**Popular Topics to Consider for a Java Interview**

* OOPS concepts (Data Abstraction, Encapsulation, Inheritance, Polymorphism)
* Basic Java constructs like loops and data types.
* String handling.
* Collection framework.
* Multithreading.
* Exception handling.
* Generics.
* Synchronisation.

What are the important Java programs asked in interviews?

In most programmer/developer interviews, candidates are required to complete real time coding tasks to help interviewers properly assess and verify their skills. Self-reported skills in resumes can often be unreliable, so live coding tasks allow interviewers to test a candidate's understanding of the coding language, problem-solving capabilities, speed and performance under pressure.

These are the important types of programs asked in Java interviews:

Basic programs

Number programs

Array programs

String programs

Matrix programs

Searching and sorting programs

Pattern programs

Linked list programs

Tree programs

Technical Skills: Definitions and Examples\*\*

Preparing for a Java interview

Before you appear for a Java interview, it is recommended that you develop a good understanding of all aspects of the language. Additional to your mastery of the language, interviewers can also test your code for its design, readability and efficiency. Apart from a formal education in computer sciences, there are several digital and printed resources that allow candidates to pick up coding skills on their own and further their understanding of coding languages like Java.

It is also good practice to prepare a portfolio of your works to share with potential interviewers. Your portfolio should ideally cover your qualifications, specific skills, work experience and notable projects.

50 common Java interview programs

Here is a list of Java programs commonly asked about during interviews:

**1. Fibonacci series**

Program to display any given number of integers of the Fibonacci series. In the Fibonacci series, each number is equal to the sum of the two numbers that precede it.

**2. Checking for prime number**

Program to verify whether a given number is a prime or composite.

**3. String palindrome**

Program to verify whether a given string is a palindrome or not. A string is a palindrome if it is equal to the reverse of itself.

**4. Integer palindrome**

Program to verify whether a given integer is a palindrome or not.

**5. Armstrong number**

Program to verify whether a given number is an Armstrong number. An Armstrong number is equal to the sum of the cubes of its digits.

**6. Avoiding deadlocks**

Program where a resource can be accessed by more than one thread, without encountering a deadlock. To avoid deadlocks, you have to procure resources in a certain order and ensure that they are released in reverse order.

**7. Factorial**

Program to calculate and display the factorial of any given number.

**8. Reversing strings (palindrome )**

Program to reverse the order of characters in any given string.

**9. Removing repeated elements from an array**

Program to identify and remove all repeated elements from an array. Arrays could be of various types like integer, character or string.

**10. Printing patterns**

Program to print a given pattern composed of ASCII characters. The solution must employ an effective algorithm, as opposed to directly printing the pattern as it is.

**11. Printing repetitive characters in a string**

Program to identify and print all repeated characters of a string.

**12. Finding the greatest common denominator of two numbers**

Program to print the greatest common denominator of any two given numbers.

**13. Finding the square root of a number**

Program to print a given number's square root **without employing the math.sqrt()** function.

**14. Reversing an array in place**

Program to reverse the order of elements in a string, integer or character array.

**15. Reversing the order of words in a sentence**

Program to reverse the order of words in any given sentence.

**16. Determining leap year**

Program to determine whether a given year is a leap year or not.

**17. Performing binary search**

Program to perform binary search for a given character/integer within a sorted array. In a binary search, the search interval is repeatedly divided into half with every iteration.

**18. Checking for anagrams**

Program to check if two given strings are anagrams. An anagram is of the same length and is composed of the same characters, but in a different order.

**19. Designing a vending machine**

Program to create a vending machine interface based on a given set of guidelines. This is a popular OOAD (Object-Oriented Analysis and Design) problem that is frequently asked in Java interviews.

**20. Reversing a number (Integer reverse)**

Program to print any given number with the digits in reverse order.

**21. Finding the first unique character of a string**

Program to identify and print the first unique character in any given string.

**22. Finding the middle element of a linked list**

Program to print the middle element of a linked list. You may be required to execute this with a single-pass algorithm, meaning the program will read the input only once before running iterations.

**23. Performing pre-order traversal**

Program to perform pre-order traversal of a hierarchical data structure. Hierarchical data structures like trees can be traversed in multiple ways. Linear data structures like arrays, linked lists, queues and stacks can only be traversed in one logical way.

**24. Performing pre-order traversal without recursion**

Program to perform pre-order traversal of a tree without recursion. Recursion in Java involves a function/method calling itself within the code.

**25. Performing in-order traversal**

Program to perform in-order traversal of a hierarchical data structure.

**26. Performing in-order traversal without recursion**

Program to perform in-order traversal of a tree, employing an iterative solution. Iteration, an alternative to recursion, involves a loop being repeatedly executed till a specific condition is met.

**27. Performing post-order traversal**

Program to perform post-order traversal of a hierarchical data structure.

**28. Performing post-order traversal without recursion**

Program to perform post-order traversal of a tree, employing an iterative solution.

**29. Printing all leaves of a binary tree**

Program to print the values of all leaves in a binary tree. Additionally, you may be asked to print the values in a certain order or hierarchy.

**30. Sorting an array using quick-sort**

Program to sort a given array of integers using the quick-sort algorithm.

**31. Performing insertion sort**

Program to sort a given array using the insertion sort algorithm.

**32. Performing bubble sort**

Program to sort a given array using the bubble sort algorithm.

**33. Transposing a matrix**

Program to print the transpose of a given matrix. A transpose of a matrix has all its rows and columns interchanged.

**34. Printing all permutations of a string**

Program to print all character arrangement permutations for any given string.

**35. Reversing a string in place**

Program to reverse the order of characters in any given string. However, the reversing has to be done in place, meaning the solution should not involve creating a duplicate string for reversal.( without using a extra variable or space)

**36. Adding matrices**

Program to add any two given matrices and print the result.

**37. Multiplying matrices**

Program to multiply any two given matrices and print the result.

**38. Removing spaces in a string**

Program to identify and remove all white spaces in a given string of characters.

**39. Reversing a linked list**

Program to reverse the order of contents of a singly linked list.

**40. Finding the length of a linked list**

Program to determine the length of a singly linked list in one iteration of the code.

**41. Checking for loops in a linked list**

Program to check whether a given linked list contains a loop or not. Corrupt linked lists can sometimes have two nodes pointing to the same location, thereby forming a cycle or loop.

**42. Finding the start of looping in a linked list**

Program to identify and print the starting instance of looping in a linked list.

**43. Finding the middle element of a linked list**

Program to determine and print the middle element(s) of a linked list.

**44. Finding the nth element from the tail of a linked list**

Program to find the nth element from the end of a given linked list, where n is a variable value provided by the user. You may be required to find a solution that does not involve multiple iterations.

**45. Converting a linked list to a binary tree**

Program to convert a doubly linked list to a binary tree.

**46. Sorting a linked list**

Program to sort a given linked list in ascending or descending order of values in each node.

**47. Performing bucket sort**

Program to sort the contents of a given array using the bucket sort algorithm. Bucket sort is a linear sorting algorithm that requires you to know the highest value present in the array before sorting it.

**48. Performing counting sort**

Program to sort the contents of a given array using the counting sort algorithm. Counting sort is also a linear sorting algorithm like bucket sort, and it helps to know the subtle differences between the two.

**49. Performing merge sort**

Program to sort the contents of a given array using the merge sort algorithm. You may have to provide a recursive or iterative solution, depending on what is required.

**50. Checking if two strings are rotations of each other**

Program to determine whether any two given strings are rotations of each other. For example, zxy is a rotation of xyz, but zyx is not.