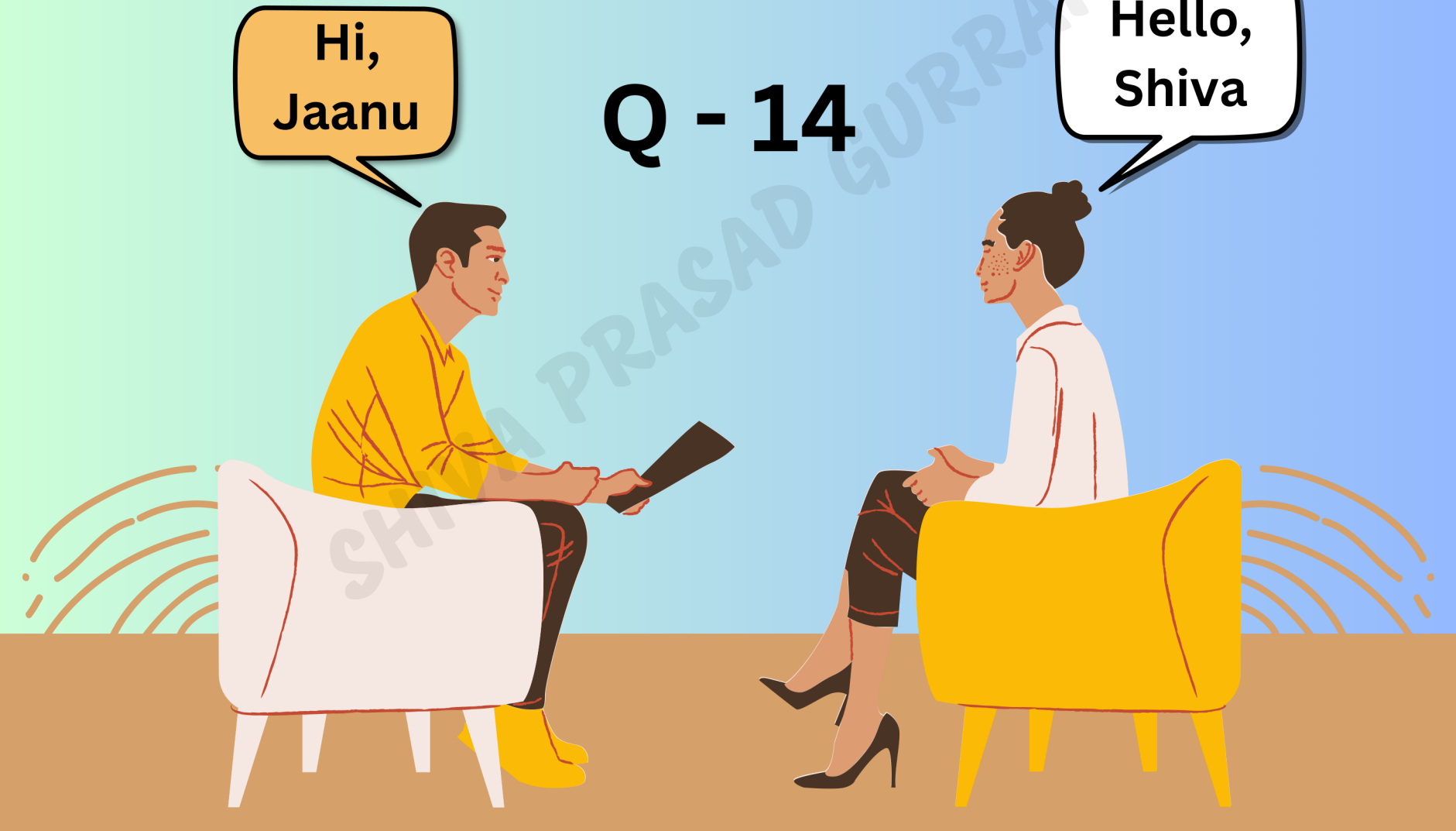
Java Interview Series

Shiva Prasad Gurram

LIVE



Q - 14


What is HashSet in Java and How it works internally?
Let me explain you in detail
SIVA PRASAD GURRAM

What is HashSet?

HashSet is a class (Essentially, it is a Data Structure) in Java which implements **Set** Interface and extends **AbstractSet**. It allows you to store **only unique** elements that means no duplicates are allowed.

- HashSet uses **HashMap** internally to store data in the form of **Key-Value pair**.
- Elements are stored using **hashing** technique, therefore it is not stored in an ordered fashion and the elements will be returned in random order.
- Elements are stored in the form of key-value pair (internally by HashMap) where key will the actual element value and value will be a constant called **PRESENT**.
- It contains only **unique elements.**
- Hashset is **not synchronized** (not thread-safe).
- This class permits at most one **null** element.
- HashSet has default **initial capacity of 16**.
- HashSet has **default load factor of 0.75 or 75%.**

- The only way to retrieve objects from the HashSet is through iterating the entire HashSet. This can be achieved by using iterator, for, for-each ,etc.

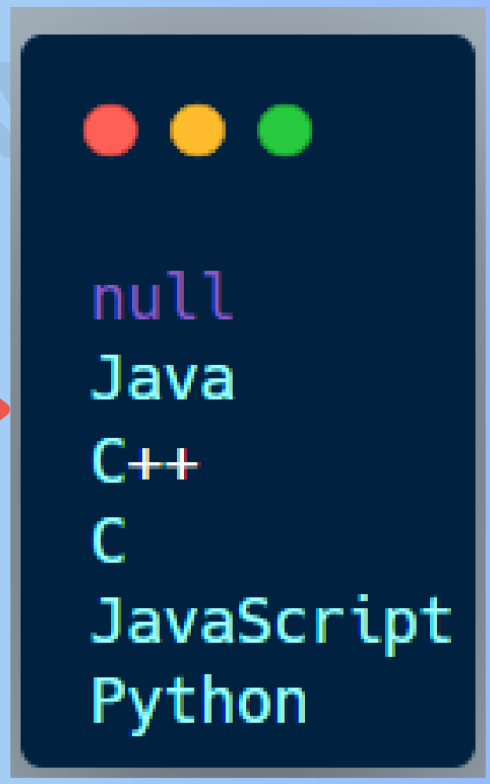
- The iterators returned by HashSet class iterator method are **fail-fast:** if the set is modified at any time after the iterator is created, in any way except through the iterator's own remove method, the Iterator throws a **ConcurrentModificationException**. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

- The class also offers **constant time performance** for the basic operations like add, remove, contains and size, assuming the hash function distribute the elements properly among the buckets.

Example of HashSet

```
HashSet<String> languages = new HashSet<>();
languages.add("Java");
languages.add("Python");
languages.add("JavaScript");
languages.add("C");
languages.add("C++");
languages.add(null);

Iterator<String> itr = languages.iterator();
while(itr.hasNext())
{
    System.out.println(itr.next());
}
```

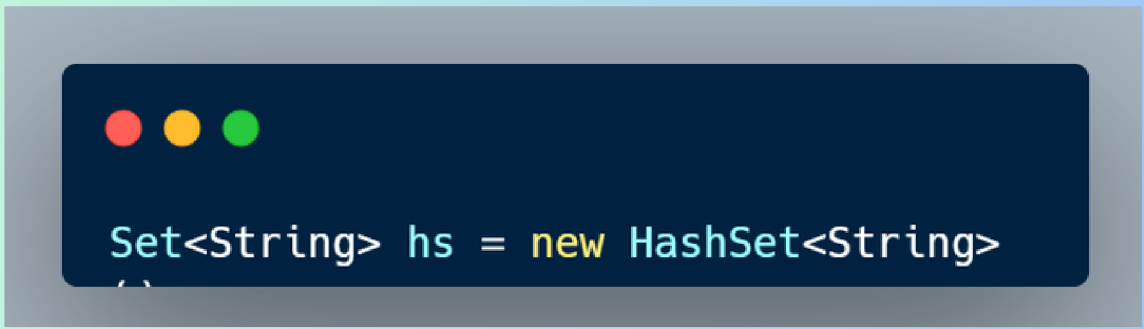


How HashSet works Internally?

In order to understand about HashSet internal behavior , one **must understand the HashMap internal working behavior**. This is because HashSet internally uses the HashMap behavior.

I highly recommend to go through the HashMap internal behavior How HashMap works internally? before proceeding.

The moment you create a HashSet the default constructor will get called and internally a backing HashMap object gets created automatically.



Constructors of HashSet

1) Default Constructor

Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load factor (0.75)

```
public HashSet() {
map = new HashMap<>();
}
```

In HashSet class, a map instance can be created at class level like below

```
private transient HashMap<E,Object> map;
```

2) Parameterized constructor with initialCapacity

Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and default load factor (0.75)

```
public HashSet(int initialCapacity) {
  map = new HashMap<>(initialCapacity);
}
```

3) Parameterized constructor with initialCapacity & loadFactor

Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and the specified load factor.

```
public HashSet(int initialCapacity, float loadFactor) {
   map = new HashMap<>(initialCapacity, loadFactor);
}
```

4) Parameterized constructor with Collection

Constructs a new set containing the elements in the specified collection

```
public HashSet(Collection<? extends E> c) {
  map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));
  addAll(c);
}
```

Lets see how add(E e) works

Adds the specified element to the set if it is not already present and then it returns **true**. If this set already contains the specified element, no operation would be performed and it returns **false**.

```
public boolean add(E e) {
    return map.put(e, PRESENT)==null;
}
```

Whenever we try to add an element using HashSet, it **stores internally as a key-value pair**.

Here Key is an element which we are trying to add using HashSet and Value is a **CONSTANT called PRESENT** (Same for all the HashSet elements). The PRESENT is defined in the HashSet class as below.

```
private static final Object PRESENT = new Object();
```

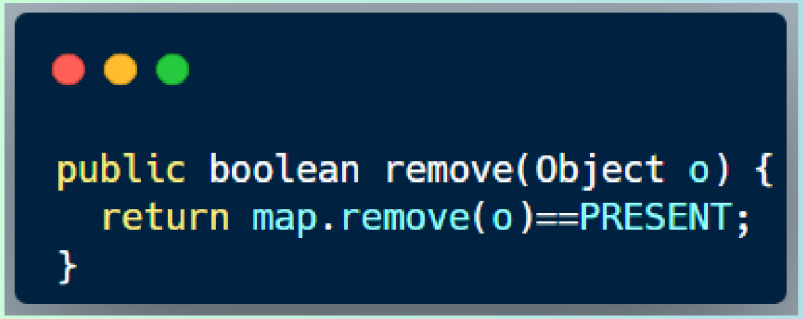
Ex: add("Java")

In background **"Java"** will behave like a "key" for the map and **PRESENT** will be it's value. So, add("Java") is similar to map.put("Java",PRESENT).

return map.put(e, PRESENT)==null => returns true if the map doesn't contains the specified element(e) , else false will be returned which means element already exists.

Lets see how remove(Object o) works

Removes the specified element from this set if it is present. This method returns true if this set changed as a result of the call.



```
public boolean remove(Object o) {
  return map.remove(o)==PRESENT;
}
```

Once this method returns true the element will not be present in Set.

Follow up question to reader

1) Why HashSet doesn't have get(Object o) method?

2) What are dis-advantages with HashSet?

How to access previous articles?

1) You can join my Telegram group, where I have uploaded pdf's of earlier articles.

@dailydsawithspg

2) You can visit my blog, where you can find earlier articles.

https://shivaprasadgurram.hashnode.dev/

Thank You

For more articles like this, follow me.

Shiva Prasad Gurram