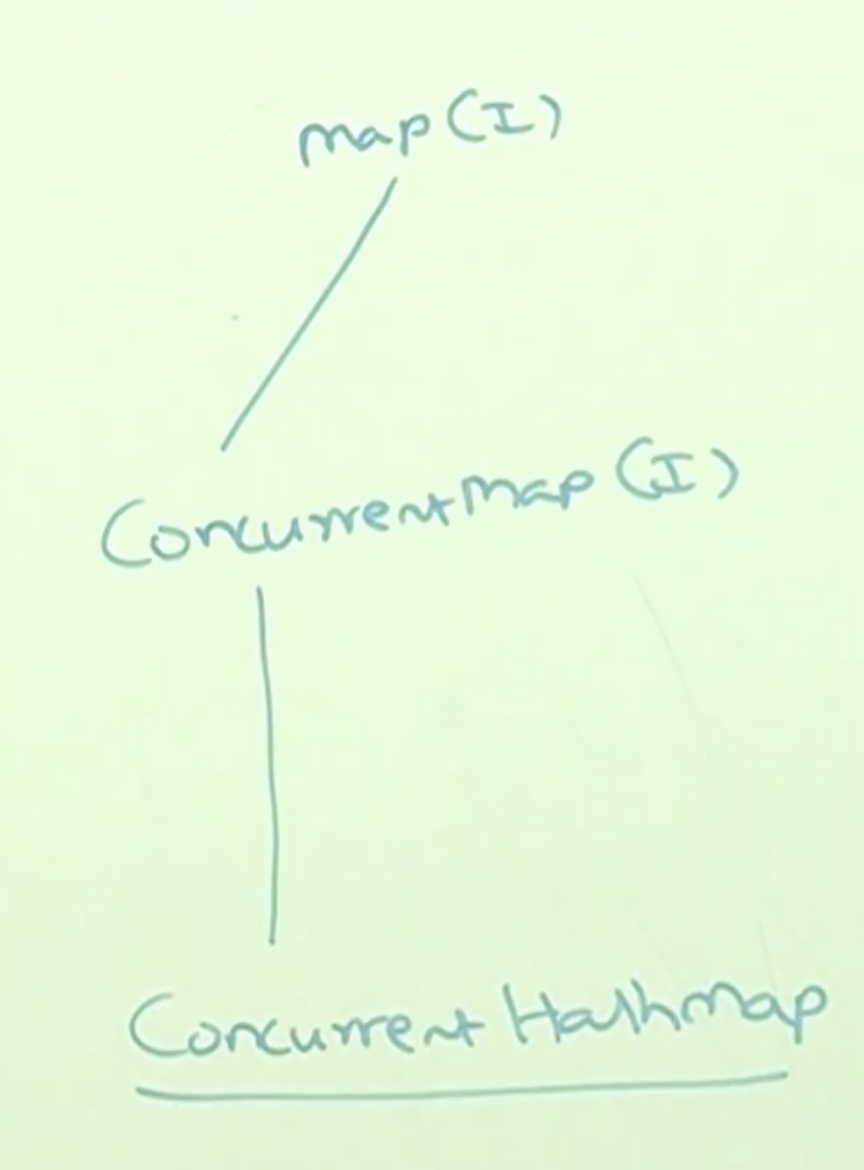
**Concurrent Collection– Part\_03**

* **ConcurrentMap(I):**

All concurrent interfaces and classes present in java.util.concurrent package.



Whatever the methods are there for Map(I), those will be available for ConcurrentHashMap also, adding to that ConcurrentMap has the following methods.

* **Methods of ConcurrentMap:**

putIfAbsent()

public Object putIfAbsent(Object key, Object value)

In the normal Map, duplicate keys are not allowed, when we try to put an Entry to the map with the already existing key, the Map will overwrite the value for the existing key.

In case of putIfAbsentMethod(), it will add the entry only the given key is not available, else it will not do anything.

Example:

Normal put method:

m.put(101, “durga”);

m.put(101, “shiva”);

System.out.println(m); // {101=shiva}

putIfAbsent()

m.put(101, “durga”);

m.putIfAbsent(101, “shiva”);

System.out.println(m); // {101=durga}

remove()

boolean remove(object key, Object value);

Normal remove() method has only one parameter which is Key, it will remove the entry based on the key. Whereas remove() method of ConcurrentMap has two parameters (key and value), for the given key if the value matches then it will remove the entry. Else it won’t do anything.

Example:

m.put(101, “durga”);

m.remove(101);

System.out.println(m); // {}

m.put(101, “durga”);

m.remove(101, “shiva”);

System.out.println(m); // {101=durga}

m.remove(101, “durga”);

System.out.println(m); // {}

replace()

boolean replace(Object key, Object oldValue, Object newValue);

This method replace the value, if the given key and value matches, else it won’t do anything.

m.put(101, “durga”);

m.remove(101, “shiva”, “ravi”);

System.out.println(m); // {101=durga}

m.replace(101, “durga”, “shiva”);

System.out.println(m); // {101=shiva}