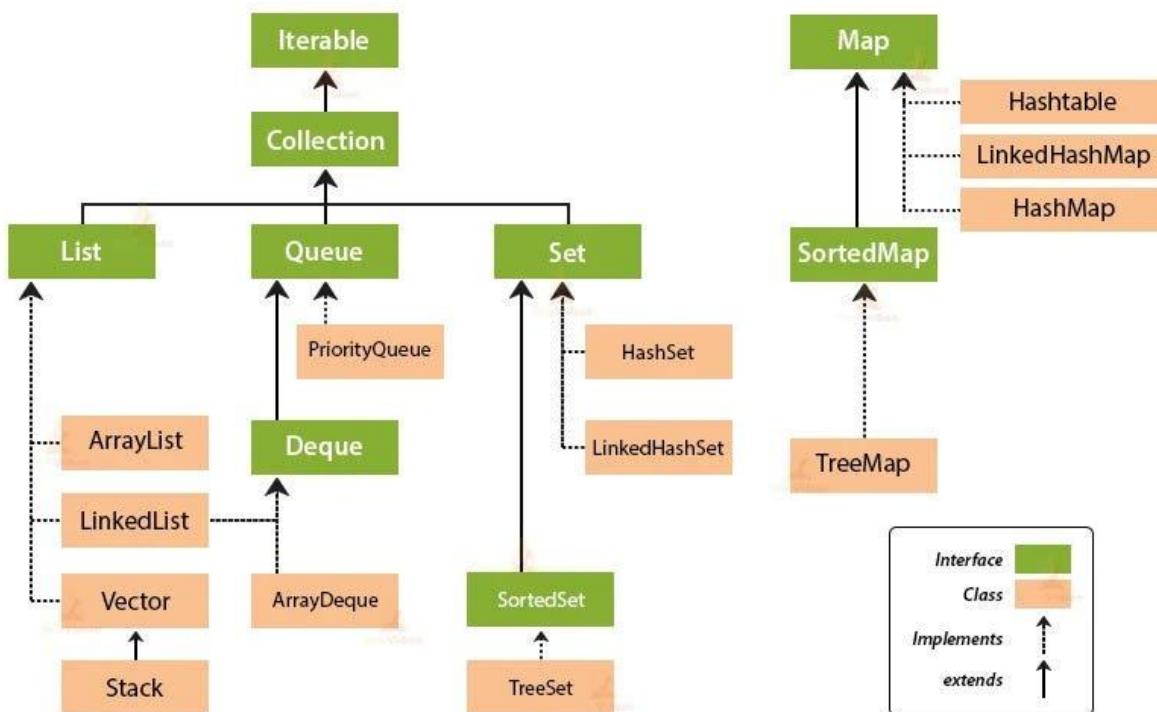


## Collection Framework Hierarchy in Java



سؤال: بيشتغل إزاي؟ **HashMap**

إجابة إنترفيو:

HashMap **hashing** بيتستخدم لتحديد bucket بكافأة **collision** ويعامل مع.

التوبيك:

**Hashing + Buckets + Collision Resolution**

الشرح بالرسم الذهني:

Key -> hashCode() -> hash -> index

Index 0: null

Index 1: [K1=V1] -> [K2=V2]

Index 2: null

Index 3: [K3=V3]

فاضي **bucket** لو:

✓ insert

لو فيه **collision**:

- Java < 8 → LinkedList
  - Java ≥ 8 → Red-Black Tree
- 

⚠ Tricky جداً:

- load factor = 0.75
  - resize = capacity \* 2
  - treeify threshold = 8
  - untreeify = 6
- 

❓ مهمين؟ **equals** و **hashCode** ليه

✓ إجابة:

يعرف يحط ويدور صح علشان **HashMap**.

⚠ Bug شائع:

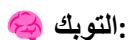
override equals بدون hashCode → **HashMap** بيوظ.

1 الفرق بين **hashCode()** و **equals()**

✓ إجابة إنترفيو مختصرة:

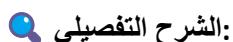
**equals()** بيقارن منطقياً

hashCode() بيحدد مكان التخزين في الـ hash-based collections.



## Object Equality Contract

---



### equals()

- بيرد على سؤال:

ده مساوي للثاني منطقاً؟ object هل الـ

مثال:

```
String a = new String("Omar");
```

```
String b = new String("Omar");
```

```
a.equals(b); // true
```

```
a == b; // false
```

---

### hashCode()

- بيرد على سؤال:

أنهـي؟ دـه فـي object أـحـط الـ

رـقم (int) يـرجـع بـيـرـجـع.

---



### بـيـسـتـخـدـمـهـمـ؟ إـزـايـ HashMap

1. يـحدـد الـ hashCode() يـنـدـي bucket

2. لو فيه عـنـاصـر:

- عـلـشـانـ يـتـأـكـدـ (equals()) يـنـدـي
- 



### (Contract) القواعد المهمة:

- لو `equals()` رجعت `true` ببقى متساوي (`hashCode()`) لازم
  - متساوي (`hashCode()`) لو مش شرط `equals() true`
- 

 **Tricky:**

override `equals` → HashSet يسمح بـ تكرار 

---

 **2 الفرق بين Comparable و Comparator**

 إجابة إنترفيو مختصرة:

Comparable ترتيب داخلي،  
Comparator ترتيب خارجي.

---

 **التوبك:**

**Sorting Strategy**

---

 **الشرح:**

**Comparable**

- بيحدد الترتيب الطبيعي
- جوه الكلاس نفسه

```
class User implements Comparable<User> {  
    public int compareTo(User u) {  
        return this.age - u.age;  
    }  
}
```

 ترتيب واحد بـ `age`.

---

## Comparator

- ترتيب خارجي
- أكثر من ترتيب

Comparator<User> byName =

(u1, u2) -> u1.name.compareTo(u2.name);

☞ مرن أكثر.

---

### ⚠ Tricky:

TreeSet لازم:

- Comparable أو
  - Comparator
- 

### 3 الفرق بين Iterable و Iterator

✓ إجابة إنترفيو مختصرة:

Iterable يتسمح بالـ for-each.

Iterator يبني على العناصر.

---

💡 التوبك:

## Iteration Mechanism

---

🔍 الشرح:

### Iterable

- Interface فيها method:

Iterator<T> iterator();

علشان:

```
for (T t : collection)
```

---

## Iterator

- مسؤول عن traversal

```
Iterator<Integer> it = list.iterator();  
while (it.hasNext()) {  
    System.out.println(it.next());  
}
```

---

### ⚠ Tricky:

- remove() من Iterator في
- remove() في collection أثناء loop → Exception

### 💡 الفرق بين Collection و Collections

✓ إجابة إنترفيو مختصرة:

Collection Interface:  
Collections Utility Class.

---

🧠 التوبك:

## API Design

---

🔍 الشرح:

## Collection

- Interface
- parent ↴ List, Set, Queue

```
Collection<String> c = new ArrayList<>();
```

---

## Collections

- Class
- فيها static methods

Collections.sort(list);

Collections.synchronizedList(list);

---

### ⚠ Tricky:

- Collection = data
  - Collections = helper methods
- 

دوره	المفهوم
مقارنة منطقية	equals
توزيع في buckets	hashCode
ترتيب داخلي	Comparable
ترتيب خارجي	Comparator
يسمح بـ for-each	Iterable
يمشي على العناصر	Iterator
Interface	Collection
Utility class	Collections