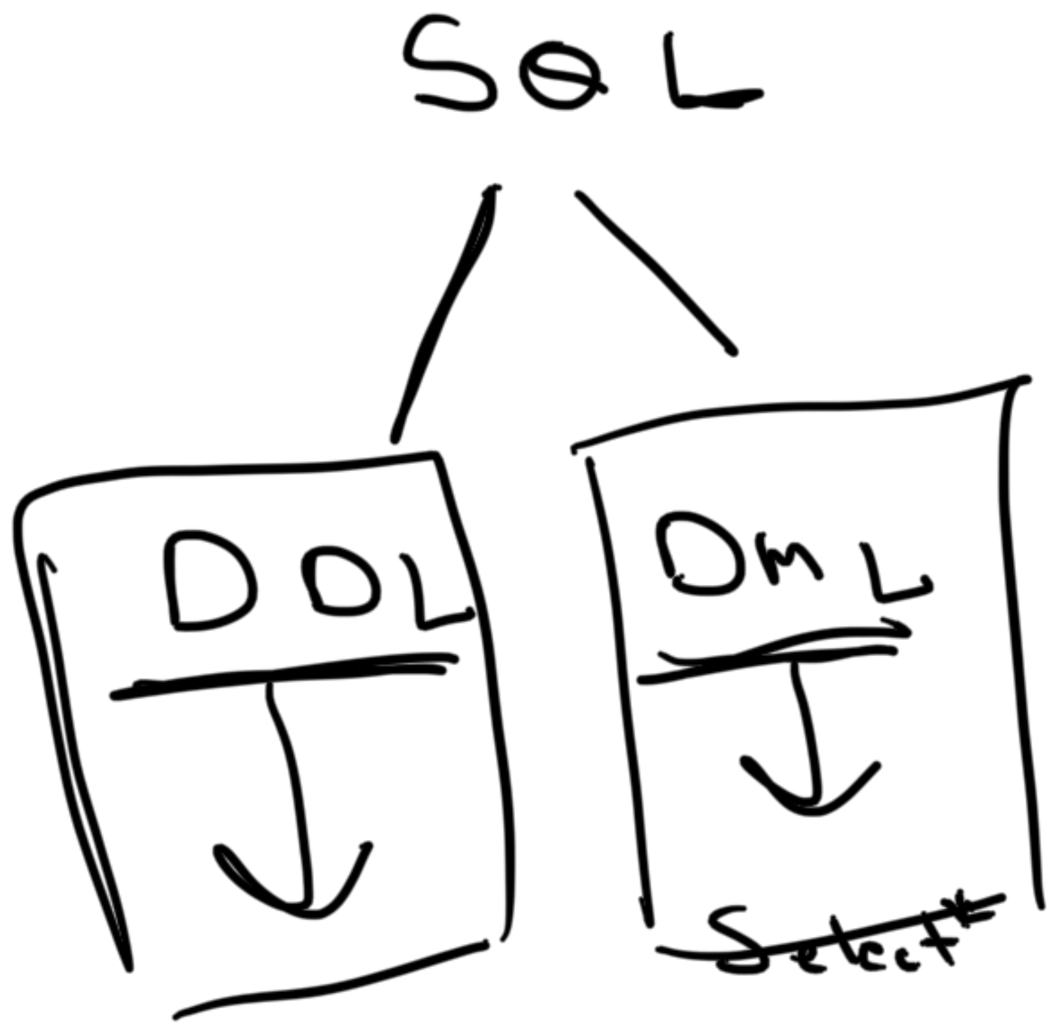
→ Caching

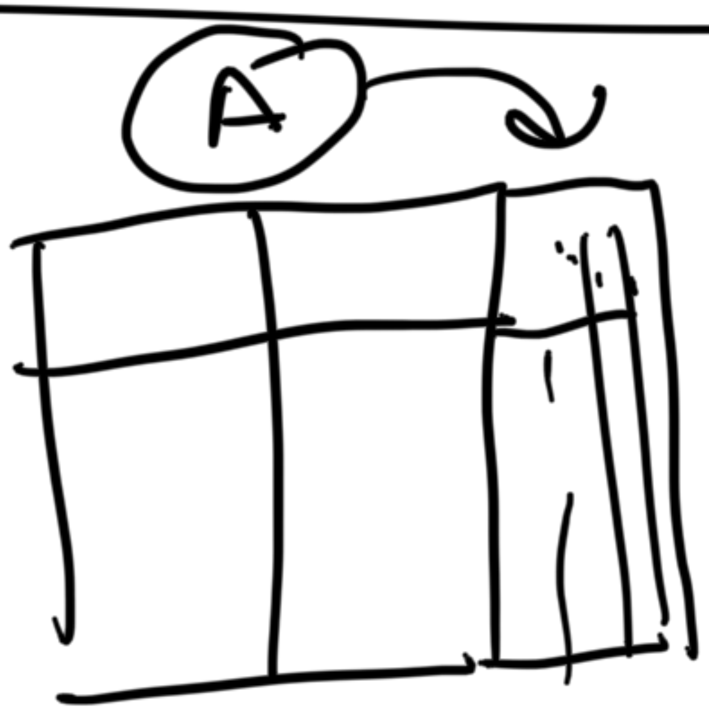
→ Thread | Connection
Pool

→ Validate

→ Monitoring

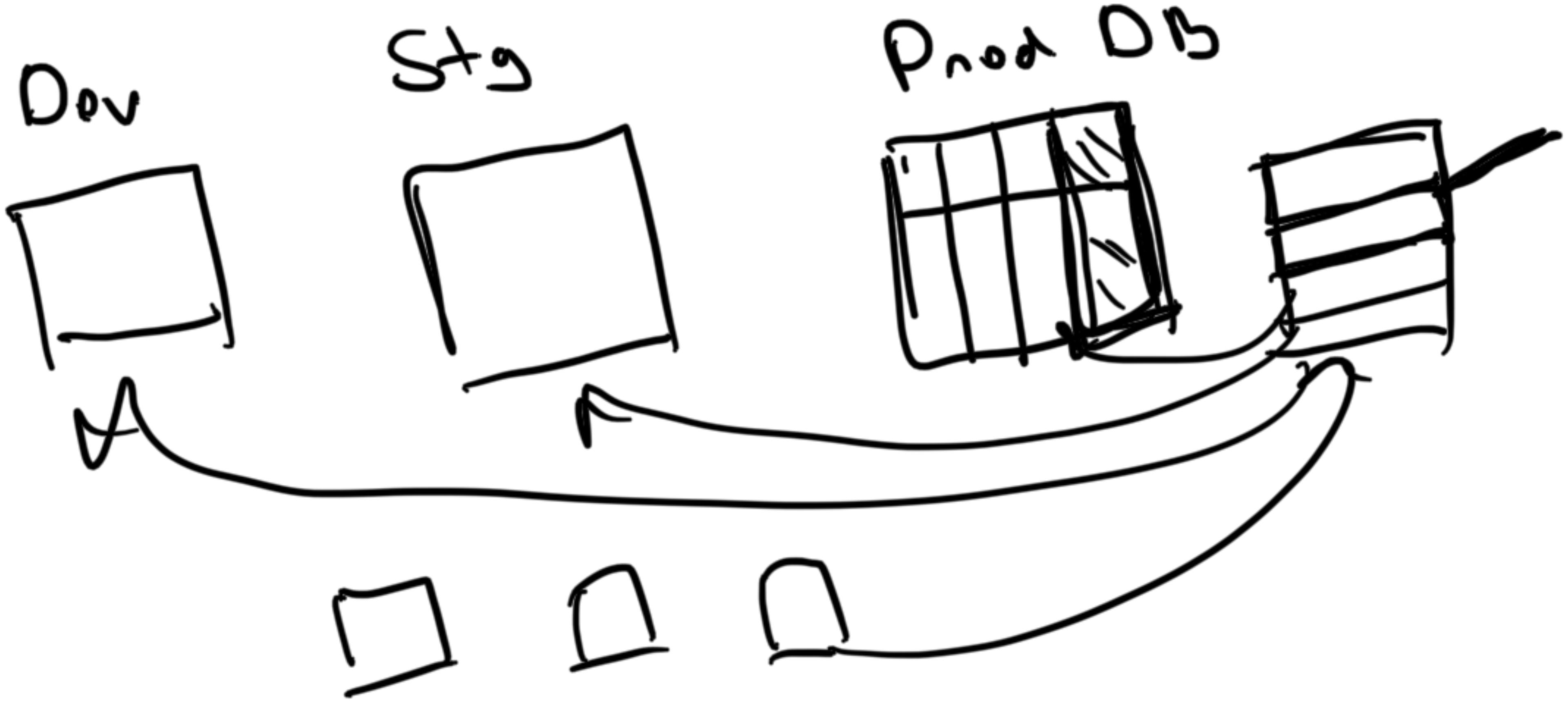
→ Audit





4:22 – 4:25

$\boxed{9:55}$



JDBC	JPA
- connection	- Create
- statement	- read
- execute	

| Repository $\boxed{\text{JPA Repository}}$

$\boxed{15^{th}}$

Design a cache

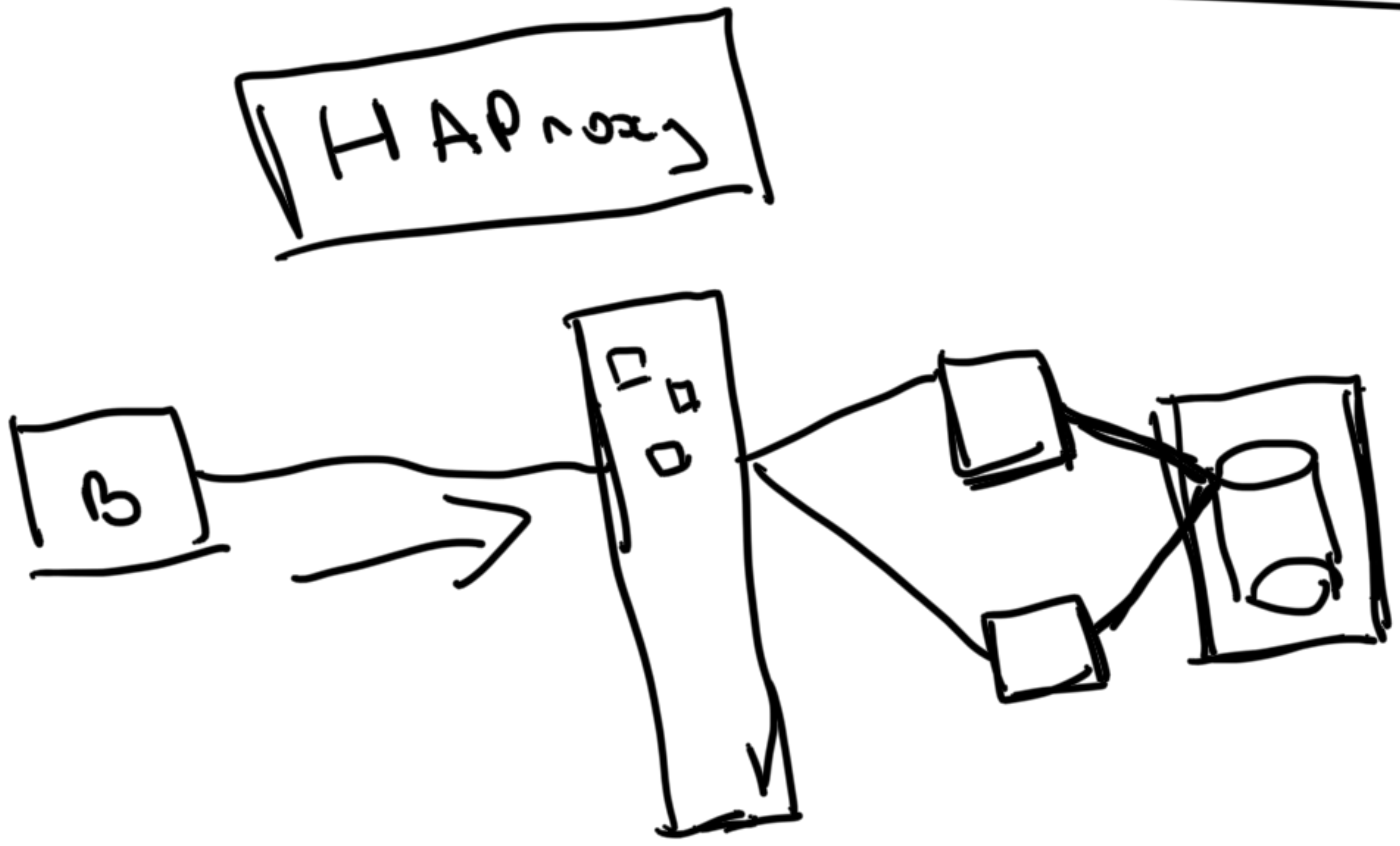
→ Introduction to Caching HLD

Motivation RAD

→ Base set

→ POC [Proof of concept]

→ Iterative



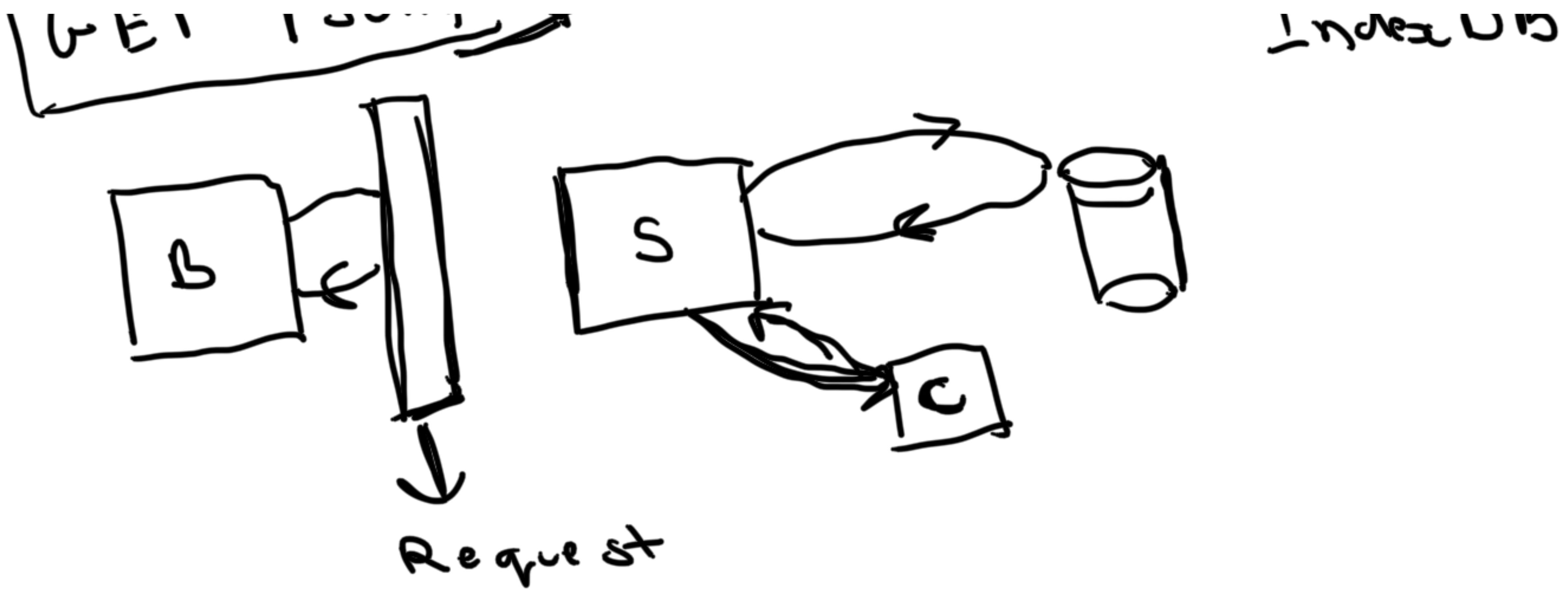
Why do we need caching?

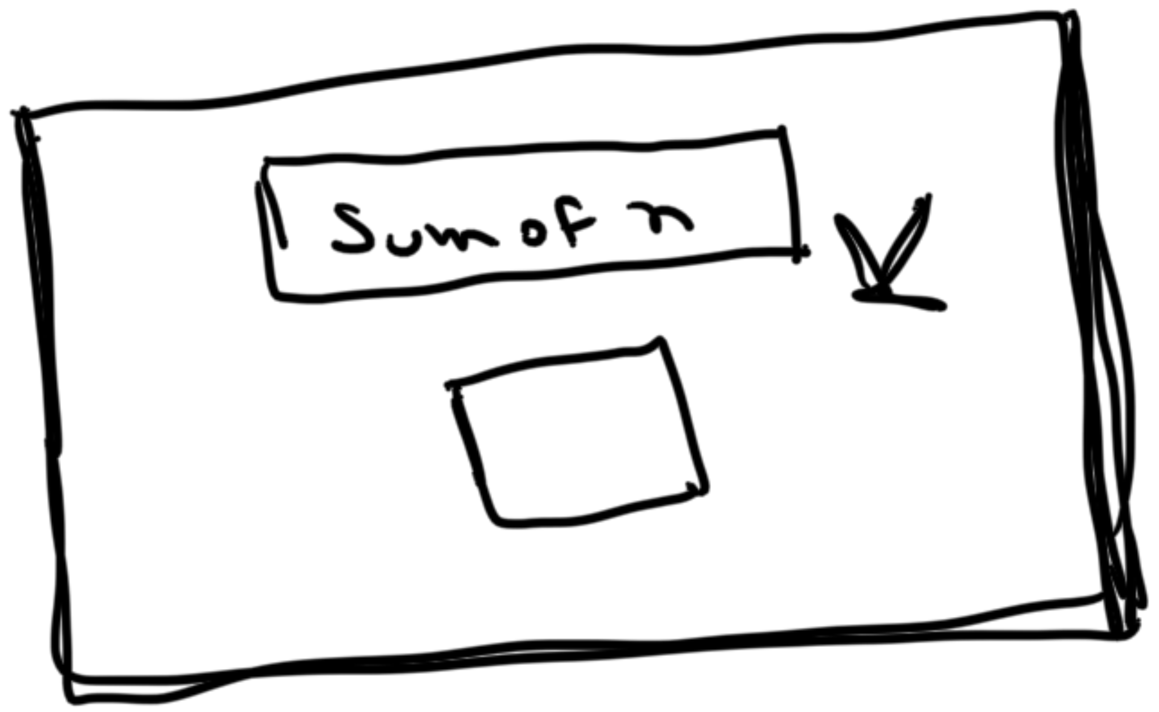
① Disk access

$$\begin{tikzcd} \text{Memory read} \arrow[d] & > & \text{Disk} \arrow[d] \\ \text{Volatile} & & \text{Persistent} \end{tikzcd}$$

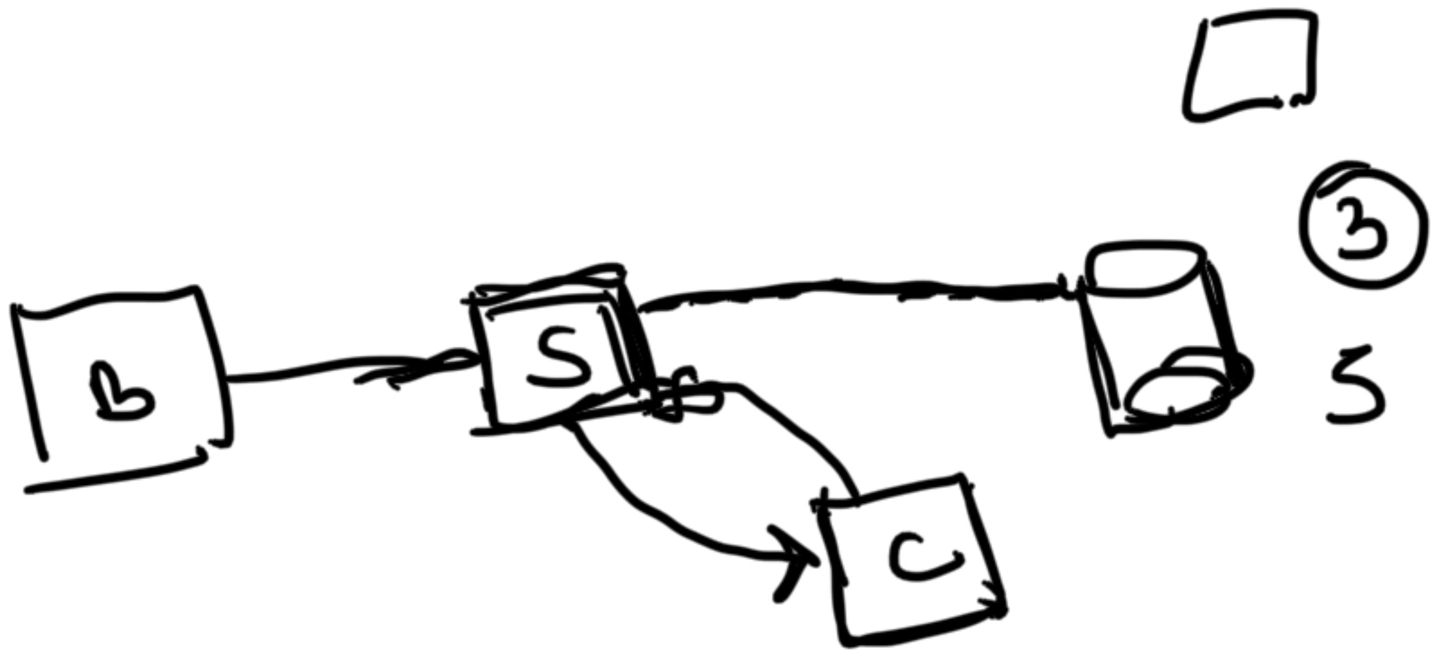
② Temporary store

→ Repitive computations


Index DB
B
S
C
Request


Sum of n

Problems using a cache



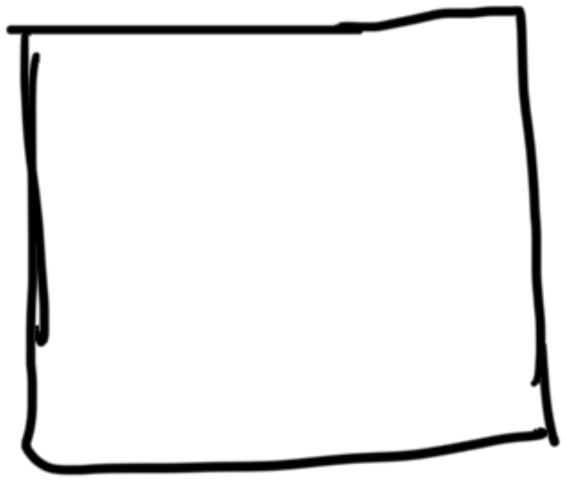
→ Inconsistent [Stale]

→ Expire - 10s

Cache

- → reduce latencies
- → reduce computations
- → At any level
    - → bef. DB
    - → Browser

Requirements



① Key-Value pairs

$\langle$ Any, Any $\rangle$

(2) $\boxed{\text{Size - 10}}$

(3) Each item will have an expiry time of 1 hr.

Behaviour

(1) CRUD

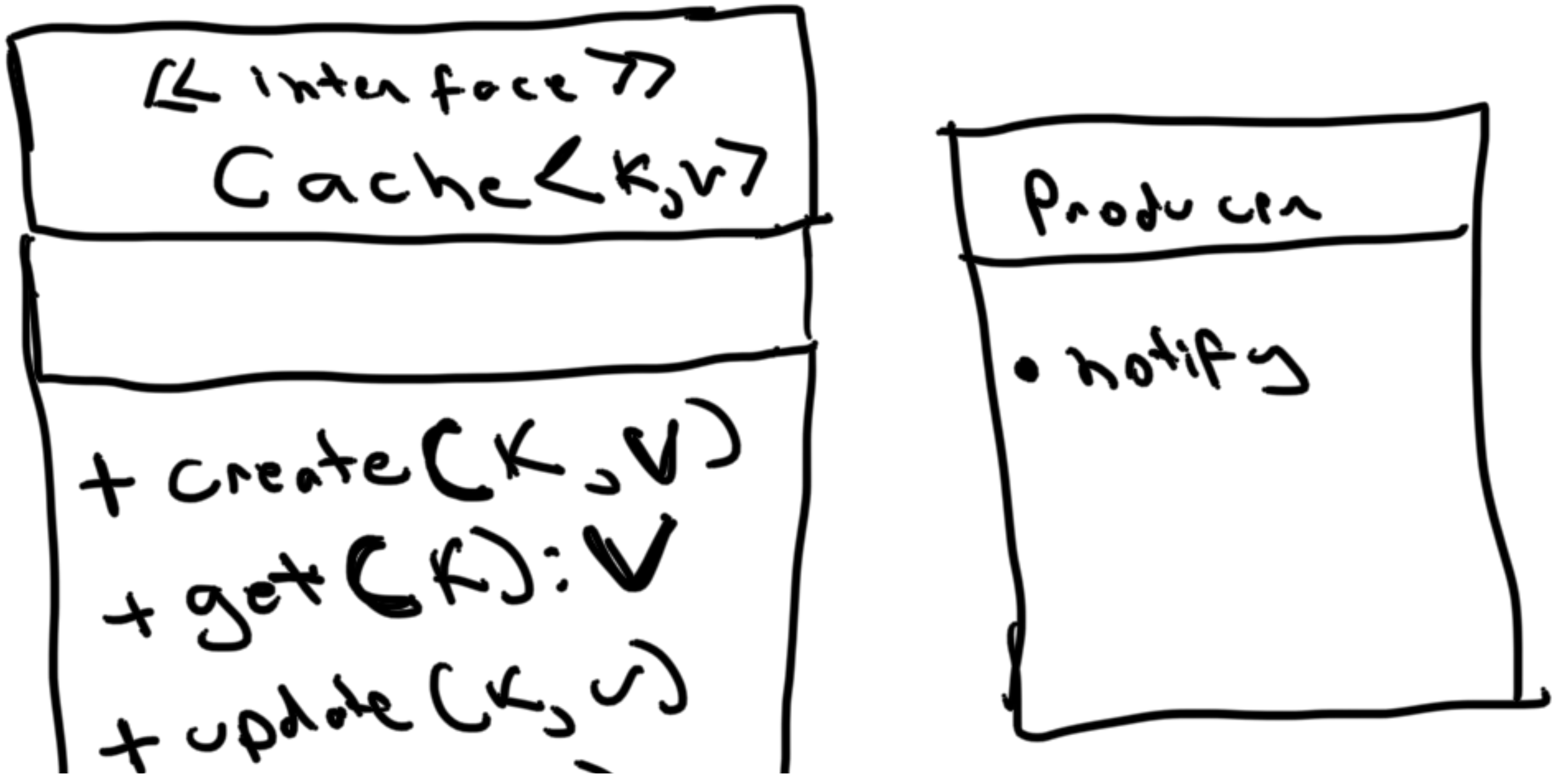
- **C**reate an entry
- **R**ead an entry
- **U**pdate
- **D**elete

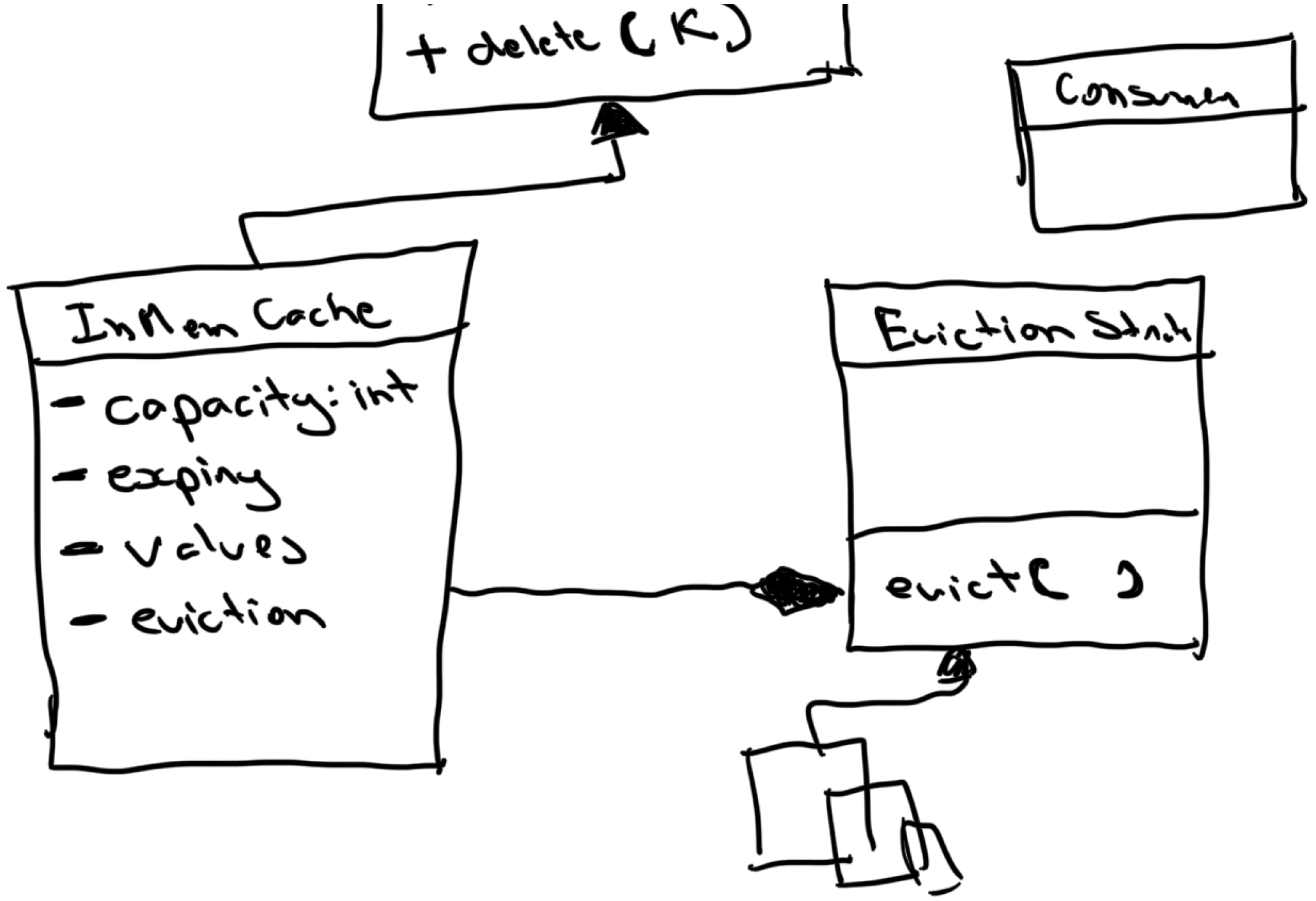
(2) If cache > 10 : Throw an error

Cache

- behaviour

- attributes

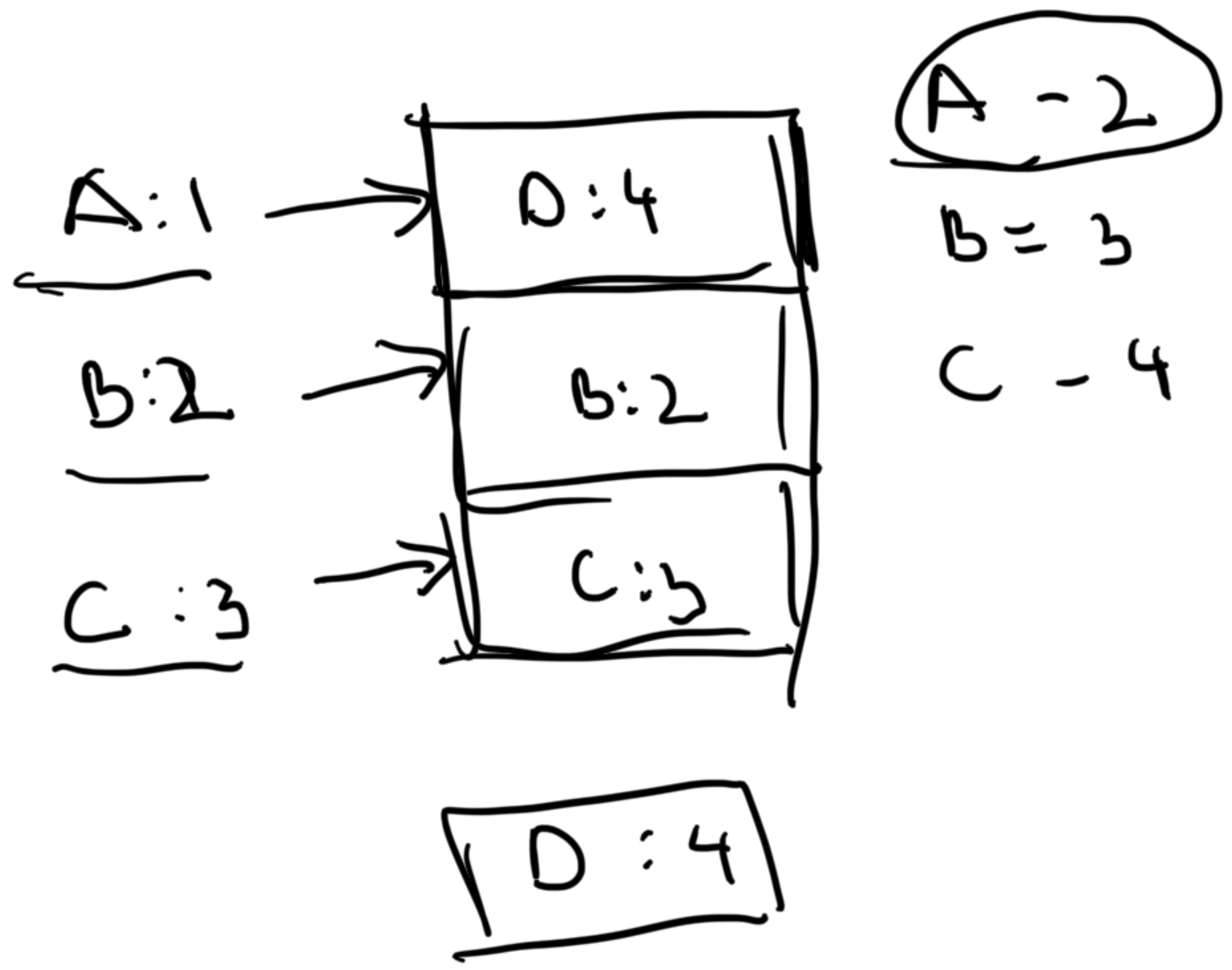




① How to handle overflows

→ Remove

→ remove existing items



Eviction

→ LRU - Least Recently used

→ FIFO

→ LFU - Least frequently used

→ Random

---

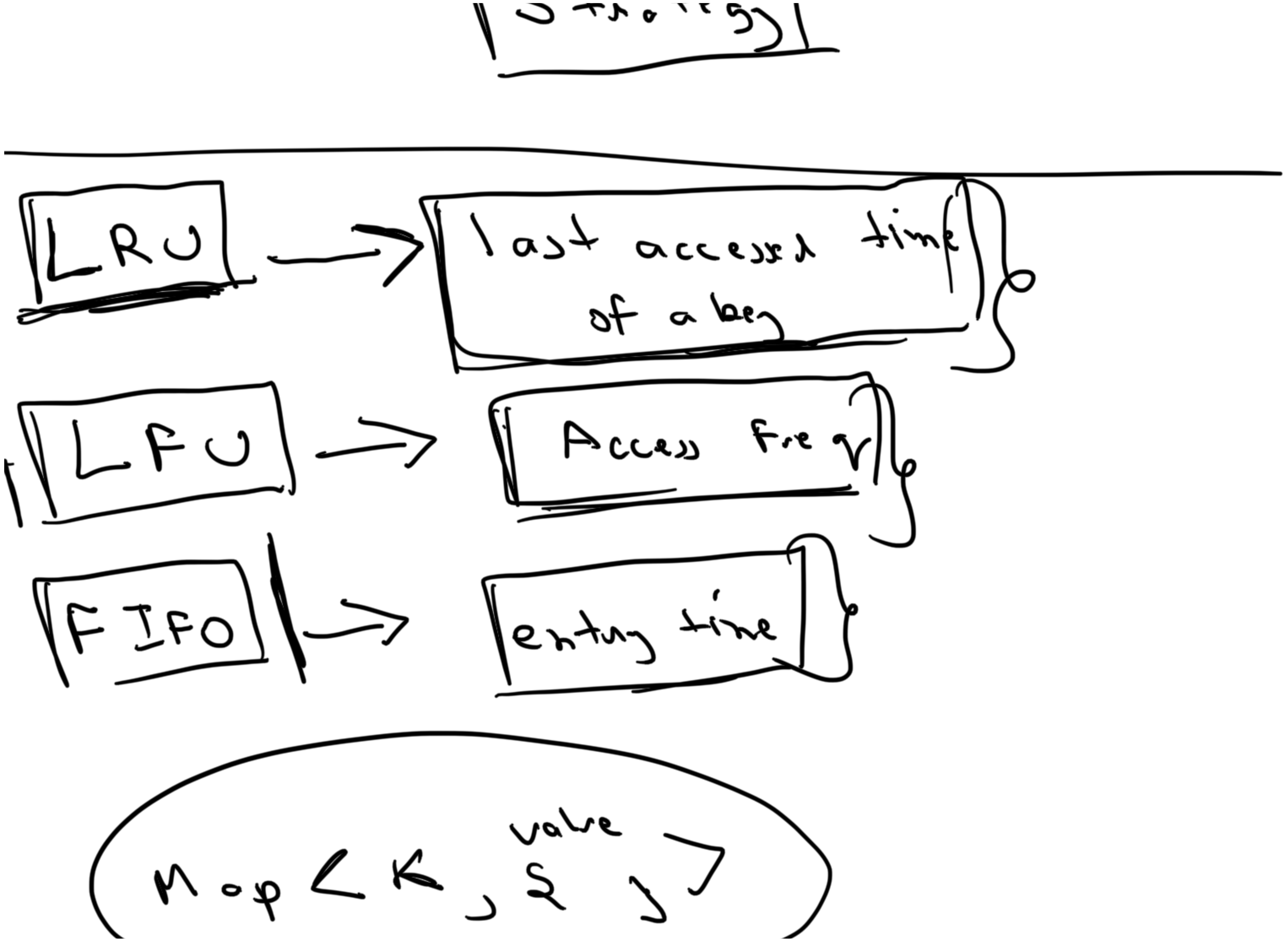
1. Support multiple eviction policies

2. Reusable

↓

Policy?

[illegible]


Strategy
LRU
last accessed time of a key
LFU
Access freq
FIFO
entry time
Map < K , value { } >

- Different types of policies
- Each policy requires some metadata
- Instead of info. in cache, can we use strategy classes instead?

LRU – last access time

LRU – last access time

[illegible] ( ) {

→ get ( ) {

→ update Time ( )

}

LFU → frequency

→ get ( ) {

update frequency ( )

}

}

FIFO → entry time

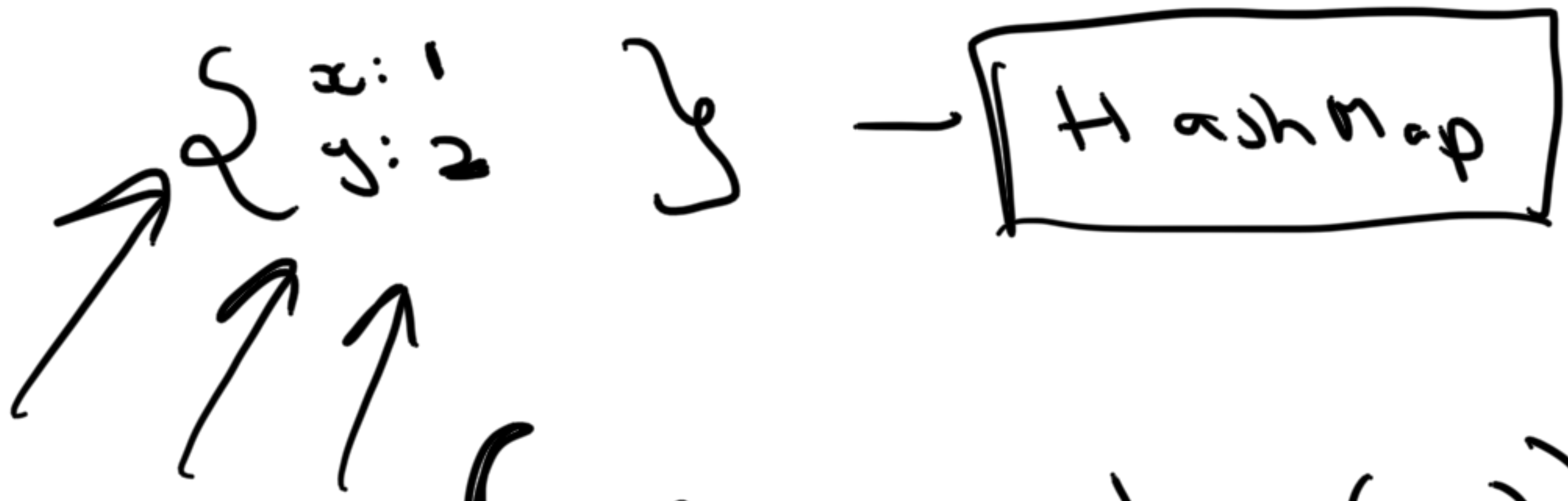
```
create () {
    update entry time
}
```

---

① whenever Create → I want to evict on access time

② Get → frequency

Aomic Data types



(map.contains(x))

Hashtable $\rightarrow$ slow

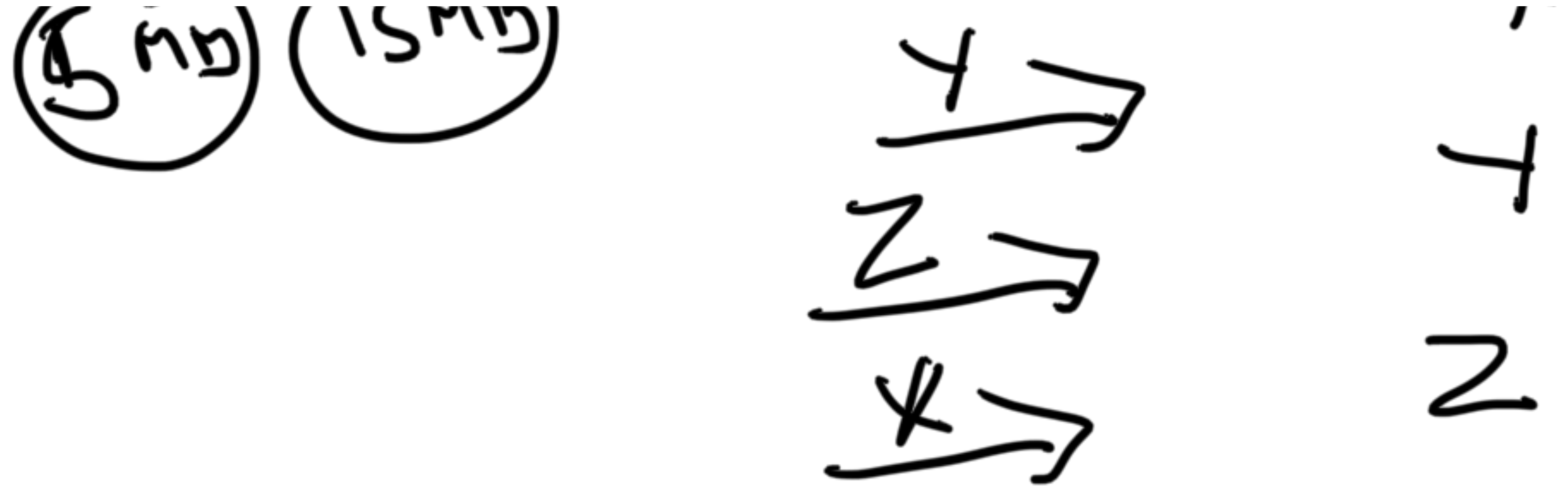
— Thread safe

— single threaded

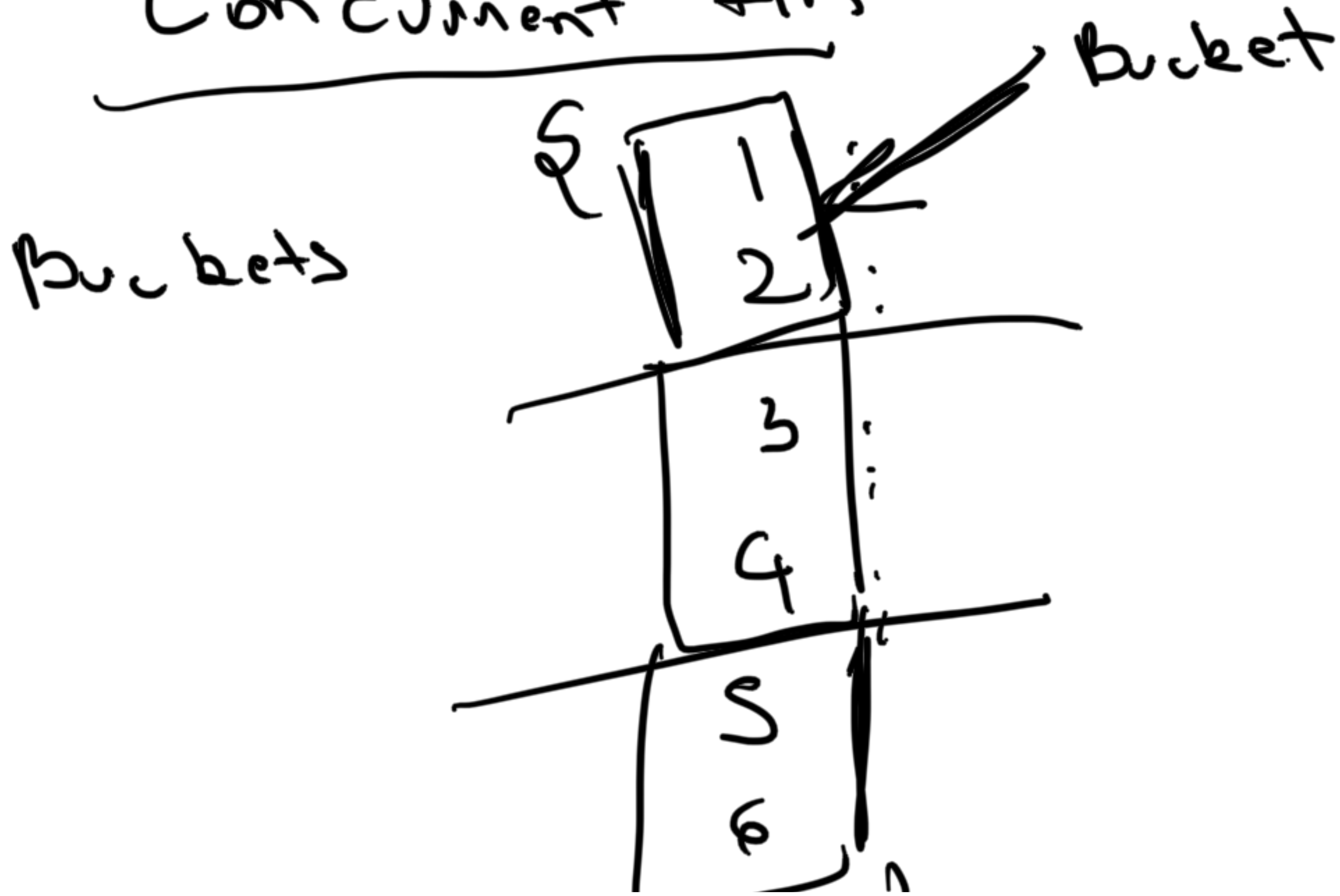
Hash Table

Concurrent Hashmap

20 Mb



Concurrent HM



HashMap → Not thread safe

Hashtable → Single threaded

CHM → multi threaded

→ Buckets