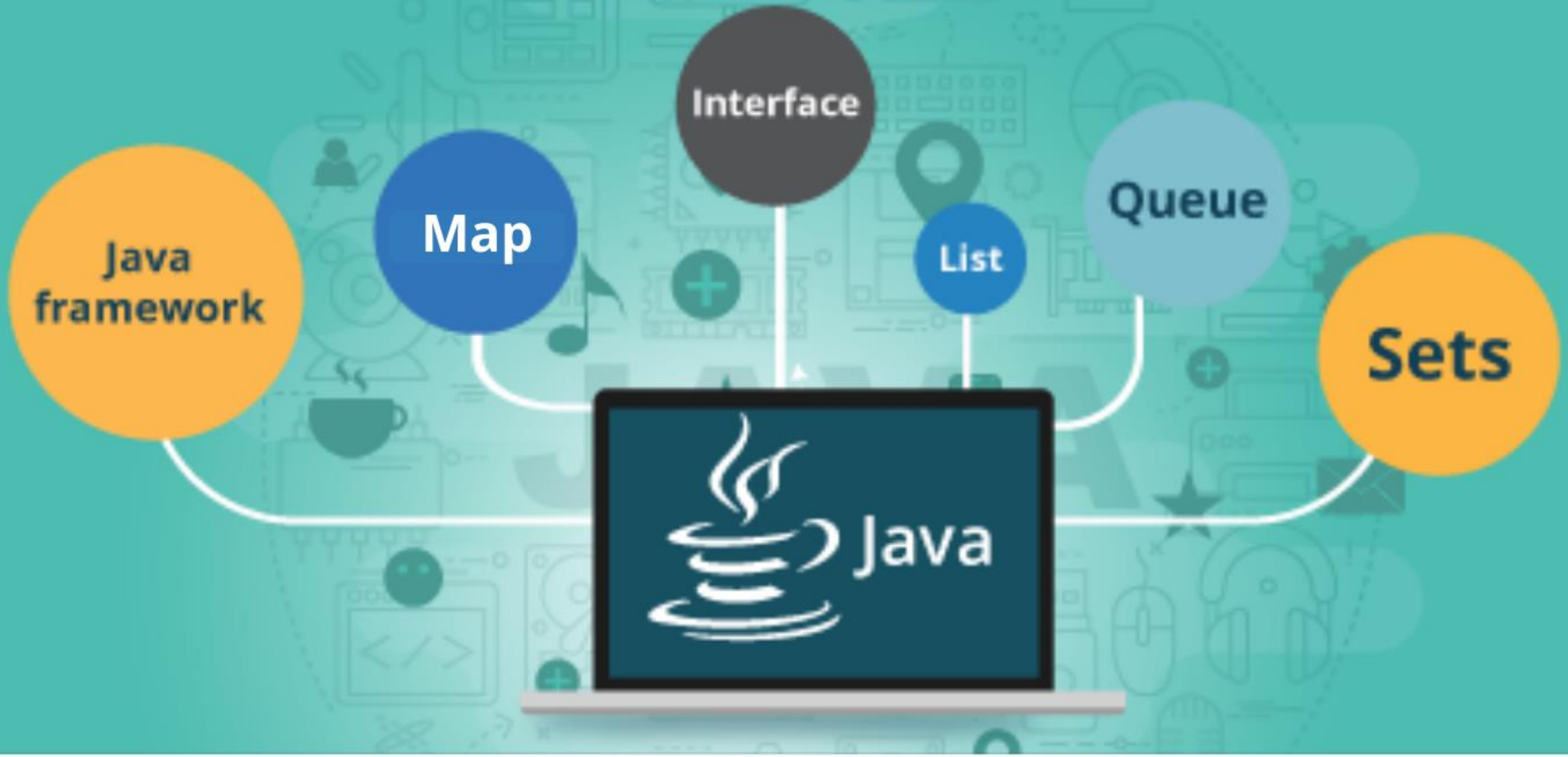
Tricky Questions & Answers

Java Techie



Collection Interview Q&A

Interview Q&A

1. what all collection you used in your project ?

List(I)	Set(I)	Map(I)
ArrayList	HashSet	HashMap
LinkedList	LinkedHashSet	LinkedHashMap
	TreeSet	TreeMap
CopyOnWriteArrayList	CopyOnWriteArraySet	ConcurrentHashMap

java.util.concurrent.*

Interview Q&A

2. What is the difference between list and set ?

3. What is the Difference between ArrayList and LinkedList ?

ArrayList	LinkedList
ArrayList internally uses a dynamic array to store the elements	LinkedList internally uses a doubly linked list to store the elements.
Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory	Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
ArrayList is better for storing and accessing data	LinkedList is better for manipulating data.

Interview Q&A

4. List object creation scenario

```
ArrayList arrayList=new ArrayList<String>();
List<String> list=new ArrayList<>();
```

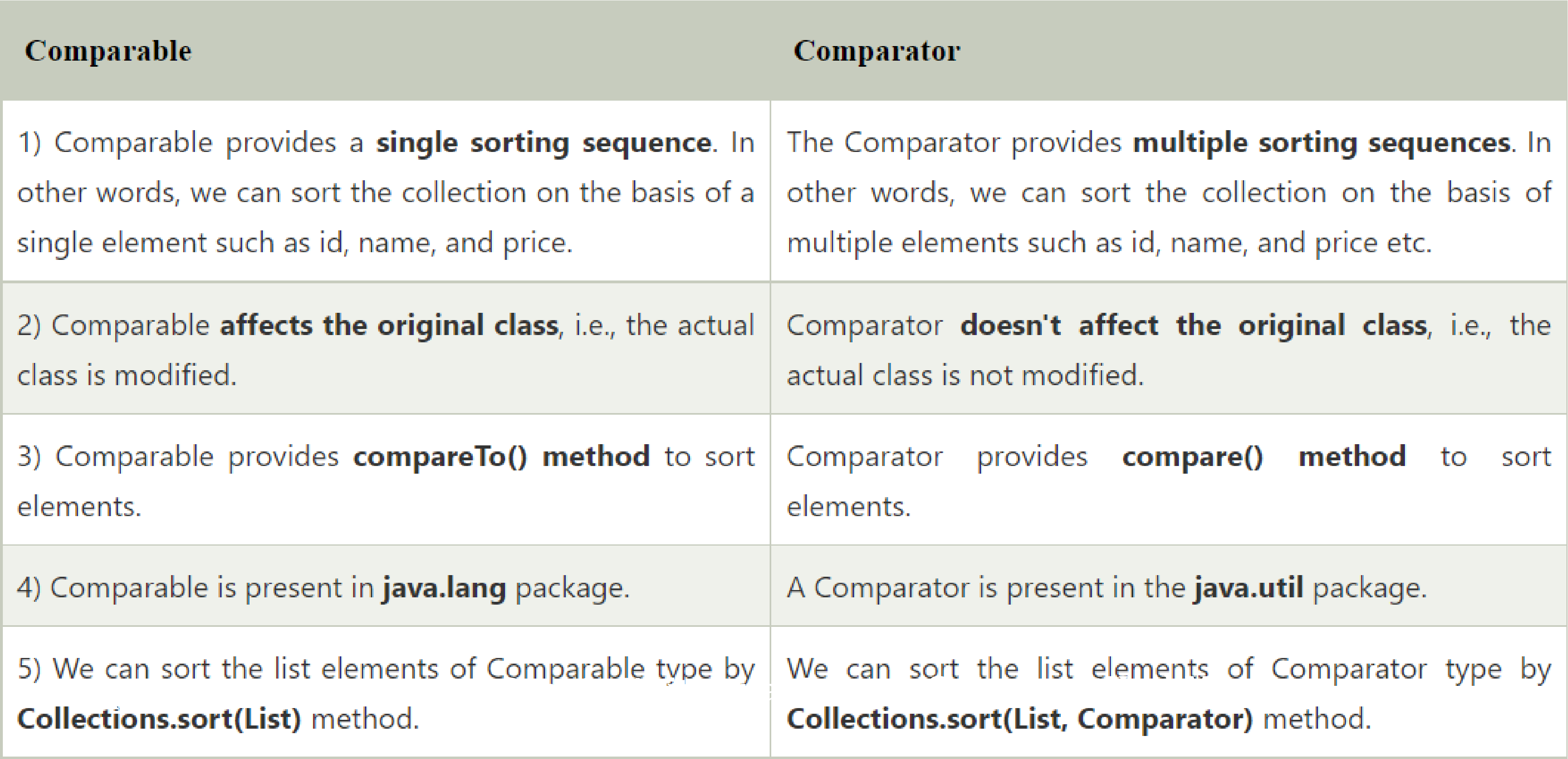
5. Declaring a List field with the final keyword ?

6. How Can I write Custom ArrayList where I don’t want to allow duplicate ?

7. Why Set doesn’t allow duplicate Element ?

8. What is the difference between Comparable and Comparator ?

Interview Q&A



Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

Interview Q&A

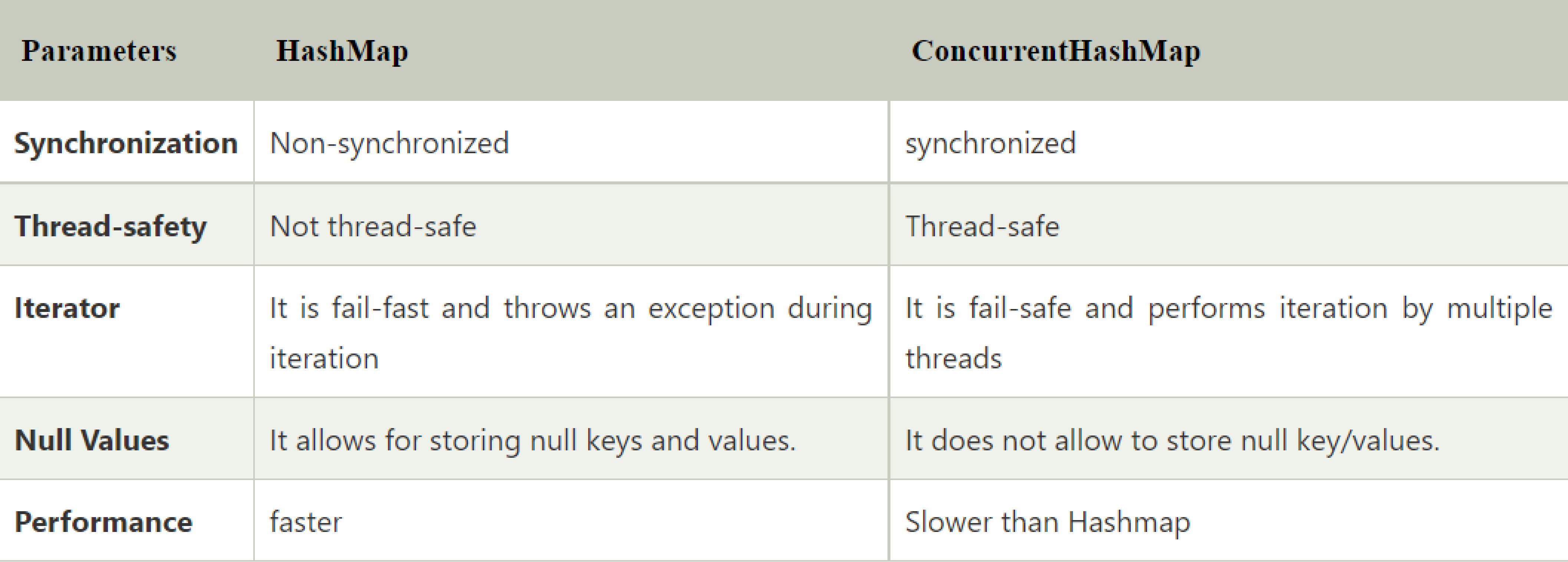
9. Multi Comparing using Comparator Scenario

10. What is the difference between fail fast and fail safe iterator

- A iterator which will fail fast when we do any modification while iterating a collection is called fail fast iterator Ex : (ArrayList ,HashMap and Vector)
- Iterator who allow us to modify in middle while iterating a collection is called Non-Fail Fast Iterator Ex: (CopyOnWriteArrayList , CopyOnWriteArraySet, ConcurrentHashMap)
- Example
- Internal Flow

Interview Q&A

11. What is the need of ConcurrentHashMap and How it is different from HashMap ?

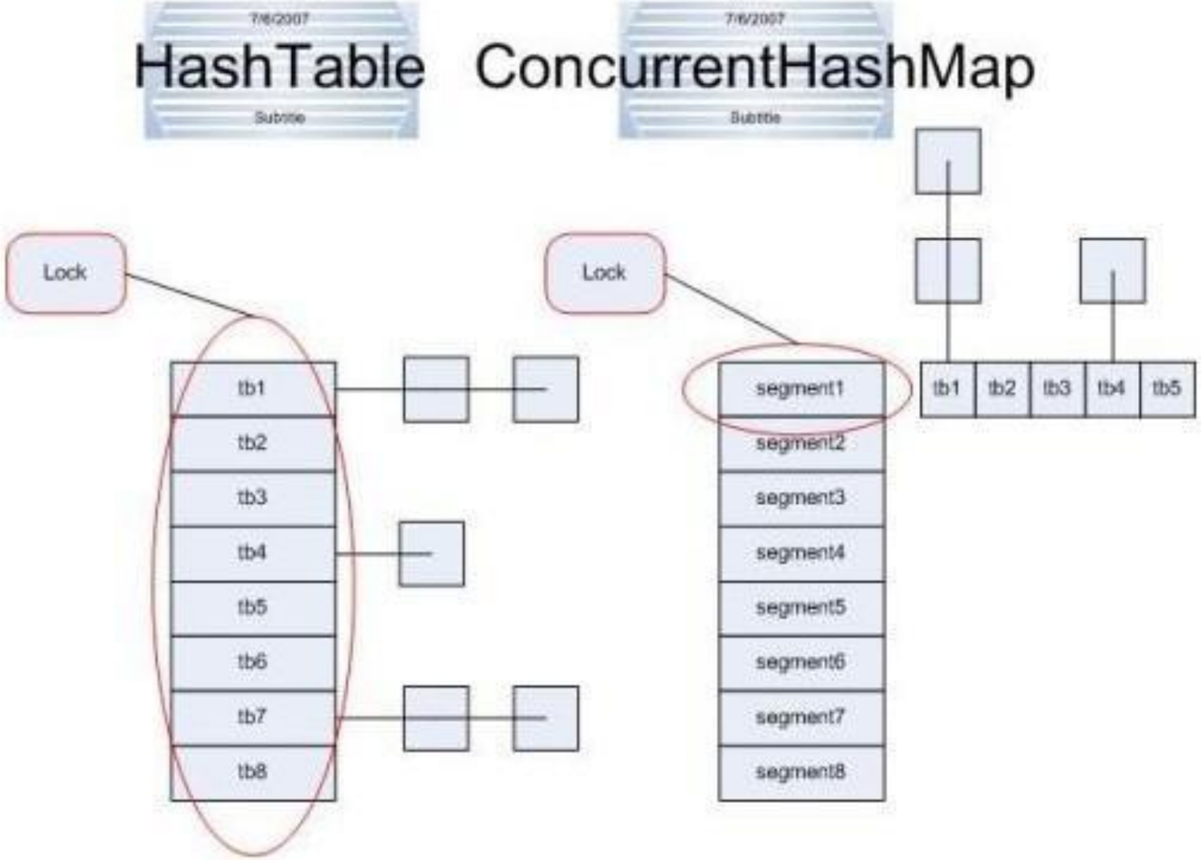


Parameters	HashMap	ConcurrentHashMap
Synchronization	Non-synchronized	synchronized
Thread-safety	Not thread-safe	Thread-safe
Iterator	It is fail-fast and throws an exception during iteration	It is fail-safe and performs iteration by multiple threads
Null Values	It allows for storing null keys and values.	It does not allow to store null key/values.
Performance	faster	Slower than Hashmap

Interview Q&A

12. If we have Hashtable which is already synchronized then why we need ConcurrentHashMap ?

Ans : Locking Mechanism still same as per HashMap (lock whole underlaying DS)



Interview Q&A

13. We Can Synchronize a HashMap using Collections then why can’t we use that instead using ConcurrentHashMap ?

Ans : if we used Collections.synchronizedMap(map) it will act as a synchronized Hashtable only where again locking mechanism is different

14. How HashMap Internally Works

Interview Q&A

Map<Employee, String> map=new HashMap<>();

Employee e1=new Employee(101,”Basant”)

Employee e2=new Employee(102,”peter”)

Employee e3=new Employee(103,”John”)

Employee e4=new Employee(104,”Sham”)

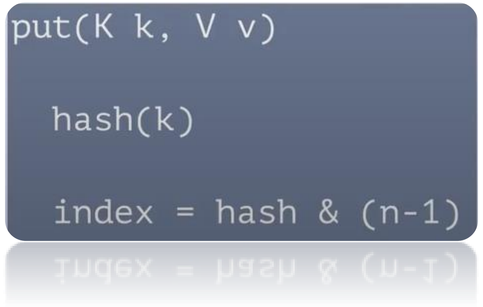
e2.equals(e3)

map.put(e1,”Dev”)

map.put(e2,”QA”)

map.put(e3,”UI”)

map.put(e4,”UI”)



```
put(K k, V v)

  hash(k)

  index = hash & (n-1)
```

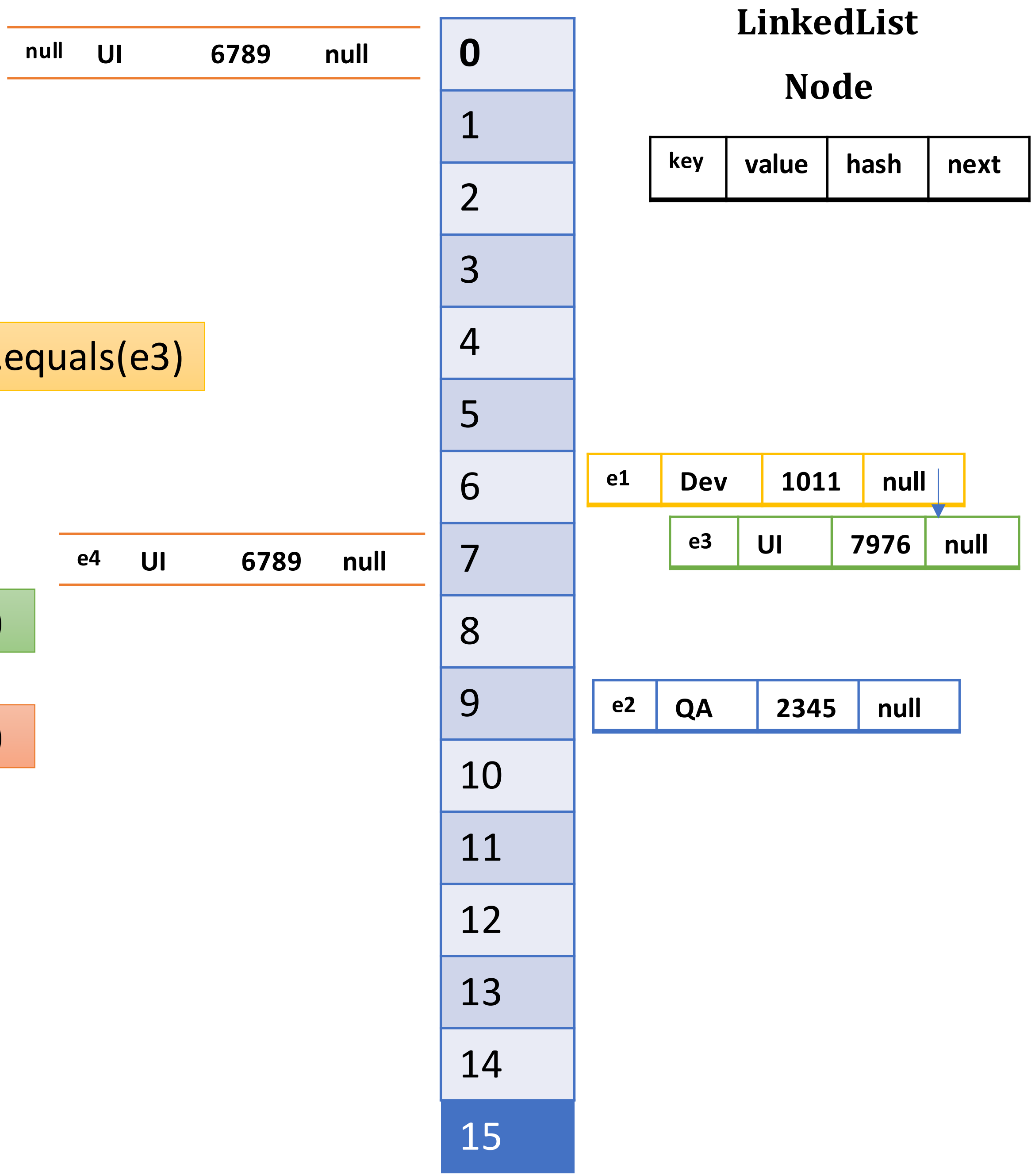
6

9

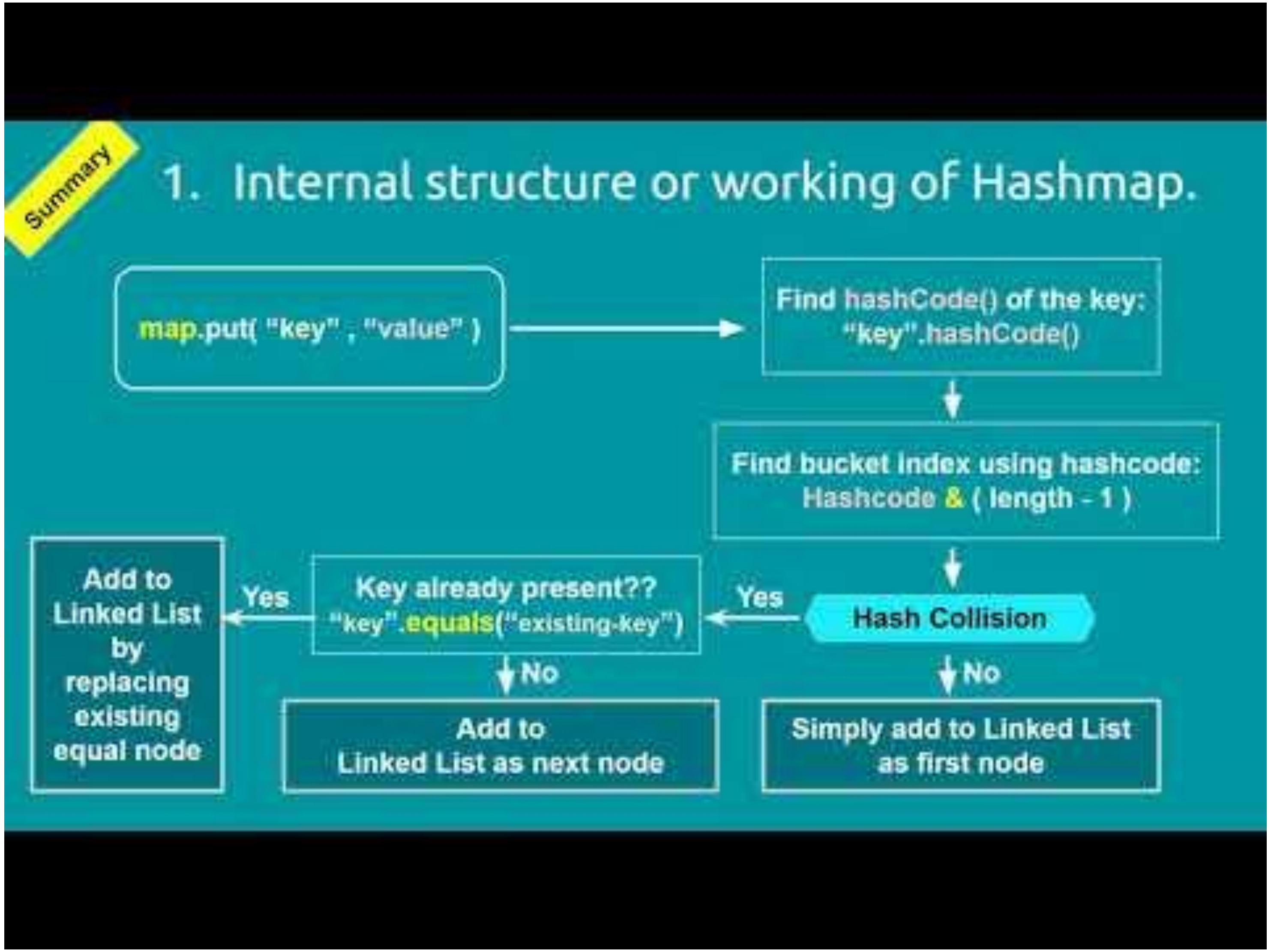
6

7

map.put(null,”UI”)



Interview Q&A



Summary

1. Internal structure or working of Hashmap.

map.put("key" , "value")

Find hashCode() of the key:
"key".hashCode()

Find bucket index using hashcode:
Hashcode & (length - 1)

Hash Collision

Yes

Key already present??
"key".equals("existing-key")

Yes

Add to Linked List by replacing existing equal node

No

Add to Linked List as next node

No

Simply add to Linked List as first node

Interview Q&A

15. If key is null in HashMap then where that entry will store in map ?

16. Map enhancement in java 8

17. How TreeMap internally works