1. Advanced version of Hashtable
2. Allows multiple null values and one null key
3. It statys non-synchronized .It is not very thread safe.(User cant share between various threads)
4. Traversed using Iterator
5. Very fast due to non-synchronisation
6. Iterator present in Hashmap is fail-fast.
7. HashMap extends AbstractMap and implements Map Interface
8. HashMap is a key-value collection in java
9. Uses hashtable to store the map to make the get and put methods remain the same

**Performance Changes in Java 8**

1. Performance was bad due to hash collision – when multiple keys end up in same bucket – the values along the keys are placed in the linked list
2. So retrieval time of elements from HashMap increases from O(1) to O (n)
3. Once the threshold value is reached, the hash will change from using linked list to binary tree.
4. Performance of O(log n) in worst case and O(1) with proper hashes
5. Alternate Sting hash function added in Java 7 is been removed.

How hashmap works internally ?

1. Hashmap maintains the array of buckets
2. Hashmap calculates index for each and every operation.
3. Key object is used to calculate the index value.
4. Hash value is calculated using hash(key) private method in Hashmap
5. Index= hash(key) & (n-1 ) to ensure the index value fals between 0 and n-1 .

How put operation works

1. Key object is checked for null and hashcode for null key is 0.
2. Hash value is calculated using the key’s hashcode by calling its hashcode()
3. Hash value is used to calculate the index in the array for storing the entry objects
4. Hashcode() method may return high or low hash value . So JDK designers introduced hash() to bring the hash value between 0 and n-1
5. indexFor (hash, table.length) is called to calculate the exact indes position for storing the entry object.

How Collision is resolved

Two unequal objects can have same hash code value

How two different object will be stored in the same array location – called bucket

Entry calss always has a attribute “next” -> always points to the next Oject

If the Object is sitting on calculated index , then next attribute is checked . If its null ,then place the value. This process is carried out till it finds the next value as null

If we add another value object with same key, it will replace the existing key.

Hashmap calls equals() for each entry object .

Key.equals(k) is retruned true in case of both the keys are streated as same key object.

Permits null

Hashmap uses the hashcode() to find the appropirate bucket first and if it finds more than one entry in bucket then it used equals method.

If you are not defining equals method in your class as per your requirment than it will use the defination from parent class Object, which is simple "==" operation.

It is always preferable to override hashcode and equals method if you are using your class as key in hashmap.

Why order is not mainted in Hashmap