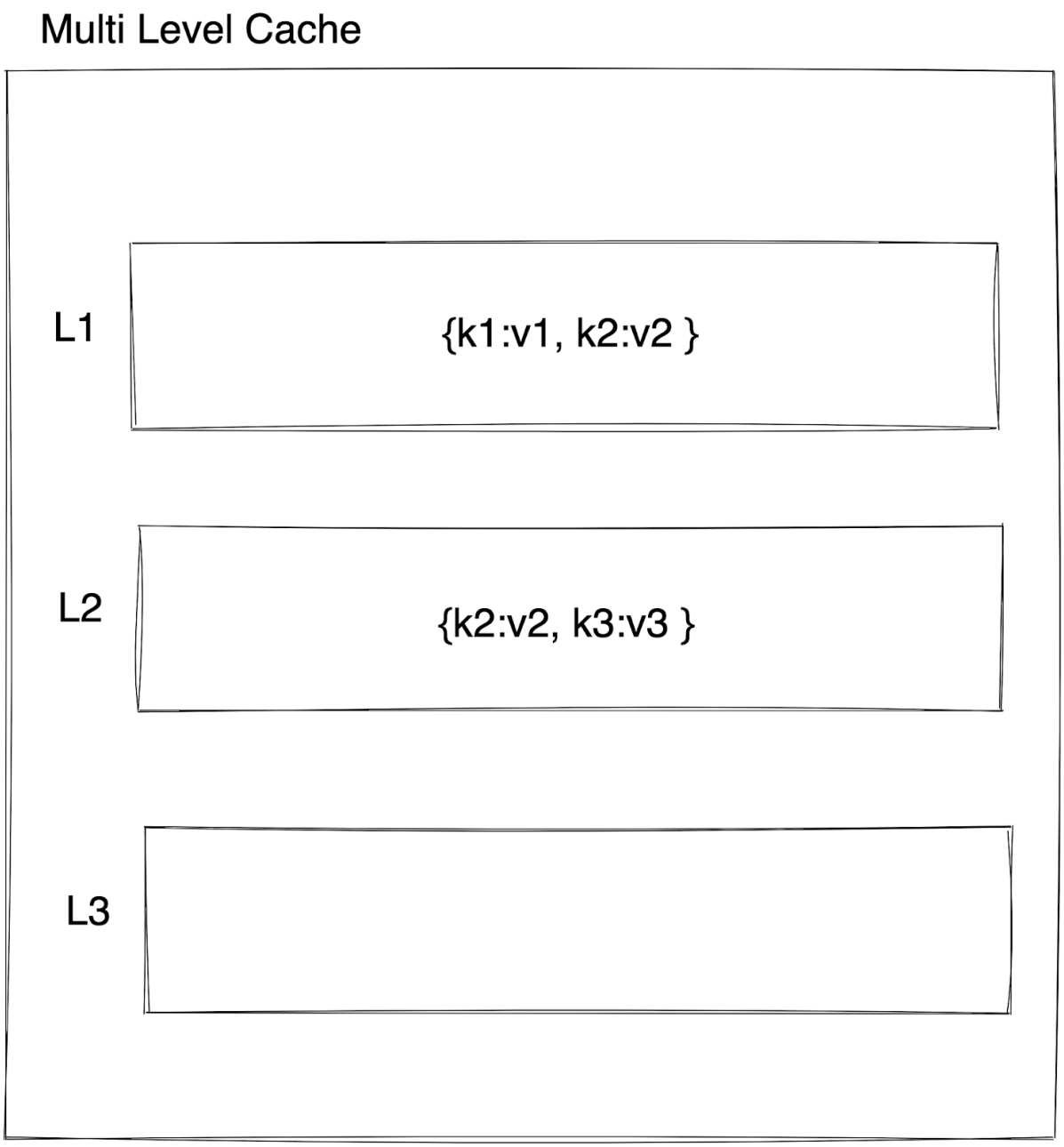
Design and implement an in-memory Multi Level Cache Management System with N levels, say (L1 -> L2 -> L3 … -> LN).



Each level will store Key-Value pairs of data. Both KEY and VALUE are of type String. L1 is the top most level and Ln is the last level.

You are given following configuration and details about the system:

1. Max number of levels of cache. The system starts with only 1 level initially and grows to a max number of levels as per the config.
2. The capacity of each level, i.e. number of elements that can be stored. The size is fixed for every level, although each level could have a different capacity.

This Cache system should be able to perform following operations:
READ KEY Operation

Read will start from L1 level. If KEY is found at this level, then this value will be returned. If KEY is not found at this level, then try to read from the next level. Keep doing this until the value of KEY is found at some level or the last level has been reached. If the value of KEY is found at some level, then this VALUE should also be written into the first level and the expected write behavior should follow as specified in the next section.

WRITE KEY Operation

Writes will start from level L1. If a level is full, evict any key-value pair from this level and write the evicted k-v pair into the next level. If the next level is also full, evict any k-v pair from that level and insert in the next level and so on. If you reach the last level, add a new level and insert the evicted K-V pair in this new level.

DELETE KEY operation

Deletes the given key from all the levels the key is present in.

Eviction Strategy: We need the strategy to be pluggable. For this problem, implement LFU (Least Frequently Used) i.e remove the key which has been used least frequently.

You will be evaluated based on

- Functional coverage
- Code modularity, readability, exception handling
- Separation of concerns
- Clean Abstractions
- Testability
- Application of OO design principles
- Taking care of any concurrency issues

- **[execution time limit] 3 seconds (java)**