Assignment 10

① Explain hashing. What is hash collision & how is hash collision handled by Java?

→ hashing: Hashing is the solution that can be used in almost all such situations & performs extremely well compared to ds like Array, Linked list, Balanced BST in practice.

(a) O(1) search time on average

(b) O(n) in worst case.

defn: In hashing there is a hash func/ that maps keys to some values. But these hashing func/ may lead to collision that is two or more keys are mapped to same value.

Hash collision: A hash collision occurs when a hash algorithm produces the same hash value for two diff/ input values.

→ An alternative method for handling the collision problem is to allow each slot to hold a reference to a collection (or chain) of items.

Chaining: allows many items to exist at the same location in the hash table. When collision happens the item is still placed in the proper slot of the hash table.

(2) Explain how hashing improves performance? How does it provide best time complexity?

→ hashing is the solutions & perform extremely well compared to data structure, like Array, Linked List, Balanced BST in practice.

(i) With hashing we get O(1) seach time on average & O(n) in worst case.

(ii) Hashing is an improvement over direct Access table. the idea is to use hash func/ that converts key to a smaller number & use the small no/ as index in a table called hash table.

(3) what will happen if we write hashcode() method to return a constant integer, say 10? will it work? what is the impact on performance.

→ If the equals method is implemented as per the contract & the hashcode method returns a constant value, then we will still be able to retrieve the value for the key from a hashmap, but the performance will be slow compared to the method returning a unique hashcode.

→ this is bec/ internally all the keys will be stored in the same bucket, & the hashmap implementation needs to verify the equality against all the keys present in the hashmap.

④ difference b/w HashSet, TreeSet, LinkedHashSet

HashSet	TreeSet	LinkedHashSet
1) HashSet uses HashMap internally to store its elements.	① It uses TreeMap internally to store its elements	① It uses LinkedHashMap internally to store its elements
2) It doesn't maintain any order of elements	② Order the elements according to supplied comparator. If no comparator is supplied, elements will be placed in their natural ascending order.	② maintain insertion order of elements i.e Elements are placed as they are inserted.
3) It provides better performance compared to both.	③ give less performance than both as it has to sort the elements after each insertion & removal operations	③ performance is b/w both almost similar to HashSet but slightly slower bec/ it maintain LinkedList internally to maintain the insertion order of elements
4) O(1) is the Time complexity for insertion, removal & retrieval operations	④ O(log(n)) is the time complexity.	④ O(1) is the time complexity.
5) It uses equals() & hashcode() methods to compare the elements	⑤ It uses compare() & compareTo() method to compare the elem/	⑤ It uses equals() & hashcode() methods to compare the elements
⑥ HashSet allows maximum one null element	⑥ don't allow null element. If we try to insert null elem/ throws Exception.	⑥ It allows one null element.
⑦ less memory required.	⑦ More memory is required than HashSet	⑦ requires more memory than HashSet bec/ it have LinkedList ds.
⑧ use HashSet if you don't want to maintain any order of elements	⑧ use it if you want to sort the elements according to some comparator	⑧ use it if you want to maintain insertion order of elements.

(5) If a class doesn't implement Comparable & Comparator, can we use it in TreeSet? what exception it throw?

→ Yes, we can use TreeSet but only one element can be added., which does not implement Comparable bec/ it does not need to be compared with other elements.

(6) TreeSet requires elements to implement the Comparable interface if a custom comparator is not set.

Exception: ClassCastException is thrown if the specified object cannot be compared with the elements