ARRAY IN JAVA

1. What is an array and how is it declared in Java?

Answer:

An array is a collection of elements of the same data type, accessed using an index starting from 0. You declare an array with:

Java

dataType[] arrayName = new dataType[size];

Use code with caution. Learn more

2. What are the different ways to initialize an array?

Answer:

You can initialize during declaration:

Java

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

Use code with caution. Learn more

Or assign values later using the index:

Java

String[] names = new String[3];

names[0] = "Alice";

Use code with caution. Learn more

3. What is the difference between length and length()?

Answer:

length is a property of the array object, accessed without parentheses, returning the number of elements. length() is a method of the String class, returning the

number of characters.

4. Can you explain multidimensional arrays?

Answer:

Yes, arrays can have multiple dimensions, creating a grid-like structure. Each element is accessed using multiple indices.

5. How do you iterate through an array?

Answer:

You can use for loops or enhanced for-each loops to access each element based on its index.

6. What are common array operations and their time complexities?

Answer:

Accessing an element: O(1). Traversing the entire array: O(n). Searching for an element linearly: O(n). Sorting: O(n log n) with efficient algorithms.

7. When should you use arrays and when are alternatives like ArrayList better?

Answer:

Use arrays when the size is fixed and data type is primitive. Use ArrayList for dynamic size and object references, but with potential performance trade-offs.

8. What are common exceptions related to arrays and how to handle them?

Answer:

ArrayIndexOutOfBoundsException: Accessing an invalid index. Use proper bounds checking.

**ArrayStoreException`: Assigning incompatible data type. Ensure correct type

consistency.

9. How can you efficiently search, sort, or modify elements in an array?

Answer:

Consider built-in methods like Arrays.sort() for efficiency. For complex algorithms, understand their time and space complexities.

These are just examples, and specific questions will vary based on the interview.

Focus on understanding core concepts, common operations, and potential alternatives.

Be able to discuss performance implications and exception handling related to arrays.

Practice writing code examples to demonstrate your understanding.

I hope this helps you prepare for your interview!