Top 10 Coding Questions from Junior level Java Interviews

10 programming interview questions every Java developer should know.

JAVINPAUL AND SOMA

APR 15, 2024 • PAID

Hello guys, if you have 1 to 2 years of experience of you are fresher looking for a Java developer job then apart from preparing concept level questions like How ConcurrentHashMap work in Java? or Why String is Immutable, or why wait() and notify() is called from synchronized context, What is the difference between List, List<Object> , and CyclicBarrier vs CountDownLatch, you also need to be ready to solve coding problems.

The good thing is that they don't expect you to solve really tough coding questions like those which comes on Amazon, Microsoft, or Google Interviews. Instead most of the Java coding questions are revolved around popular programming exercises which every programmer should have done that during their college or learning journey.

Though, don't underestimate them because I have seen Java developer with 10 years of experience who are not able to reverse a linked list in a coding interview online. Call it a pressure of interview, or time pressure or may be just coding rust but that's not what you want if you want to succeed on interviews, hence you should prepare well.

*And, if need help with your interview preparation, you can also check my books, **Grokking the Java Interview** and **Grokking the Spring boot Interview** for better preparation, use code **friends20** to get a 20% discount also*

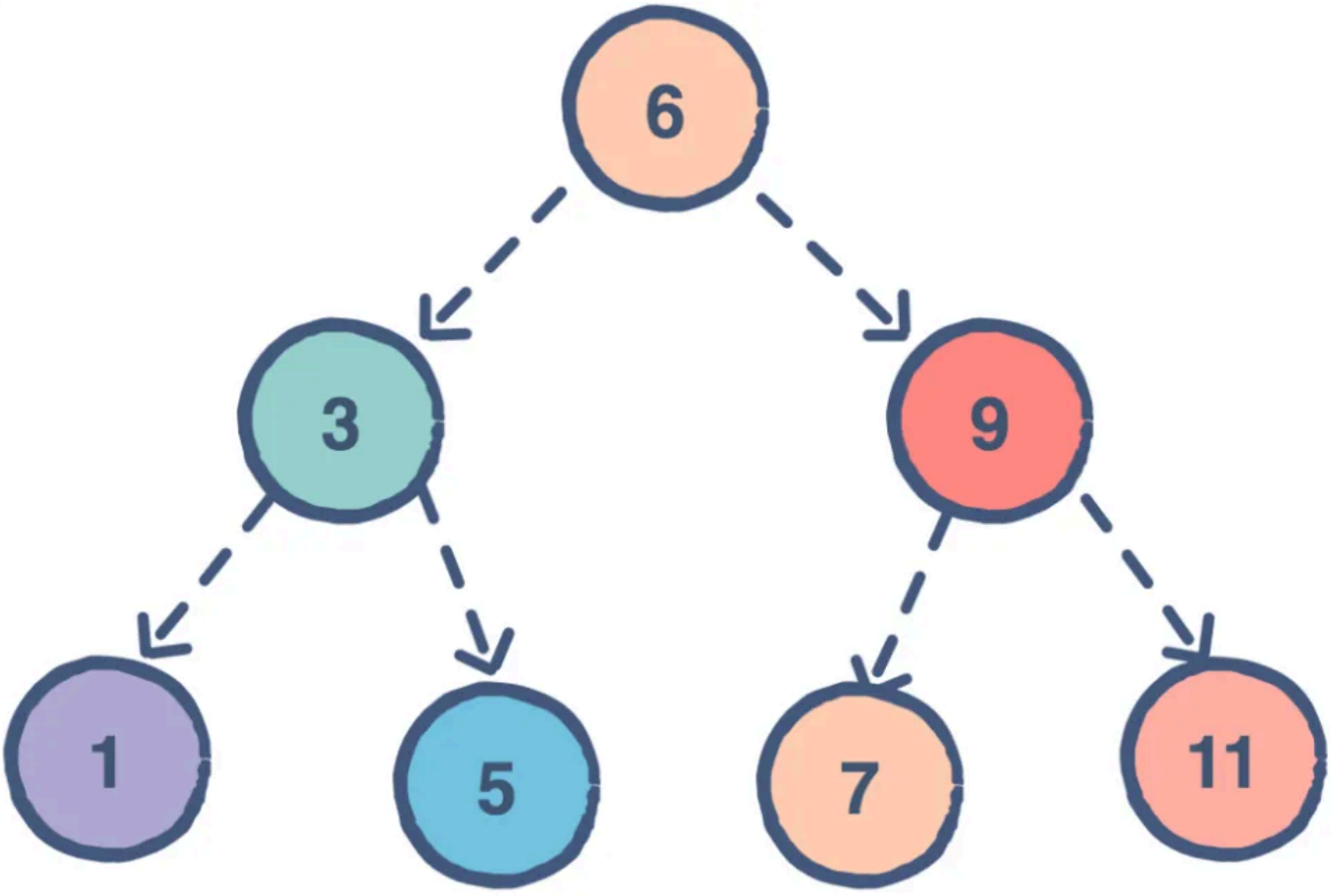
10 Common Coding Problems for Java Developers

Without any further ado, here are common coding questions every Java developer should know

1. **FizzBuzz**: This is a classic programming problem where you're asked to print numbers from 1 to N, but for multiples of 3, print "Fizz" instead of the number, and for multiples of 5, print "Buzz". For numbers which are multiples of both 3 and 5, print "FizzBuzz".
2. **Reverse a String**: Write a function to reverse a string in Java. This could be done iteratively, recursively, or using built-in methods.
3. **Palindrome**: Write a function to determine whether a given string is a palindrome (reads the same forwards and backwards), ignoring spaces, punctuation, and capitalization.

4. **Find Maximum Subarray Sum**: Given an array of integers, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.
5. **Binary Search**: Implement binary search algorithm both iteratively and recursively.
6. **Merge Sort**: Implement the merge sort algorithm to sort an array of integers.
7. **Linked List Operations**: Implement basic operations on a singly linked list such as insertion, deletion, and reversal.
8. **Fibonacci Series**: Write a function to generate the nth Fibonacci number, both recursively and iteratively.
9. **Find Duplicates in an Array**: Given an array of integers where each integer appears twice except for one integer which appears once, find and return the non-duplicated integer.
10. **Depth-First Search or Breadth-First Search**: Implement either DFS or BFS algorithm for a given graph represented either as an adjacency matrix or adjacency list.

These are just a few examples, but they cover a range of topics commonly tested in Java interviews. It's essential to not only know how to solve these problems but also to understand the underlying principles and algorithms behind them.



An example of a balanced tree

Solutions

Now, if you are looking for answers, while you can easily find them, I am also giving code solution here.

here are code solutions along with brief explanations for each of the mentioned problems:

1. FizzBuzz:

```
`for (int i = 1; i <= 100; i++) {

if (i % 3 == 0 && i % 5 == 0)

System.out.println("FizzBuzz");

else if (i % 3 == 0)

System.out.println("Fizz");

else if (i % 5 == 0)

System.out.println("Buzz");

else

System.out.println(i);

}`
```

Explanation: Iterate from 1 to 100, and for each number, check if it's divisible by 3, 5, or both, and print the appropriate message.

2. Reverse a String:

```
public String reverseString(String s) {

return new StringBuilder(s).reverse().toString();

}`
```

Explanation: Convert the string to a `StringBuilder`, use its `reverse()` method, and then convert it back to a string.

3. Palindrome:

```
public boolean isPalindrome(String s) {

s = s.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
```

```
return s.equals(new StringBuilder(s).reverse().toString());

}
```

Explanation: Remove non-alphanumeric characters and convert to lowercase, then compare the string with its reverse.

4. Find Maximum Subarray Sum:

```
`public int maxSubArray(int[] nums) {

int maxSum = nums[0];

int currentSum = nums[0];

for (int i = 1; i < nums.length; i++) {

currentSum = Math.max(nums[i], currentSum + nums[i]);

maxSum = Math.max(maxSum, currentSum);

}

return maxSum;

}`
```

Explanation: Iterate through the array, keeping track of the maximum sum of subarrays seen so far.

5. Binary Search:

```
`public int binarySearch(int[] arr, int target) {

int left = 0, right = arr.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target)

return mid;

else if (arr[mid] < target)
```

```
left = mid + 1;

else

right = mid - 1;

}

return -1;

}`
```

Explanation: Search for a target element in a sorted array by repeatedly dividing the search interval in half.

6. Merge Sort:

```
`public void mergeSort(int[] arr) {

if (arr.length <= 1) return;

int mid = arr.length / 2;

int[] left = Arrays.copyOfRange(arr, 0, mid);

int[] right = Arrays.copyOfRange(arr, mid, arr.length);

mergeSort(left);

mergeSort(right);

merge(arr, left, right);

}

private void merge(int[] arr, int[] left, int[] right) {

int i = 0, j = 0, k = 0;

while (i < left.length && j < right.length) {

if (left[i] <= right[j])

arr[k++] = left[i++];

else
```

```
arr[k++] = right[j++];

}

while (i < left.length)

arr[k++] = left[i++];

while (j < right.length)

arr[k++] = right[j++];

}`
```

Explanation: Recursively divide the array into halves, sort each half, and then merge the sorted halves.

7. Linked List Operations:

you would need to define a `ListNode` class and implement methods for insertion, deletion, and reversal operations.

8. Fibonacci Series:

```
public int fibonacci(int n) {

if (n <= 1) return n;

int a = 0, b = 1;

for (int i = 2; i <= n; i++) {

int temp = b;

b = a + b;

a = temp;

}

return b;

}`
```

Explanation: Generate the nth Fibonacci number iteratively using dynamic programming.

9. Find Duplicates in an Array:

```
public int findSingleNumber(int[] nums) {

int result = 0;

for (int num : nums)

result ^= num;

return result;

}`
```

Explanation: Use bitwise XOR to find the non-duplicated integer.

That's all about **common coding questions for Java interviews.** While they are the simplest problem you will encounter on Java interview, preparing them will give you confidence to tackle Java interviews. It's also important to prepare both concept-wise questions and coding questions to do well on Java interviews.

And, if need help with your interview preparation, you can also check my books, Grokking the Java Interview and Grokking the Spring boot Interview for better preparation, use code ***friends20*** *to get a 20% discount also*

5 Likes · 3 Restacks

← Previous

Next →

A guest post by

Soma

Java and React Developer

Subscribe to Soma

1 Comment

Write a comment...

Soma Apr 15 Author

Thanks for publishing

LIKE REPLY SHARE