**Arrays.toString() in Java with Examples**

* Returns a string representation of the contents of the specified array.
* The string representation consists of a list of the array’s elements, enclosed in square brackets (“[]”). Adjacent elements are separated by the characters “,” (a comma followed by a space).
* Returns “null” if array is null.

// Java program to demonstrate working of Arrays.toString()

import java.io.\*;

import java.util.\*;

class GFG {

public static void main(String[] args)

{

// Let us create different types of arrays and

// print their contents using Arrays.toString()

boolean[] boolArr = new boolean[] { true, true, false, true };

byte[] byteArr = new byte[] { 10, 20, 30 };

char[] charArr = new char[] { 'g', 'e', 'e', 'k', 's' };

double[] dblArr = new double[] { 1, 2, 3, 4 };

float[] floatArr = new float[] { 1, 2, 3, 4 };

int[] intArr = new int[] { 1, 2, 3, 4 };

long[] longArr = new long[] { 1, 2, 3, 4 };

**Object[] objArr = new Object[] { 1, 2, 3, 4 };**

short[] shortArr = new short[] { 1, 2, 3, 4 };

System.out.println(Arrays.toString(boolArr));

System.out.println(Arrays.toString(byteArr));

System.out.println(Arrays.toString(charArr));

System.out.println(Arrays.toString(dblArr));

System.out.println(Arrays.toString(floatArr));

System.out.println(Arrays.toString(intArr));

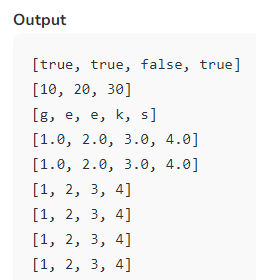
System.out.println(Arrays.toString(lomgArr));

System.out.println(Arrays.toString(objArr));

System.out.println(Arrays.toString(shortArr));

}

}



import java.util.Arrays;

class HelloWorld {

public static void main( String args[] ) {

// Retrieve an item from an array

int myArray[] = new int[]{1,2,3};

System.out.println(myArray[0]); // Outputs 1

// Update an item in an array

myArray = new int[]{4, 5, 6};

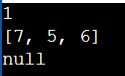
myArray[0] = 7;

System.out.println(Arrays.toString(myArray)); // Outputs {7,5,6}

}

}

**Output**

****

Instead of myArray = new int[]{4, 5, 6}; you can’t write the below as these generate error

1. int myArray[] ={4, 5, 6}; because You cannot declare a variable with the same name twice within the same scope
2. myArray[] ={4, 5, 6};

Here's why myArray = new int[]{4, 5, 6}; doesn't generate an error:

* **Assignment vs. Redeclaration:** This line doesn't declare a new variable named myArray. It assigns a new array object to an existing variable.
* **Reusing the Variable:** The variable myArray was already declared earlier with int myArray[] = {1, 2, 3};. This line simply changes the value it holds to refer to a different array object.
* **No Conflict:** Since you're not trying to create a new variable with the same name within the same scope, there's no conflict or error.

**Breaking It Down:**

1. new int[]{4, 5, 6}; creates a new array object in memory.
2. = assigns this new array object to the existing variable myArray.
3. The original array [1, 2, 3] is no longer accessible through myArray because the variable now points to the new array [4, 5, 6].

**Essentially, you're changing the value the variable holds, not creating a new variable with the same name.**