**Adding Key-Value Pairs to a Map in Java**

**1. put(key, value) Method**

* Adds a key-value pair to the map.
* If the key **exists**, it **replaces** the existing value.
* Returns the **previous value** bound to the key (or null if the key is new).

**2. putIfAbsent(key, value) Method (Java 8)**

* Adds the key-value pair **only if the key is not present** or is associated with null.
* Helps in **removing faulty null values** in the map.

**3. NullPointerException Issue**

* Iterating over map.values() can cause NullPointerException when auto-unboxing null into int.
* Example:

for (int value : map.values()) { // Fails if value is null

System.out.println(value);

}

**4. Fixing Null Values with putIfAbsent**

* Replace null values with a **default value** (e.g., -1) to prevent NullPointerException:

for (String key : map.keySet()) {

map.putIfAbsent(key, -1);

}

* Ensures the map has **no null values**.

**5. Expected Output After Fix**

value = -1

value = 1

value = -1

value = 3

value = 5

This summary should help you quickly recall key concepts for your interview. 🚀

**Getting a Value From a Key in Java**

**1. get(key) Method**

* Retrieves the value associated with a given key.
* Returns null if the key is not found.

**2. getOrDefault(key, defaultValue) (Java 8)**

* Returns the value for the given key **if it exists**.
* Returns the **default value** if the key is **not present** in the map.

**3. Example Usage**

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

map.put(1, "one");

map.put(2, "two");

map.put(3, "three");

List<String> values = new ArrayList<>();

for (int key = 0; key < 5; key++) {

values.add(map.getOrDefault(key, "UNDEFINED"));

}

System.out.println("values = " + values);

**4. Using Streams (Alternative Approach)**

List<String> values =

IntStream.range(0, 5)

.mapToObj(key -> map.getOrDefault(key, "UNDEFINED"))

.collect(Collectors.toList());

System.out.println("values = " + values);

**5. Expected Output**

values = [UNDEFINED, one, two, three, UNDEFINED]

This will help you quickly recall key concepts for your interview. 🚀

**Removing a Key from a Map in Java**

**1. remove(key) Method**

* Removes a **key-value pair** from the map.
* Returns the **value** that was associated with the key.
* If the key is not found, it **returns null**.

**2. remove(key, value) (Java 8 Overload)**

* Removes the **key-value pair** **only if both key and value match**.
* Returns true if the pair was removed, false otherwise.

**3. Example Usage**

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

map.put("one", 1);

map.put("two", 2);

map.put("three", 3);

// Removing a key

int removedValue = map.remove("two");

System.out.println("Removed Value: " + removedValue); // Output: 2

// Removing a key-value pair (Java 8)

boolean isRemoved = map.remove("three", 5); // Key exists but value doesn't match

System.out.println("Was the pair removed? " + isRemoved); // Output: false

isRemoved = map.remove("three", 3); // Exact match

System.out.println("Was the pair removed? " + isRemoved); // Output: true

**4. Expected Output**

Removed Value: 2

Was the pair removed? false

Was the pair removed? true

This ensures safe key removal and prevents accidental deletions. 🚀

**Checking for the Presence of a Key or Value in Java**

**1. containsKey(key) Method**

* Returns true if the map **contains the specified key**.
* Returns false if the key is **not found**.

**2. containsValue(value) Method**

* Returns true if the map **contains the specified value**.
* Returns false if the value is **not present**.

**Checking and Modifying the Content of a Map in Java**

**1. isEmpty() Method**

* Returns true if the map is **empty** (contains no key-value pairs).
* Returns false if the map has at least one key-value pair.

**2. size() Method**

* Returns the **number of key-value pairs** in the map.

**3. clear() Method**

* Removes **all key-value pairs**, making the map empty.

**4. putAll(otherMap) Method**

* **Adds all key-value pairs** from otherMap to the current map.
* If **keys exist in both maps**, values from otherMap **replace** the existing values.

This helps in managing and merging map content effectively. 🚀

**Getting Views of Keys, Values, and Entries in Java Maps**

**1. keySet() Method**

* Returns a Set<K> containing **all keys** in the map.
* Modifying this set **removes** the corresponding key-value pair from the map.

**2. values() Method**

* Returns a Collection<V> containing **all values** in the map.
* Removing a value **only removes the first occurrence**, not all matching values.

**3. entrySet() Method**

* Returns a Set<Map.Entry<K, V>> containing **all key-value pairs**.
* Best choice for iterating over a map efficiently.

**4. Example Usage**

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

map.put(1, "one");

map.put(2, "two");

map.put(3, "three");

map.put(4, "four");

// Get keys, values, and entries

Set<Integer> keys = map.keySet();

Collection<String> values = map.values();

Set<Map.Entry<Integer, String>> entries = map.entrySet();

// Print results

System.out.println("Keys: " + keys); // Output: [1, 2, 3, 4]

System.out.println("Values: " + values); // Output: [one, two, three, four]

System.out.println("Entries: " + entries); // Output: [1=one, 2=two, 3=three, 4=four]

// Removing a key

keys.remove(3);

System.out.println("Map after removing key 3: " + map); // Key 3 is removed

// Removing a value (removes only the first occurrence)

values.remove("four");

System.out.println("Map after removing value 'four': " + map);

// Iterating efficiently using entrySet()

for (Map.Entry<Integer, String> entry : map.entrySet()) {

System.out.println("Entry: " + entry);

}

**5. Expected Output**

Keys: [1, 2, 3, 4]

Values: [one, two, three, four]

Entries: [1=one, 2=two, 3=three, 4=four]

Map after removing key 3: {1=one, 2=two, 4=four}

Map after removing value 'four': {1=one, 2=two}

Entry: 1=one

Entry: 2=two

**6. Important Notes**

* Removing a key from keySet() **removes the key-value pair from the map**.
* Removing a value from values() **removes only the first occurrence**.
* **You cannot add elements** to keySet() or values()—doing so throws an UnsupportedOperationException.
* **Best iteration method:** Use entrySet() for efficiency. 🚀