Java Map Implementations

In Java, `Map` is an **interface** in the `java.util` package that maps **keys to values**. It cannot contain duplicate keys, and each key can map to **at most one value**.

There are three commonly used implementations of the `Map` interface:

✅ 1. `HashMap`

- **Order**: No guarantee of order (neither insertion nor sorted).
- **Null Keys/Values**: Allows **one null key** and **multiple null values**.
- **Performance**: Fast lookup, insertion, and deletion (O(1) on average).
- **Internally**: Uses **hash table**.

Example:

```java
import java.util.HashMap;

public class HashMapExample {
    public static void main(String[] args) {
        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "Apple");
        map.put(3, "Banana");
        map.put(2, "Cherry");

        System.out.println(map); // Order not guaranteed
    }
}
```

✅ 2. `LinkedHashMap`

- **Order**: Maintains **insertion order**.
- **Null Keys/Values**: Allows one null key and multiple null values.
- **Performance**: Slightly slower than `HashMap` (due to maintaining order).
- **Internally**: Uses a **hash table + doubly-linked list** to maintain order.

Example:

```java
import java.util.LinkedHashMap;

public class LinkedHashMapExample {
    public static void main(String[] args) {
        LinkedHashMap<Integer, String> map = new LinkedHashMap<>();
        map.put(1, "Apple");
        map.put(3, "Banana");
        map.put(2, "Cherry");
        System.out.println(map); // Maintains insertion order
    }
}
```

✅ 3. `TreeMap`

- **Order**: Sorted according to the **natural ordering** of keys or a custom **Comparator**.
- **Null Keys/Values**: Does **not allow null keys**, but allows null values.
- **Performance**: Slower than `HashMap`, operations are **O(log n)**.
- **Internally**: Implements a **Red-Black Tree**.

Example:

```java
import java.util.TreeMap;

public class TreeMapExample {
    public static void main(String[] args) {
        TreeMap<Integer, String> map = new TreeMap<>();
        map.put(3, "Banana");
        map.put(1, "Apple");
        map.put(2, "Cherry");

        System.out.println(map); // Sorted by keys
    }
}
```

🔍 Comparison Table

Feature	HashMap	LinkedHashMap	TreeMap
Ordering	No order	Insertion order	Sorted (by key)
Null Keys	One null key	One null key	No null key
Null Values	Allowed	Allowed	Allowed
Performance	O(1)	O(1)	O(log n)
Thread-safe	❌ No	❌ No	❌ No
Use When	Fast operations	Order matters	Sorted map needed

🔐 Notes:

- If you want **thread-safe** maps, use `Collections.synchronizedMap()` or `ConcurrentHashMap`.
- Use `Map<K, V>` as a reference type when coding, and choose the implementation as needed.

Would you like a real-world scenario comparison for when to use each?

Here are practical **examples of using `HashMap`, `LinkedHashMap`, and `TreeMap` in test automation scenarios** (e.g., with Selenium + TestNG or JUnit). These examples are commonly used to manage test data, validate responses, or map UI elements.

✅ 1. `HashMap` in Test Case: Storing Test Data (Unordered)

Use when order doesn't matter — e.g., data-driven testing.

```java
import java.util.HashMap;
import org.testng.annotations.Test;

public class HashMapTest {

    @Test
    public void loginDataTest() {
        HashMap<String, String> loginData = new HashMap<>();
        loginData.put("admin", "admin123");
        loginData.put("user", "user123");
        loginData.put("guest", "guest123");

        for (String username : loginData.keySet()) {
            String password = loginData.get(username);
            System.out.println("Testing login with: " + username + "/" + password);
            // Simulate login(username, password);
        }
    }
}
```

✅ 2. `LinkedHashMap` in Test Case: Validating Dropdown Options (Order Matters)

Use when insertion order matters — e.g., verifying UI dropdown values.

```java
import java.util.LinkedHashMap;
import java.util.Map;
import org.testng.annotations.Test;
import org.testng.Assert;

public class LinkedHashMapTest {

    @Test
    public void dropdownValidationTest() {
        LinkedHashMap<Integer, String> expectedOptions = new LinkedHashMap<>();
        expectedOptions.put(1, "Select");
        expectedOptions.put(2, "Male");
        expectedOptions.put(3, "Female");
        expectedOptions.put(4, "Other");

        // Simulate getting dropdown options
        String[] actualOptions = {"Select", "Male", "Female", "Other"};

        int i = 0;
        for (Map.Entry<Integer, String> entry : expectedOptions.entrySet()) {
            Assert.assertEquals(actualOptions[i], entry.getValue(), "Mismatch at position " +
i);
            i++;
        }
    }
}
```

✅ 3. `TreeMap` in Test Case: Sorted Validation (e.g., Sorting Data Table)

Use when data needs to be sorted by keys automatically — e.g., sorting usernames, IDs, etc.

```java
import java.util.TreeMap;
import java.util.Map;
import org.testng.annotations.Test;

public class TreeMapTest {

    @Test
    public void sortedUserTest() {
        TreeMap<Integer, String> users = new TreeMap<>();
        users.put(102, "Alice");
        users.put(101, "Bob");
        users.put(103, "Charlie");

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

        // Output is sorted by ID automatically
    }
}
```

🔍 Use Case Summary

Map Type	When to Use in Testing	Example Use Case
`HashMap`	Fast lookup, no order needed	Username-password map for login tests
`LinkedHashMap`	Maintain insertion order	Dropdown or menu validations
`TreeMap`	Automatically sort by keys	Validating sorted data in tables or reports

---

Let me know if you'd like to see these with **Selenium WebDriver** examples as well!

Great! Below are **Selenium WebDriver test case examples** using `HashMap`, `LinkedHashMap`, and `TreeMap` in realistic UI testing scenarios. These examples use **Java + Selenium + TestNG**.

✅ 1. `HashMap` Example — Multiple Logins (Order Doesn't Matter)

Use `HashMap` when the order of usernames doesn't matter — just want to run login tests for multiple users.

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;
import java.util.HashMap;

public class LoginTestWithHashMap {
    WebDriver driver;

    @BeforeClass
    public void setUp() {
        driver = new ChromeDriver();
        driver.get("https://example.com/login");
    }

    @Test
    public void loginTest() {
        HashMap<String, String> credentials = new HashMap<>();
        credentials.put("admin", "admin123");
        credentials.put("user", "user123");
        credentials.put("guest", "guest123");

        for (String username : credentials.keySet()) {
            driver.findElement(By.id("username")).clear();
            driver.findElement(By.id("username")).sendKeys(username);

            driver.findElement(By.id("password")).clear();
            driver.findElement(By.id("password")).sendKeys(credentials.get(username));

            driver.findElement(By.id("loginBtn")).click();

            // Assert something like login success message
            // driver.findElement(By.id("logout")).click(); // prepare for next iteration
        }
    }

    @AfterClass
    public void tearDown() {
        driver.quit();
    }
}
```

✅ 2. `LinkedHashMap` Example — Validate Dropdown Options in Order

Use `LinkedHashMap` to verify **exact dropdown order** on UI (important in forms).

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;
import org.testng.Assert;

import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

public class DropdownOrderTest {
    WebDriver driver;

    @BeforeClass
    public void setUp() {
        driver = new ChromeDriver();
        driver.get("https://example.com/register");
    }

    @Test
    public void genderDropdownTest() {
        LinkedHashMap<Integer, String> expectedOptions = new LinkedHashMap<>();
        expectedOptions.put(1, "Select");
        expectedOptions.put(2, "Male");
        expectedOptions.put(3, "Female");
        expectedOptions.put(4, "Other");

        WebElement dropdown = driver.findElement(By.id("gender"));
        List<WebElement> options = dropdown.findElements(By.tagName("option"));

        int index = 0;
        for (Map.Entry<Integer, String> entry : expectedOptions.entrySet()) {
            Assert.assertEquals(options.get(index).getText(), entry.getValue(), "Mismatch at
index " + index);
            index++;
        }
    }

    @AfterClass
    public void tearDown() {
        driver.quit();
    }
}
```

✅ 3. `TreeMap` Example — Validate Sorted Table Rows (by Key/ID)

Use **`TreeMap`** to verify that table data (like user IDs or names) is sorted.

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

import java.util.List;
import java.util.TreeMap;
import java.util.Map;

public class SortedTableTest {
    WebDriver driver;

    @BeforeClass
    public void setUp() {
        driver = new ChromeDriver();
        driver.get("https://example.com/userlist");
    }

    @Test
    public void verifySortedUserIDs() {
        List<WebElement> rows =
driver.findElements(By.xpath("//table[@id='userTable']/tbody/tr"));

        TreeMap<Integer, String> userMap = new TreeMap<>();

        for (WebElement row : rows) {
            int userId = Integer.parseInt(row.findElement(By.xpath("td[1]")).getText());
            String userName = row.findElement(By.xpath("td[2]")).getText();
            userMap.put(userId, userName);
        }

        System.out.println("Users sorted by ID (TreeMap):");
        for (Map.Entry<Integer, String> entry : userMap.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }

        // Optional: add assert to confirm expected sorted order
    }

    @AfterClass
    public void tearDown() {
        driver.quit();
    }
}
```

✅ Summary

Map Type	Used For	Selenium Example
`HashMap`	Store and iterate over test data (no order)	Login test for multiple users
`LinkedHashMap`	Verify exact order (e.g., dropdown items)	Gender dropdown values
`TreeMap`	Automatically sort keys (e.g., table validation)	Sorted table data by user ID

Would you like the same with **data from Excel or JSON files** integrated as well (for data-driven testing)?