CONCURRENT HASHMAP

(1) Internal working of concurrent hashmap ?

(a) Java 7: Segment-Based Locking

- The map is divided into 16 segments (default), each acting as a smaller hashmap.
- Only the segment being accessed is locked during write operations.
- Reads are generally lock-free unless a write is occurring on the same segment.

(b) Java 8: CAS-Based Approach

- No segmentation; instead, uses Compare-And-Swap (CAS) for most operations.
- CAS (Compare-And-Swap) is an atomic instruction that updates a value only if it matches the expected value.
- Locking is used only in rare cases (e.g., resizing or resolving collisions).
- CAS ensures atomic updates: a thread updates a value only if no other thread has modified it.

eg- Thread A last saw x = 45;
now Thread A work is x to 50;
if x is still 45, then change it to 50 else don't change and retry

- This approach enhances scalability and reduces contention.

Ex-

```java
public static void main(String[] args) {
    ConcurrentHashMap<Integer, String> map = new ConcurrentHashMap<>();

    // Add entries to the ConcurrentHashMap
    map.put(1, "Apple");
    map.put(2, "Banana");
    map.put(3, "Cherry");

    // Retrieve and print values
    System.out.println("Initial Map: " + map);

    // Update a value
    map.put(2, "Blueberry");
    System.out.println("After updating key 2: " + map);

    // Retrieve a specific value
    String value = map.get(3);
    System.out.println("Value at key 3: " + value);
    // Use putIfAbsent to add an entry only if the key is not present
    map.putIfAbsent(4, "Date");
    System.out.println("After putIfAbsent for key 4: " + map);

    // Remove an entry
    map.remove(1);
    System.out.println("After removing key 1: " + map);

    // Replace a value atomically
    boolean replaced = map.replace(2, "Blueberry", "Blackberry");
    System.out.println("Replace key 2's value conditionally: " + replaced);
    System.out.println("Map after replacement: " + map);

    // Iterate over the ConcurrentHashMap
    map.forEach((key, val) -> System.out.println("Key: " + key + ", Value: " +
val));
}
```