Q Diff b/w stack & heap ?

(1) Java heap space is used throughout the application, but stack space is used for the methods — or methods - currently running.

(2) Heap contains all the ~~objects~~ objects are created, but stack contains any reference to those objects.

(3) Objects stored in the Heap can be accessed throughout the application. primitive local variables are only accessed the stack Memory block that contain their methods.

(4) Heap space exists as long as the application runs & is larger than stack, which is temporary but faster.

Q. What is Java HashMap? It is in java.util.

→ (1) Java HashMap class implements the Map interface which allows us to store key & value pair.
(2) Where key must be unique. If you try to insert the duplicate key, it will replace the element of the corresponding key.
(3) contains value based on key.
(4) HashMap contains only unique keys.
(5) It may have one null key & multiple null values.

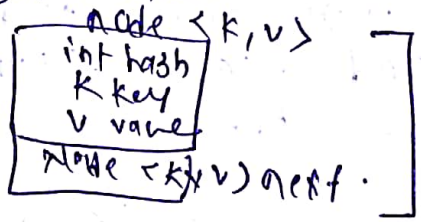
(4) It is not synchronized.
(5) It does not maintain the insertion order
(6) The initial default capacity is 16, & load factor 0.75 (getting doubled value)

Q. Working of HashMap?

It internally uses the technique of hashing.

* Hashing :- It is the process of converting an object into an integer value. The integer value help in indexing & faster searches.

HashMap contains the array of nodes, & the node is represented as a class. It uses an array & linked list data structure internally for storing key & value.



It uses the equals() & hashCode() method.

- equals() :- It checks the equality of two objects. It compare the key, whether key is equal or not. It is method of Object class. It can be overridden.
  * If you override the equals() method, then it is mandatory to override the hashCode() Method.
* hashCode() :- It is object class method. It returns the memory reference of object in integer form. The value received from the method is used as the bucket number. The bucket number is the address of the element inside the map. HashCode() for null key is 0.

Buckets:- Array of the node is called as buckets. Each node has a data structure like LinkedList

* map.put(key, value);

Calculating the Index.

index = hashCode(Key) & (n-1). ← size of array.

eg. Index = 265932 & (16-1) = 4;;

The value "4" is the index value, where the key & value will store in HashMap.

* **HashCollision:-** If for multiple object, If index value is same for two or more keys, In this case, Equals() method check that both keys are equal or not, If keys are ~~not equal~~ same, & replace with current value, otherwise Connect this node object to the existing node object through LinkedList. Hence both are at '4'.

map.put(key);

Q. Difference b/w HashTable & HashMap.

- HashTable & HashMap both are used to store data in key & value form. Both are using hashing technique to store unique keys.

HashMap.	HashTable
(1) It is non synchronized.	(1) It is synchronized. (object level lock)
(2) It is not thread safe.	(2) It is thread safe.
(3) HashMap allow one null key & multiple null values.	(3) HashTable doesn't allow null key or null values.
(4) It is fast	(4) It is slow.
(5) We can make it synchronized, by Map m = Collections.synchronizedMap(hashMap)	(5) It can't be unsynchronized
(6) Unique key & duplicate values allowed.	(6) Unique key & duplicate value are allowed.
(7) It is traversed by Iterator	(7) It is traversed by Enumerator & Iterator.

Q. Diff b/w LinkedList & Array List

ArrayList	LinkedList
(1) It implements List	(1) It implements List.
(2) It is for Dynamic. implements by Double-tial	(2) It implements Dynamic Array
(2) It uses the Dynamic array implementation.	(2) It uses the Double - Linked List.
(3) Initial capacity 10. & increase by 50%.	(3)
(4) If you add/remove element in b/w we have to shift all element (slow) O(n)	(4) If you add Element b/w (+ve) we don't need to shift all elements. (Fast)

(3) For seaching It is fast	(5) For seaching It is slow.

Q. ArrayList vs Vector.

ArrayList	Vector
(1) It is not-synchronized	(1) It is synchronized.
(2) It increment by 50 %	(2) It increment by 100 %.
(3) It is not Legacy class. It is introduced in JDK 1.2	(3) Vector is legacy class.
(4) It is fast	(4) It is slow.
(5) It uses Iterator	(5) It used Enumeration.

* HashSet *

- HashSet class is used to ~~create~~ create the collection that uses a hashTable for storage. It inherit AbstractSet class & Set interface.

(1) HashSet stores the elements by using a mechanism called hashing.
(2) HashSet contains unique element only.
(3) HashSet ~~allow null value~~. permit one (p.T.o). null element.

④ Hashset class is non-synchronized.
⑤ It does not maintain insertion order. Here, the elements are inserted on the basis of their hashcode.
⑥ Initial capacity is 16. L·F - 0·75

* When u ~~store~~ store the objects into set, to avoid duplicate u have to override the hashcode() & equals() method from object class.

* For Treeset (sorting) (doesn't allow null value).
  - when you store "object" in Treeset it gives an exception. (~~Comparable~~ class. lang. Comparable)
  ∴ bcoz Treeset elements are sorted & unique. for that.
    ① Natural ordering. java. lang. Comparable.
    ② java. util. Comparator. (I)

class — implements comparable.
override) int compareTo (Object o)
(Highest priority).

class — implements comparator.
override) int compare (object o1, object o2)

(maintain Ascending orders)

for Descending.
Navigable set<Integer> des = set. descendingSet();

ANJUMAN-I-ISLAM'S
KALSEKAR TECHNICAL CAMPUS, NEW PANVEL

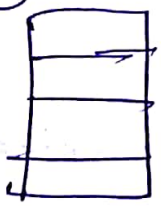
DATE:

	Iterator	ListIterator
1	Iterator is used for traversing List & Set both	It is used only for traverse List.
2	Traverse in only forward Direction.	Traverse in both direction (forward & backward).
3	We cannot obtain the index while traversing a list.	We can obtain the index while traversing.
4	You cannot add element to collection while traversing using iterator throws ConcurrentModificationException.	We can add element at any point of time while traversing a list using ListIterator.
5	We cannot replace existing element.	By using set(E e) method we can replace the last element returned by next() or previous().
6	hasNext(); next(); remove()	add(E e), hasNext(), hasPrevious(), next(), nextIndex(), previous(), previousIndex(); remove(), set(E e)

* ConcurrentHashMap *

- The ConcurrentHashMap class of the Java collections framework provides a thread-safe map. That is, multiple threads can access the map at once without affecting the consistency of entries in a map.

* By default capacity of map will be 16, LF = 0.75
* provides method for bulk operations like forEach(), search() & reduce()
* By default, the ConcurrentHashMap is divided, 16 segments. This is the reason why 16 threads are allowed to concurrently modify the map at same time. However, any number of threads can access the map at a time.
* It provides its own synchronization.
* segment or bucket level locking.
* so, Performance is good.
* 16 → concurrency level (default)
* read operation thread don't required any lock
* But update require lock, which is segment level or bucket level.



C++	Java.
(1) Platform-dependent	(1) platform-independent.
(2) Mainly used for system programming	(2) Mainly used for application programming. It is widely used in window, web-based, mobile, application.
(3) It supports multiple Inheritance.	(3) It does not support multiple inheritance through class, It can achived by Interfaces.
(4) operator overloading	(4) Not support
(5) You can write pointer	(5) Internally uses pointer but you cannot write pointer.
(6) C++ support structure & unions.	(6) Don't support structure & unions.
(7) Virtual keyword is supp so we can decide whether or not to override a funcn.	(7) Don't support virtual keyword. We can override all non-static methods by defult.