■ 5-Month Java DSA Roadmap

Designed for Java Backend Developers transitioning into high-performance and OpenJDK-level engineering.

Goal: Strengthen logic, algorithms, and problem-solving with Java	**Duration:** 5 Months (2–3 hrs/day)	**Focus:** Practical DSA for backend & system design thinking

Month 1 — Foundations & Arrays-Strings Mastery

■ **Goal:** Build solid problem-solving mindset & Java DSA base.

■ **Topics:** • Big O notation, time / space analysis
• Arrays (rotation, prefix sum, two-pointer)
• Strings (palindrome, sliding window, frequency map)
• Recursion + Backtracking basics
• Java collections (List, Set, Map internals)

■ **Practice:** Solve ~25–30 problems (Two Sum, Kadane's, Merge Intervals, Longest Substring, Subarray Sum=K).

■ **Resources:** 'Data Structures & Algorithms Made Easy in Java' by Narasimha Karumanchi, Kunal Kushwaha DSA Series.

■ **Milestone:** Be able to dry-run & optimize array/string problems confidently.

Month 2 — Linked List, Stack & Queue

■ **Goal:** Master pointer logic + stack/queue operations.

■ **Topics:** • Linked List (single, double, cycle detection, reverse)
• Stack (infix→postfix, min-stack, valid-parentheses)
• Queue & Deque (circular queue, sliding window maximum)

■ **Practice:** 30 problems (reverse LL, detect cycle, LRU cache, valid parentheses).

■ **Resources:** Visualgo.net, LeetCode Medium section.

■ **Milestone:** Understand memory flow & trace object references in Java.

Month 3 — Trees, BST & Hashing

■ **Goal:** Understand recursion deeply & non-linear DS.

■ **Topics:** • Binary Tree (DFS, BFS, height, diameter, level order)
• BST (insert/search/delete, inorder successor)
• HashMap & HashSet deep dive (rehashing, collisions)

■ **Practice:** 35–40 problems (LCA, Path sum, Top view, Pair sum using hashmap).

■ **Milestone:** Design efficient hierarchical & lookup structures.

Month 4 — Graphs, Heaps & Greedy

■ **Goal:** Strengthen advanced logic & algorithmic design.

■ **Topics:** • Graph representation (adjacency list/matrix)
• BFS/DFS, connected components
• Dijkstra, Topological Sort, Union Find
• PriorityQueue, Greedy algorithms

■ **Practice:** 35–40 problems (shortest path, heap merge K sorted lists).

■ **Milestone:** Confidently solve complex logic and graph-based problems.

Month 5 — Dynamic Programming & System Thinking

■ **Goal:** Develop algorithmic mindset & optimization skill.

■ **Topics:** • DP fundamentals (memoization/tabulation)
• Classic DP (Fibonacci, Knapsack, LIS, LCS)
• Sliding window advanced patterns
• Bit manipulation & math (GCD, sieve)

■ **Practice:** 40–45 problems focusing on DP state transitions.

■ **Milestone:** Break down unseen problems and design efficient solutions.

■ Weekly Routine

Mon–Wed	Learn 1 concept + solve 3 easy/medium problems
Thu–Fri	Solve 2–3 medium/hard problems + review mistakes
Sat	Mock test (LeetCode contest or GFG quiz)
Sun	Revise notes + write mini-summary

■ Supplementary Skills

• Use debugger & profiler in IntelliJ to visualize runtime behavior.
• Implement LinkedList, Stack, HashMap manually once.
• Explore Java's Concurrent package (Atomic, Locks) — connects DSA to backend performance.

■ After 5 Months

■ Solve 150–180 DSA problems with strong confidence.
■ Understand Java's internal data structure design.
■ Confidently discuss algorithmic choices in backend architecture & OpenJDK-level projects.