# **Complete Java + DSA Bootcamp Syllabus**

## NOTE:

- All topics will contain problems from LeetCode Easy to Hard, explained in an easy to understand manner.

- Complete Custom Implementation of all Data Structures and Algorithms.

## Lectures

- Introduction to Git

- Introduction to Programming

- Types of languages

- Flowcharts & Pseudocode

- Flow of the program

- Introduction to Java

- Introduction

- How it works

- Setup Installation

- Input and Output in Java

- Data-types

- Coding best practices

- Arrays

- Introduction

- Memory management

- Input and Output

- ArrayList Introduction

- Sorting

- Insertion Sort

- Selection Sort

- Bubble Sort

- Count Sort

- Radix Sort

- Searching

- Linear Search

- Binary Search

- Modified Binary Search

- (more later)

- Sliding window

- Two Pointer

- Subarray Questions

- Strings

- Introduction

- How Strings work

- Comparison of methods

- Operations in Strings

- StringBuilder in java

- Maths for DSA

- Introduction

- Complete Bitwise Operators

- Prime numbers

- HCF / LCM

- Sieve of Eratosthenes

- Newton's Square Root Method

- Number Theory

- Euclidean algorithm

- Advanced Concepts for CP (later in course)

- Bitwise + DP

- Extended Euclidean algorithm

- Modulo Properties

- Modulo Multiplicative Inverse

- Linear Diophantine Equations

- Fremat's Theorem

- Wilson's Theorem

- Lucas Theorem

- Chinese Remainder Theorem

- Functions

- Introduction

- Solving the above math problems in code

- Scoping in Java

- Shadowing

- Variable Length Arguments

- (more content in OOP such as overloading etc)

- Space and Time Complexity Analysis

- Introduction

- Comparion of various cases

- Solving Linear Recurrence Relations

- Solving Divide and Conquer Recurrence Relations

- Big-O, Big-Omega, Big-Theta Notations

- Get equation of any relation easily - best and easiest approach

- Complexity discussion of all the problems we do

- Space Complexity

- Memory Allocation of various languages

- NP Completeness and Hardness

- Recursion

- Introduction

- Why recursion?

- Flow of recursive programs - stacks

- Convert recursion to iteration

- Tree building of function calls

- Tail recursion

- Sorting:

- Merge Sort

- Quick Sort

- Cyclic Sort

- Backtracking

- Sudoku Solver

- N-Queens

- N-Knights

- Maze problems

- Recursion String Problems

- Recursion Array Problems

- Recursion Pattern Problems

- Subset Questions

- Object Oriented Programming

- Introduction

- Classes & its instances

- this keyword in Java

- Properties

- Inheritance

- Abstraction

- Polymorphism

- Encapsulation

- Overloading & Overriding

- Static & Non-Static

- Access Control

- Interfaces

- Abstract Classes

- Singleton Class

- final, finalize, finally

- Exception Handling

- Stacks & Queues

- Introduction

- Interview problems

- Push efficient

- Pop efficient

- Queue using Stack and Vice versa

- Circular Queue

- Linked List

- Introduction

- Fast and slow pointer

- Cycle Detection

- Single and Doubly LinkedList

- Reversalof LinekdList

- Dynamic Programming

- Introduction

- Recursion + Recursion DP + Iteration + Iteration Space Optimized

- Complexity Analysis

- 0/1 Knapsack

- Subset Questions

- Unbounded Knapsack

- Subseq questions

- String DP

- Trees

- Introduction

- Binary Trees

- Binary Search Trees

- DFS

- BFS

- AVL Trees

- Segment Tree

- Fenwick Tree / Binary Indexed Tree

- Square Root Decomposition

- Heaps

- Introduction

- Theory

- Priority Queue

- Two Heaps Method

- k-way merge

- top k elements

- interval problems

- Hashmaps

- Introduction

- Theory - how it works

- Comparisons of various forms

- Limitations and how to solve

- Map using LinkedList

- Map using Hash

- Chaining

- Probing

- Huffman-Encoder

- Tries

- Graphs

- Introduction

- BFS

- DFS

- Working with graph components

- Minimum Spanning Trees

- Kruskal Algorithm

- Prims Algorithm

- Dijkstra’s shortest path algorithm

- Topological Sort

- Bellman ford

- A\* pathfinding Algorithm