**Shallow Equality vs. Deep Equality in Java Arrays**

In Java, comparing arrays can be done using **shallow equality** (using ==) or **deep equality** (using Arrays.equals() or Arrays.deepEquals()). Let's explore the differences:

**One-Dimensional int[] Arrays**

Given two one-dimensional integer arrays A and B:

int[] A = {1, 2, 3};

int[] B = {1, 2, 3};

**Shallow Equality (==)**

* A == B **checks if A and B refer to the same memory location.**
* If A and B are different objects (even if their contents are identical), A == B returns **false**.

Example:

int[] A = {1, 2, 3};

int[] B = {1, 2, 3};

System.out.println(A == B); // false (different memory locations)

However, if we do:

int[] A = {1, 2, 3};

int[] B = A;

System.out.println(A == B); // true (both reference the same array)

**Deep Equality (Arrays.equals())**

* Arrays.equals(A, B) **compares the actual contents** of both arrays.
* Returns **true** if A and B have the same length and identical elements in the same order.

Example:

System.out.println(Arrays.equals(A, B)); // true (content is identical)

**Two-Dimensional int[][] Arrays**

Now consider 2D arrays:

int[][] A = {{1, 2, 3}, {4, 5, 6}};

int[][] B = {{1, 2, 3}, {4, 5, 6}};

**Shallow Equality (==)**

* A == B **checks if A and B refer to the same 2D array object in memory.**
* If they are separate objects, it returns **false**.

Example:

System.out.println(A == B); // false (different objects)

* Even if A and B contain identical sub-arrays, A == B will still be **false**.

**Shallow Content Equality (Arrays.equals(A, B))**

* Arrays.equals(A, B) **compares only the top-level array references**, **not** their contents.
* Since A and B are separate objects with different inner array references, Arrays.equals(A, B) will return **false**.

Example:

System.out.println(Arrays.equals(A, B)); // false (compares only references)

* However, if both reference the same object, it returns **true**:

int[][] B = A;

System.out.println(Arrays.equals(A, B)); // true

**Deep Equality (Arrays.deepEquals(A, B))**

* Arrays.deepEquals(A, B) **recursively compares the contents of both arrays**.
* It checks if all inner arrays have the same length and elements in the same order.

Example:

System.out.println(Arrays.deepEquals(A, B)); // true (content is identical)

**Summary Table**

| **Comparison Type** | **1D Array (int[] A, B)** | **2D Array (int[][] A, B)** |
| --- | --- | --- |
| A == B (Shallow Equality) | **Checks memory reference** (returns true only if they are the same object) | **Checks reference of top-level array only** (ignores inner arrays) |
| Arrays.equals(A, B) (Shallow Content Equality) | **Checks element-wise equality** | **Compares only top-level references, not contents** (returns false even if contents match) |
| Arrays.deepEquals(A, B) (Deep Content Equality) | **Same as Arrays.equals()** (since it's 1D) | **Recursively compares all elements and inner arrays** (true if contents match) |

**Conclusion**

* Use Arrays.equals(A, B) for **1D arrays**.
* Use Arrays.deepEquals(A, B) for **2D arrays** (or higher dimensions).
* Avoid using == for array comparisons unless checking **if both variables point to the same object**.