INTERVIEW SOLUTION

For Answer Framing

"Ye Zindagi Na Milegi Dobara... Prepare Acche Se Karna..."

Because technical interviews don't give second chances easily!

Dr Kiran Waghmare
CDAC Mumbai

Types of Interview Questions

- **1. Technical Questions**
  - Core Programming
  - Coding & Problem Solving
  - System Design
  - Database & SQL
  - Web Technologies
  - DevOps & Tools
- **2. Behavioral & HR Questions**

STEP-BY-STEP NON-TECHNICAL INTERVIEW PREPARATION

- 1: Master Your Introduction (Elevator Pitch)
- 2: Prepare STAR-based Responses for Common Questions
- 3: Know Your Resume Inside-Out
- 4: Research the Company & Role
- 5: Practice Common HR Questions
- 6: Work on Communication & Soft Skills
- 7: Prepare a Few Questions to Ask the Interviewer

Technical Interview

Answer Framing Technique

STEP-BY-STEP TECHNICAL INTERVIEW PREPARATION

- **1: Understand the Interview Rounds**
  - Online coding test (HackerRank/CodeSignal)
  - DSA-focused technical rounds
  - OOP + Java core questions
  - System Design / Low-level Design
  - Project discussion (from your resume)
  - HR/Behavioral round

STEP-BY-STEP TECHNICAL INTERVIEW PREPARATION

- **2: Master Core Java (Object-Oriented + Internals)**
- **Focus Areas:**
- OOP Principles (Abstraction, Inheritance, Polymorphism, Encapsulation)
- Interfaces vs Abstract Classes
- Exception Handling
- Java 8 Features
- Multithreading (Thread, Runnable)
- Collections Framework (List, Set, Map, Queue)

Action Plan:

- Revise theory using Java docs or GeeksForGeeks
- Practice code snippets daily
- Prepare 3–4 real-world use cases for each concept

STEP-BY-STEP TECHNICAL INTERVIEW PREPARATION

- **3: DSA – Data Structures and Algorit**
- **Must-cover topics:**
  - Arrays, Strings, Matrix
  - LinkedList, Stack, Queue, HashTable
  - Recursion + Backtracking
  - Trees, BST, Heaps
  - Searching & Sorting
  - Graphs (BFS, DFS)
  - Dynamic Programming

Practice Routine:

- Solve 2 easy or 1 medium DSA problem daily on LeetCode or HackerRank
- Bookmark pattern-based problems
- Track time & space complexity for every solution
- Use whiteboard or notepad or pen-paper for dry runs

Q: What is the difference between HashMap and TreeMap in Java?

- **Structured Answer:**

  - Both are part of the Java Map interface.
  - HashMap offers constant-time performance for get() and put(), while TreeMap maintains keys in sorted order and uses O(log n)
  - HashMap is faster for general lookups, but TreeMap is better when sorted data is needed — for example, leaderboard systems or range queries.
  - One thing to remember: HashMap allows one null key, TreeMap does not.

Theoretical / Conceptual Questions

- **Purpose:** To test your understanding of **core concepts**, terminology, and how things work internally.

Examples:

- What is the difference between an interface and an abstract class in Java?
- How does garbage collection work in Java?
- What is the time complexity of a binary search?

Answer Strategy:

- Define the concept in 1–2 lines.
- Give an example or use-case.
- Mention key differences (if it's a comparative question).
- Add internal details or edge-case awareness if possible.

Coding / DSA Problems

Purpose: To test your **problem-solving ability, code logic, and efficiency**.

Examples:

- Reverse a LinkedList.
- Find the longest substring without repeating characters.
- Implement Insertion sort algorithm.

Answer Strategy:

- Clarify constraints and edge cases.
- Explain your approach before coding.
- Write clean, modular code.
- Mention time & space complexity.
- Walk through a dry run or input example.

Example 1

```
public ListNode List(ListNode head) {
    ListNode prev = null, curr = head;
    while (curr != null) {
        ListNode next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
```

Answer Approach:

Use three pointers: prev, curr, and next. Iterate and reverse the .next pointers.

Complexity:

Time: O(n)

Space: O(1)

Edge Cases:

Empty list

One node

Example 2

```
public boolean test(String s) {
    int left = 0, right = s.length() - 1;
    while (left < right) {
        if (s.charAt(left++) != s.charAt(right--))
         return false;
    }
    return true;
}
```

Answer Approach:

Use two-pointer technique from both ends. Compare characters.

Complexity:

- Time: O(n)
- Space: O(1)

Edge Cases:

- Empty string
- Case sensitivity or spaces (normalize input if required)

Advantages:

- Reduces time complexity significantly (from $O(n^2)$ to O(n))
- Easy to implement
- Great for sorted arrays, palindrome checks, partitioning

For Answer Framing

"Zindagi Na Milegi Dobara... Prepare Acche Se Karna...“

Because technical interviews don't give second chances easily!


THANK YOU