📘 HashMap in Java – Detailed Notes

🚀 What is a HashMap?

A **HashMap** in Java is a data structure that stores data in **key–value pairs**.
It uses **hashing** to provide **O(1) average-time complexity** for insertion, deletion, and lookup.

Think of it like a super-fast dictionary where the "key" is your entry pass and the "value" is everything behind the door.

⚙️ Key Features

- **Stores key–value pairs**
- **No duplicate keys**
- **Allows one null key and multiple null values**
- **Not synchronized** (not thread-safe)
- **Order of elements is not guaranteed**

🧠 Internal Working

1. Every key is passed through a **hash function**.
2. This determines the **bucket index**.
3. Each bucket stores a **Node** (key, value, hash, next).
4. If two keys land in the same bucket → **collision**
   Java handles it using **LinkedList** and **balanced trees (after Java 8)**.

🧩 Commonly Used HashMap Methods

Method	Description
`put(K key, V value)`	Insert or update value for a key
`get(Object key)`	Returns value for the key
`containsKey(Object key)`	Checks if key exists
`containsValue(Object value)`	Checks if value exists
`remove(Object key)`	Removes entry for key
`size()`	Number of key-value pairs
`isEmpty()`	Checks if map is empty
`clear()`	Removes all entries
`putIfAbsent(K key, V value)`	Adds value only if key not present
`keySet()`	Returns Set of all keys
`values()`	Returns Collection of all values
`entrySet()`	Returns Set of key-value pairs (Map.Entry)
`getOrDefault(key, defaultValue)`	Returns default if key missing
`replace(key, newValue)`	Replaces value if key exists
`forEach(action)`	Iterates using lambda

📝 Examples

1️⃣ Basic HashMap Example

```
import java.util.*;

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

```java
        map.put(101, "Amar");
        map.put(102, "Aditi");
        map.put(103, "Omkar");

        System.out.println(map);
    }
}
```

2️⃣ Accessing Elements

```java
System.out.println(map.get(102));    // Aditi
System.out.println(map.containsKey(101));   // true
System.out.println(map.containsValue("Omkar")); // true
```

3️⃣ Iterating Over HashMap

🔹 Using `entrySet()`

```java
for (Map.Entry<Integer, String> entry : map.entrySet()) {
    System.out.println(entry.getKey() + " -> " + entry.getValue());
}
```

🔹 Using Lambda

```java
map.forEach((k, v) -> System.out.println(k + " => " + v));
```

4️⃣ Using `putIfAbsent()`

```java
map.putIfAbsent(104, "Jay");
map.putIfAbsent(103, "Shruti"); // Won't update since key exists
```

5️⃣ Removing Elements

```java
map.remove(102);
map.remove(103, "Omkar"); // remove only if both match
```

🧮 Real-Life Use Case Example

Case: Student Marks Management

```
HashMap<String, Integer> marks = new HashMap<>();

marks.put("Amar", 92);
marks.put("Aman", 85);
marks.put("Sanjay", 78);

// Update
marks.put("Amar", 89);

// Access
System.out.println("Aman's Marks: " + marks.get("Aman"));

// Iterate
for(String name : marks.keySet()) {
    System.out.println(name + " : " + marks.get(name));
}
```

🛡️ When to Use HashMap?

Use it when:

- You need **fast search**
- You have **unique keys**
- Order does not matter

📉 Time Complexity

Operation	Complexity
Insert	O(1) average
Search	O(1) average
Delete	O(1) average

Worst Case (tree collision)	O(log n)