This question is asking in most of the java interviews. Let's see the story of HashMap.

How HashMap works in Java?

It works based on hashing.

Hashing is nothing but, Creating and Assigning a unique code to the object.

There is a method called "hash (int h) "in HashMap class to do this.

From Java API:

```
/**
     * Applies a supplemental hash function to a giv
     * defends against poor quality hash functions.
     * because HashMap uses power-of-two length hash
     * otherwise encounter collisions for hashCodes
     * in lower bits. Note: Null keys always map to
     */
    staticint hash(int h) {
        // This function ensures that hashCodes that
        // constant multiples at each bit position h
        // number of collisions (approximately 8 at
        h ^= (h >>> 20) ^ (h >>> 12);
        return h ^ (h >>> 7) ^ (h >>> 4);
    }
```

Entry Interface: A key-value pair is called one Entry. This interface is included in Map interface.

Entry class: HashMap holds Entry class in which it implements Map.Entry.

From HashMap class:

```
staticclass Entry<K,V> implements Map.Entry<K,V> {
        final K key;
        V value;
        Entry<K,V> next;
        finalinthash;

        /**
         * Creates new entry.
         */
        Entry (int h, K k, V v, Entry<K,V> n) {
            value = v;
            next = n;
            key = k;
            hash = h;
        }
```

Let's see HashMap's put method:

From Java API:

```
/**
     * Associates the specified value with the spec
     * If the map previously contained a mapping fo
     * value is replaced.
     *
     * @param key key with which the specified valu
     * @param value value to be associated with the
     * @return the previous value associated with <
     *         <tt>null</tt> if there was no mappin
     *         (A <tt>null</tt> return can also ind
     *         previously associated <tt>null</tt>
```

```
     */
    public V put(K key, V value) {
        if (key == null)
            return putForNullKey(value);
        int hash = hash(key.hashCode());
        int i = indexFor(hash, table.length);
        for (Entry<K,V> e = table[i]; e != null; e =
            Object k;
            if (e.hash == hash && ((k = e.key) == ke
                V oldValue = e.value;
value = value;
                e.recordAccess(this);
                return oldValue;
            }
        }
        modCount++;
        addEntry(hash, key, value, i);
        returnnull;
    }
```

- HashMap allows only one Null key. First it checks if key is null, store the null in table [0]. Null key always stored in first position of Hash map.
- Next it calculates key's hashcode by using hashCode() method and find the hash value. The hash value is used to find index for the array to store entry object.
- indexFor(hash,table.length) , will find an index/position to store an Entry object.
- Before going to further discussion, we should know about the hashCode() and equal() contraditction in Java.

If two objects are equals by equals () method, then their hashcode should be same.

If two objects are not equals by equals () method, then their hashcode can be same or different.

- So, now we got the index to store the Entry object (Which is nothing but key-value object).

And we know that two objects which are not equal by equals () method can have same hash code. So we have two objects but only one location to store both the objects? So how it will store? HashMap stores these in the same location. That is called Bucket. If you see Entry class, we have next attribute (Entry<K, V> next ;). So the two objects will be stored in the same index/same bucket. (Kind of Linked List form.)

- Now what happens for below case

map.put("Yuvaraj", "70");

map.put("Yuvaraj", "68");

First 70 will be stored for the key "Yuvvraj".

If you add new value to the same key, it replaces old value with the new value.

So the value for "Yuvaraj" key is 68.

(See if loop in put method to know how it replaces).

This is the way hash map ensures the uniqueness of keys.

Get method of HashMap from Java API:

```
public V get(Object key) {
        if (key == null)
            return getForNullKey();
        int hash = hash(key.hashCode());
        for (Entry<K,V> e = table[indexFor(hash, tab
             e != null;
             e = e.next) {
            Object k;
            if (e.hash == hash && ((k = e.key) == ke
                return e.value;
        }
        returnnull;
```

```
    }
```

We already know how put method works and how it maintains the uniqueness of keys.

Once you ask the value based on key, it checks the equality of the key and returns the value of that key.

Notes:

- Data structure of HashMap is an Entry class array . transient Entry[] table;
- Bucket: Particular index location/position in Entry array
- Key object's hashCode() has to be calculated, to find the index of an Entry object.
- Key object's equals () method is used to enable the uniqueness of key.