Class HashTable

Github link: https://github.com/dacianf/FLDC

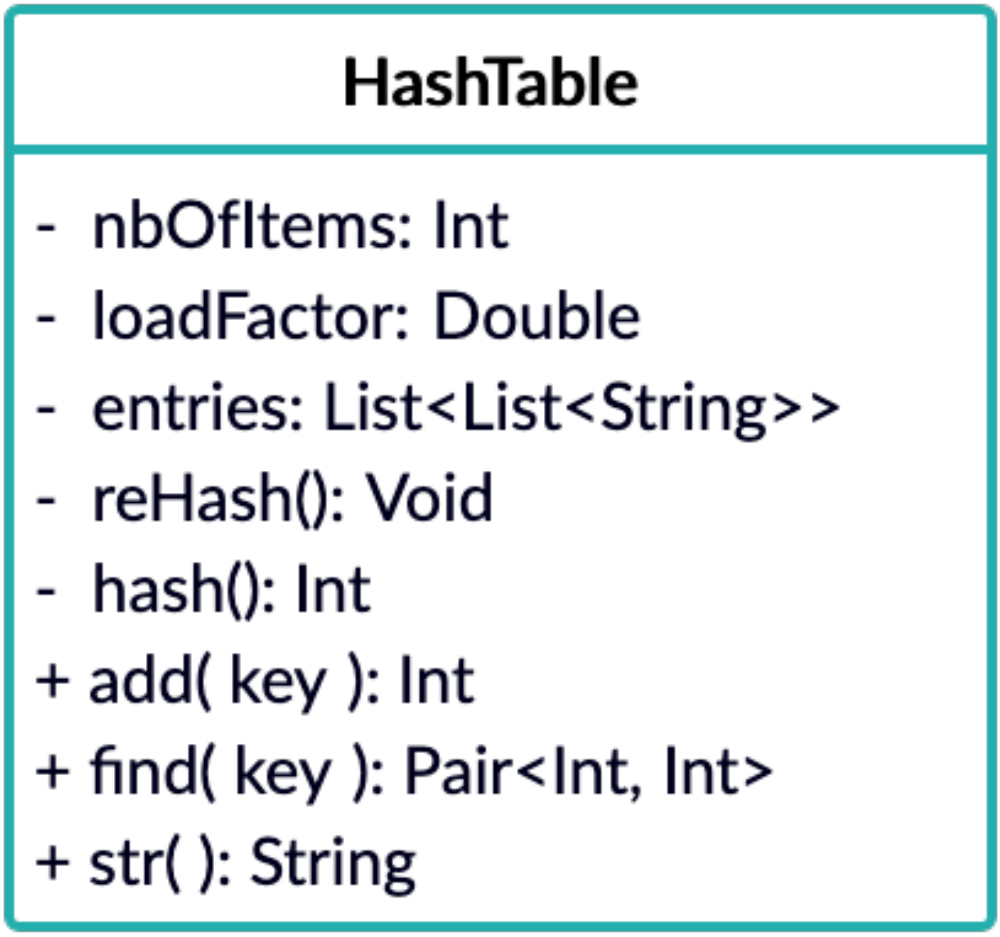
Implementation details:

The hash table is implemented using coalesced chaining.
It consists of a list of lists where the keys are hashes of the given inputs.
The hash table saves both the capacity and current size of the lists so it can be resized when the size exceeds the load factor which is (0.7 * currentSize).
The hash function used is:
hash(key) = keyHash(key) % capacity

Methods:
* public findPosition(String key)
- PRE:
key - a valid squence of characters
- POST:
-
- RETURN:
(hash, position) - if the key is found, a tuple with the hash of the key and the position in the list
(hash, -1) - otherwise

* public add(String key)
- PRE:
key - a valid squence of characters
- POST:
if the key was not already in the hashTable it will be added
- RETURN:
(hash, position) - a tuple with the hash of the key and its position in the list

* public getElement(Integer hash, Integer position)
-PRE:
hash - a valid integer which represents the hash of the item searched
position - a valid integer which represents the position of the searched item at the corresponding hash
-POST:
-
-RETURN:
None - if the there is no key at the input position
String - the key found at the input position


HashTable
- nbOfItems: Int
- loadFactor: Double
- entries: List<List<String>>
- reHash(): Void
- hash(): Int
+ add(key): Int
+ find(key): Pair<Int, Int>
+ str( ): String