**Collection Framework**

Collection framework is use for store and manipulation the group of objects.

**Iterable Interface**

It is root interface of collection classes.

It contains only one abstract method - Iterator<T> iterator().

**Iterator Interface**

It will have functionality it iterat the element in Forword direction only.

It have next(), hasNest(), remove() and forEachRemaining().

**Collection Interface**

It is child interface of iterable interface.

It is used to represent the group of individual objects as a single unit.

It has some method like add(), addAll(), remove(), removeAll() and so on.

**Collections Class**

It is utility class which has some method that are used to operate on collections.

It contains only Static method.

Like Collections.max(List), Collections.min(List).

**List Interface**

* **ArrayList and LinkedList** are both the contains duplicate elements.
* Both are maintain insertion order.
* Both are non-synchronized.
* Now ArrayList uses Dynamic array to store the elments.
* And LinkedList uses Doubly LinkedList internally to store the elements.
* ArrayList is slower than Linked list because of lots of shifting is needed.
* LinkedList is faster than ArrayList.
* We need to use ArrayList when we want to read faster.
* And we need to use LinkedList when we want to addition and remove the elements.
* ArrayList – Read.
* LinkedList – Manipulation

**Queue interface – FIFO**

PriorityQueue

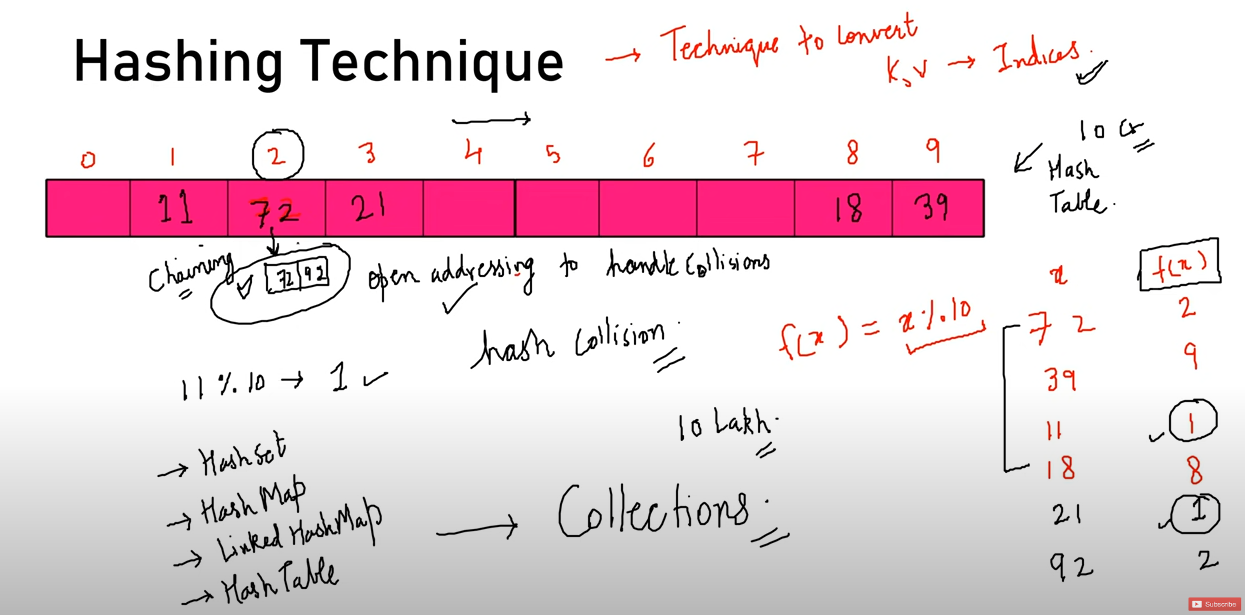
**Deque interface**

ArrayDeque

**Hashing Technique**

* It is technique to convert range key value pair into indices.
* **In simple word Hashing is the process of converting objects into its integer values which helps the indexing and faster searching.**
* **Also Hashing is nothing but algorithm or function that maps your object to some integer value.**
* It will calculate the value of element to store the elements in **hash table.**
* **HashTable** is like list of Array of Buckets.
* If same address will assign some more than one element then its mean hash collision happens.
* Prevent the hash collision it will store next empty address.
* There is all other technique in prevent hash collision which is chaining.
* Chaining mean it will use linked list to store this elements.

**Load Factor** is measure how full the hash table is allow to get before its capacity increase.



**When we should go for HashSet and LinkedHashSet?**

When we want go with insertion order then we will must go with LinkedHashSet.

**HashSet** – Not maintain insertion Order

**LinkedHashSet** – Maintained insertion Order

**TreeSet** – Maintained Ascending Order / Natural Order

**HashTable**

It is list of array also known as Buckets.

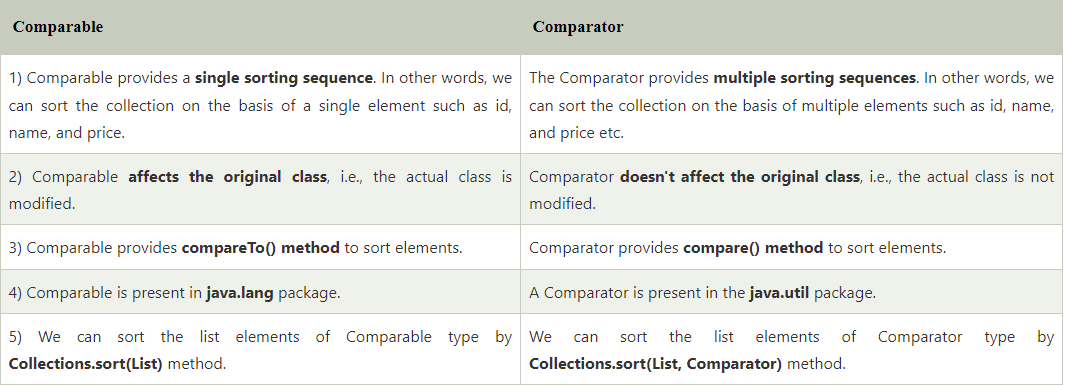
It is synchronized and not allow null key and value.

It contain unique elements.



**Comparable – Single sorting – CompareTo()**

**Comparator – Multiple Sorting – Compare()**



**Equals and HashCode Contarct**

If we override the equals() method, then it is mandatory to override the hashCode() method.

Now the contract is after we override this equals method then If two different object of equals() method will return true then this two object must have same hashCode.

For example

We have student class which having two fields id and name.

Both student object having same id but different name.

Now if we override the method then we will compaire with id and it will return true then

**MAP Interface**

It contains key – value pair

put(), putAll()

Map.Entry Interface

It subinterface of map.

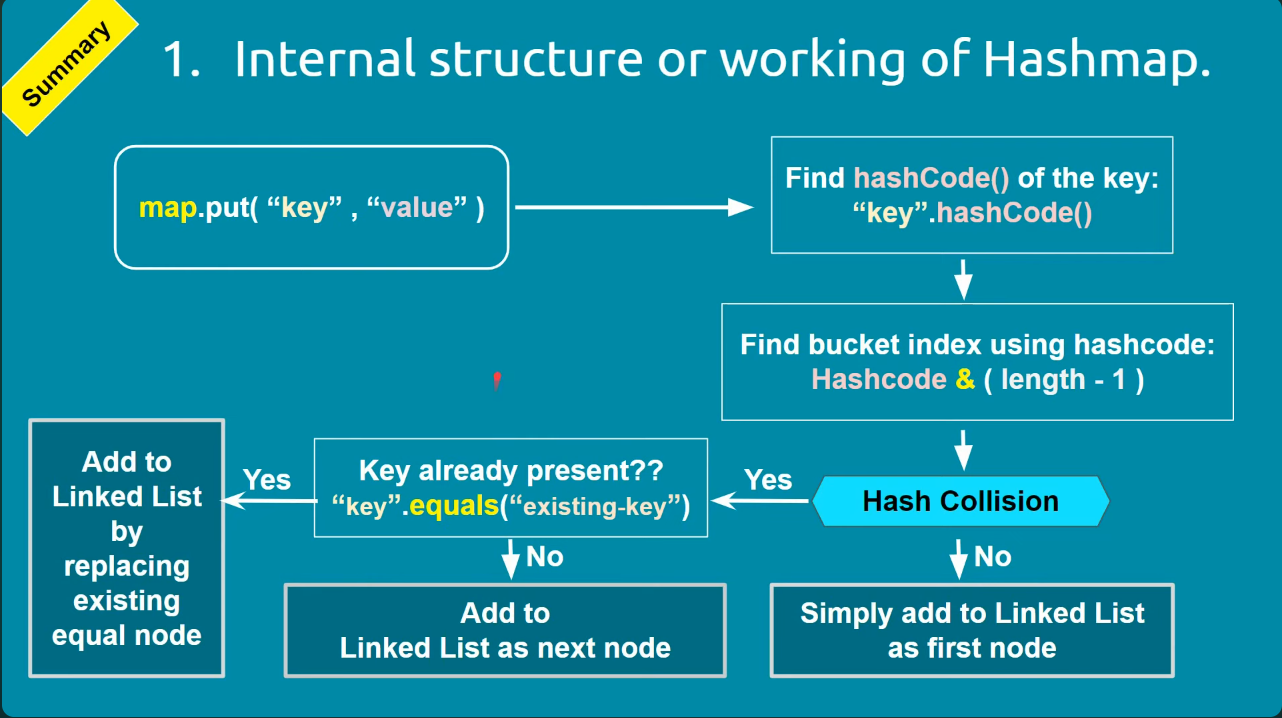
It will return collection view of the map.

It provide a method to get key and value like getKey(), getValue(), setValue(),

Map<Integer,String> map=new HashMap<>();  
map.put(100,"Amit");  
map.put(101,"Vijay");  
map.put(102,"Rahul");  
//Elements can traverse in any order  
for(Map.Entry m:map.entrySet()){  
 System.*out*.println(m.getKey()+" "+m.getValue());  
}

**HashMap (Key K, Value V)**

**Internal Working hashMap**



Internally use Hashing Technique.

**equals():**

It checks the equality of two objects.

It compares the Key, whether they are equal or not.

It is a method of the Object class. It can be overridden.

If you override the equals() method, then it is mandatory to override the hashCode() method.

**hashCode():**

This is the method of the object class.

It returns the memory reference of the 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. Hash code of null Key is 0.

**Buckets:**

Array of the node is called buckets.

Each node has a data structure like a LinkedList.

More than one node can share the same bucket.

It may be different in capacity.

* When we create Map Objet then it will create Bucket.
* Initial capacity of map is 16 then Default Bucket structure is 16 means 0 – 15.
* This all bucket is consider as a LikedList.
* This LinkedList contains Node.
* Node has a **Hash Key Value next**
* Now when call put(K, V) method it will calculate hashCode using key and then calculate bucket index value using hashcode & (length - 1) for this hasValue.
* Now it will put this **hashValue Key Value next** in calculated bucket index.
* Now next is null because no new node is insert in bucket yet.
* If next map object put method is call and this object calculated index of bucket is same as present one then hashing collision happens.
* Collision means if same bucket having multiple nodes.
* Now when Collision happens it will call internally equals() method.
* It will check and compare present node key and current node key is same or not using equals() method.
* If it is different then immediately it will store this current node in same bucket.
* Now present reference node next value having this new node reference.
* If Key as Null then it will go with zeroth bucket.

**What is enhancemet of HashMap in Java 8?**

This HashMap is use bucket as a LinkedList but in certain threshold it will convert as balance tree mechanism.

In Java 8 they use Balance tree mechanism instead of LinkedList in certain threshold.

**What is Fail-Fast and Fail-Safe?**

**Fail-fast** is like throw ConcurrentModificationException while the two threads are try to modifying simultaneously then it will throw the exception.

**Fail-safe** is like working on clone of its collection and it can’t throw any exception.

**Concurrent Collection**

CopyOnWriteArrayList – same as ArrayList but its thread safe.

CopyOnWriteArraySet – same as set but thread safe and it internally uses CopyOnWriteArrayList for operations.

**Why we need of concurrent collection in Java?**

If we try over traditional collection object operation like simultaneously run thread and try to modification on existing structure then it will throw concurrentModificationException.

Now we need operation in thread safe manner then we need to us concurrent collection.

**Why performance of concurrentHashMap is better than HashTable and SynchronizedMap?**

In traditional collection if any operation perform then lock is acquired on complete collection so only single thread can capture.

But in concurrency collection the lock will acquired only in bucket level.

**What is Concurrancy level in ConcurrentHashMap?**

Concurrency level defines how many threads are allowed to get Lock.

By default ConcurrencyHashMap has 16 buckets and also concurrency level has 16 values.

If concurrency level is 8 and bucket is 16 then each segment get lock on two buckets.

If thread1 and thread2 will working on different segment

Now if we want update segment of t1 with t2 it will not allowed

But read operation will work properly.

Why null not allowed in ConcurrentHashMap while allowed in hashMap?

Can concurrent modification exception occurs only in multithreaded environment?

No.

