1Z0-808 Exam Topic Reviewer

TopicId: 1024
Topic: One-Dimensional and Multi-Dimensional Arrays

August 5, 2025

Arrays: Java's Foundational Data Structure

Let's get back to basics. Arrays are a core part of the Java language, and you cannot pass the 1Z0-808 exam without mastering their syntax, especially the tricky parts. An array is a fixed-size, indexed collection of elements, all of the same data type. While they might seem simple, the exam will test you on every strange but valid way to declare and initialize them. Let's think like the compiler.

0.1 Declaration, Instantiation, and Initialization

Working with arrays is a three-step process, though these steps can be combined.

(a) **Declaration:** Creates a reference variable. No object exists yet; the reference is `null`. The exam loves to mix up the syntax.

```
int[] scores; // Preferred style
int scores[]; // C-style, also valid
```

(b) **Instantiation:** Creates the array object on the heap with a specified size. Elements are initialized to their default values (e.g., `0` for `int`, `null` for objects).

```
scores = new int[10]; // Creates an array of 10 integers
```

(c) **Initialization:** Assigning values to the elements.

```
scores[0] = 95;
```

Array Declaration and Initialization Traps

This is where you need to be sharp. Pay attention to these examples.

- **Mixed Declarations:** Brackets can apply to the type or the variable.

```
int[] array1, array2; // Both are int arrays
int array3[], var1;   // array3 is an int array, var1 is just an int
int[] array4, var2[]; // array4 is a 1D array, var2 is a 2D array
```

- **Array Literals (Anonymous Arrays):** You can declare, instantiate, and initialize in one go. But there's a catch.

```
int[] numbers = {10, 20, 30}; // This is valid


int[] moreNumbers;
// moreNumbers = {40, 50}; // COMPILE ERROR!
moreNumbers = new int[]{40, 50}; // This is the correct way
```

You **cannot** use the array literal shorthand to initialize an array on a separate line from its declaration.

0.2 Access, Length, and Exceptions

– **Access:** Use zero-based index, e.g., `numbers[0]`.

- **Length:** Use the `length` **property** (not a method!). The exam will try to trick you with `array.length()`.
- `ArrayIndexOutOfBoundsException`**:** This is a runtime exception thrown when you access an invalid index (less than 0 or greater than or equal to `length`). Code like `numbers[numbers.length]` will always fail.

0.3 Multi-Dimensional Arrays

These are simply arrays of arrays.

- **Declaration and Instantiation:**

```
String[][] names = new String[2][3];
```

- **Asymmetric (Ragged) Arrays:** The inner arrays do not need to be the same length. This is a common exam topic.

```
int[][] ragged = new int[3][]; // Only the first dimension size is mandat
ragged[0] = new int[5];
ragged[1] = new int[2];
ragged[2] = new int[3];
System.out.println(ragged[1][0]); // Accesses the first element of the se
```

- **Literals:**

```
char[][] letters = { {'a','b'}, {'c','d','e'} };
System.out.println(letters.length);    // Prints 2 (the size of the outer
System.out.println(letters[1].length); // Prints 3 (the size of the secon
```

Key Takeaways for the 1Z0-808 Exam

- Master the different declaration syntaxes (`int[] x` vs. `int x[]`).
- Remember the rule for initializing with a literal: must be on the same line as declaration unless you use the `new int[]...` syntax.
- It's `array.length` (a property), not `array.length()` (a method).
- For multi-dimensional arrays, you only need to specify the size of the outermost dimension at instantiation.