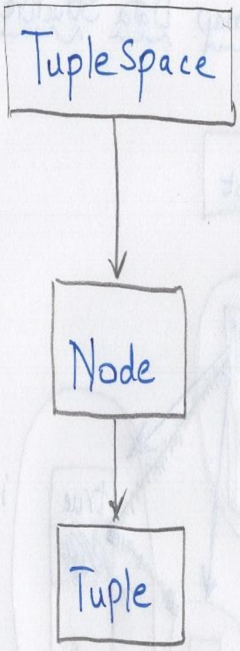
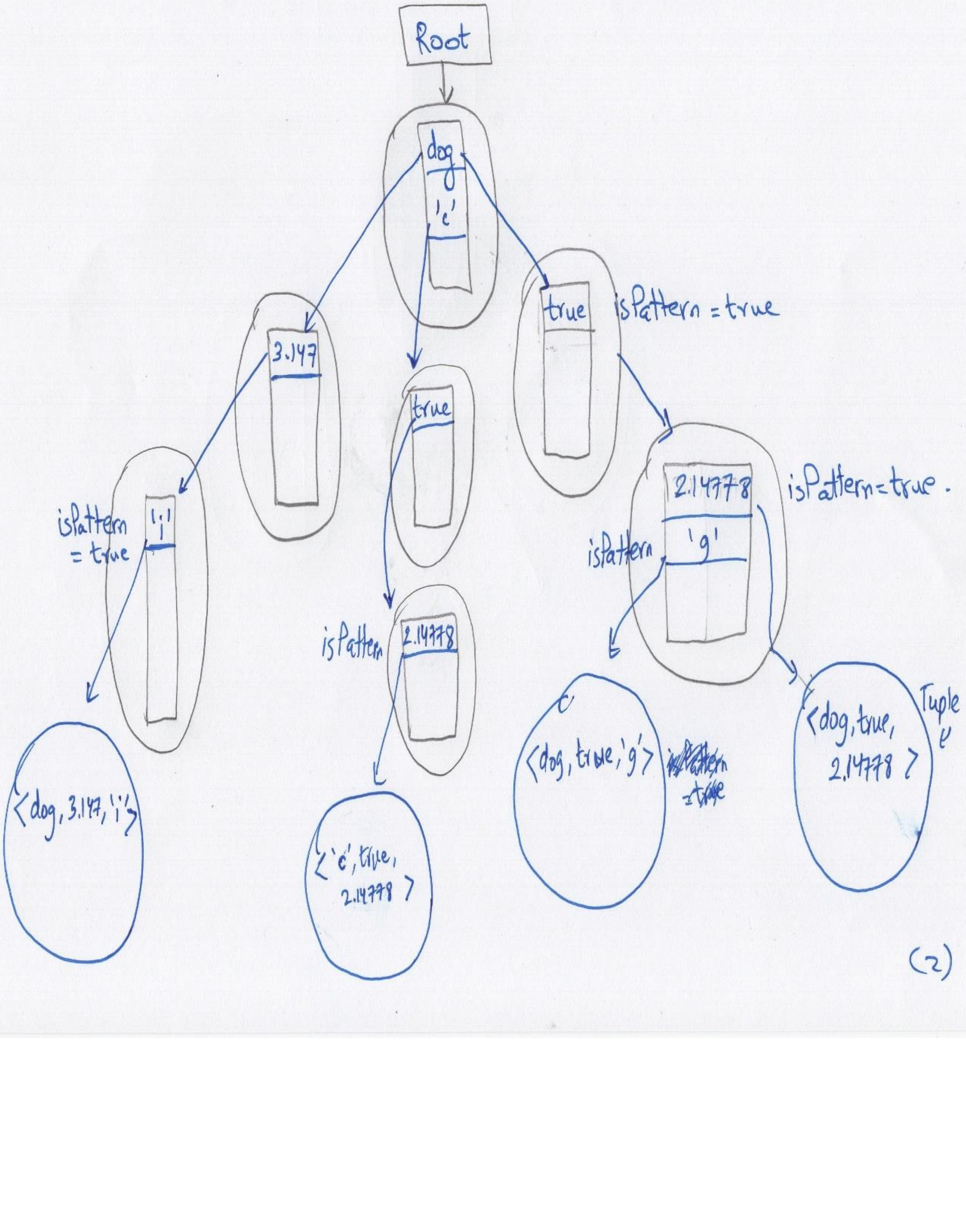
Tuple Space Design (explained on the next slide):



Design Details :-

Diagram to illustrate the efficient version of TupleSpace (TrieSpace):-

* Using Trie and hashmap Data structure :-



Stress Testing :-

- Have two implementations ~~of~~ of TupleSpace :-
  1) Trie and Hashmap. (called TrieSpace).
  2) Linked List. (called LinkedSpace).

1) Trie and Hashmap :-

- Quickly adds/search/remove the tuples of any size.
- Recursively does the operations.
- Using Trie and Hashmap data structure so that we can quickly find a specific object in the map with given node.
- Why better?
  - It's better because unlike LinkedList version, it doesn't do a linear search of all sizes tuple, it quickly traverses the tree and finds a matching tuple.

- This reduces runtime to $O(\log_2 n)$, on the other hand, the linkedList version has a runtime of $O(n^2)$.

2) Linked List :-

- Everything is shoved into a linkedlist without taking care of the size of each tuple.
- Linear search is done to look the pattern.
- Why bad?
    - It's bad because of linear search, if we have millions of tuples in the list, then it will take hours, if not days.

- Results from Stress Testing :-
    - Elegant Implementation :-
        - It takes about 50 sec to 1 min to add 100,000 tuples, search/remove 60,000 tuples together (including some wildcards).

- Naive Implementation :-
  - It takes about 3.6 mins to add/search/remove 7000 tuples, search/remove 7000 tuples together(including some wildcards).

- Check Yourself ?
  - In order to check/switch from efficient implementation to naive one, on line 17 in StressTest.java,

    Change TrieSpace trie = new TrieSpace();

    to

    LinkedSpace space = new LinkedSpace();

  - WARNING:

    Make sure to reduce the loop to a significant amount (at least 6000 - 7000) so, you can see it takes minutes.