Roadmap Day	Module	Title	Content	Session Activities	Duration	Reference Notes
1	Programming Foundation	Introduction to Java & Setup	**Intro to Java:** Overview of Java Platform (JVM, JRE, JDK). Key Features: Platform Independence, Object-Oriented, Robust. Setting up the Development Environment (JDK 21, IDE - IntelliJ IDEA). **Java Syntax & First Program:** Structure of a Java Program. The main method. Keywords, Identifiers, and Coding Conventions. Comments. **Variables & Primitive Data Types:** Variable Declaration & Initialization. Primitive Types: byte, short, int, long, float, double, char, boolean.		3	
2	Programming Foundation	Control Flow, Loops & Methods In Java	**Control Flow:** Conditional Statements: if, if-else, if-else-if ladder. switch statement (traditional and with arrows in Java 14+). **Loops:** for loop, while loop, do-while loop. Loop Control Statements: break, continue, labels. **Methods in Java**: Defining Methods, Parameters, and Return Types. Method Signature. Method Overloading (Compile-time Polymorphism).	Build a simple grade calculator using if-else. Create a number guessing game using loops. Print patterns (e.g., pyramids) using nested loops	3	
3	Programming Foundation	Arrays ,String API & Wrapper Classes	**Arrays:** Single-dimensional Arrays. Accessing, iterating (for, for-each). Multi-dimensional Arrays (Introduction) **String Class:** String Immutability. Important Methods: charAt(), length(), substring(), equals(), indexOf() etc. StringBuilder vs StringBuffer . **Wrapper Classes:** Purpose of Wrapper Classes (Integer, Double, Character, etc.). Autoboxing and Unboxing.	Write methods to perform operations on arrays: find max/min, average, search for an element	3	
4	Programming Foundation	Intro to OOP: Classes & Objects and OOps Features	**Intro to Classes and Objects:** Principles of Object-Oriented Programming. Defining a Class, Creating Objects. Constructors (Default and Parameterized). The this keyword. **Encapsulation:** Access Modifiers (public, private, protected). Implementing Getters and Setters. **Inheritance**: extends keyword. super keyword (for accessing parent class members and constructors).	Model a real-world entity like a BankAccount or Employee class with fields and behaviors. Ensure all fields are private and accessed via getters/setters.	3	

5	Programming Foundation	Core OOP: Polymorphism,Abstraction,Interfaces& Packa	**Polymorphism:** Method Overriding (Runtime Polymorphism). Rules for overriding. @Override annotation **Abstraction**: Abstract Classes. Abstract Methods. **Interfaces**: Defining and Implementing Interfaces. Default and Static Methods (Java 8+). Multiple Inheritance using Interfaces. **Packages:** Organizing classes into packages. import statements.	Create an inheritance hierarchy (e.g., Vehicle -> Car, Bike). Demonstrate polymorphism by storing different subclass objects in a parent class reference.	3	
6	Programming Foundation	Exception Handling	**Exception Handling**: Checked vs. Unchecked Exceptions. try, catch, finally blocks. throw and throws keywords. Creating Custom Exception Classes	Write code that handles FileNotFoundException, ArithmeticException, etc. Create a custom InsufficientFundsException for the BankAccount class.	3	
7	Programming Foundation	Collections Framework	**Collection Framework Overview:** Core Interfaces: Collection, List, Set, Map, Queue. **List & Set Interface:** ArrayList (resizable array). LinkedList (doubly-linked list). Iterating with Iterator and for-each. HashSet (unordered, unique elements). LinkedHashSet (ordered iteration). **Map Interface, Queue Interface &PriorityQueue (intro).** HashMap (key-value pairs). LinkedHashMap (maintains insertion order).	Build a simple student management system using an ArrayList to add, remove, and list students. Compare performance of ArrayList vs. LinkedList for adding elements at the beginning.	3	
8	Programming Foundation	Generics & Lambda Expressions	**Generics:** The need for Generics (Type Safety, eliminating casts). Generic Classes and Methods. Bounded Type Parameters (<T extends Number>). Wildcards (?, ? extends T, ? super T). **Lambda Expressions:** Functional Interfaces (Runnable, Comparator, custom). Syntax of Lambda Expressions. Method References (Class::method).	Create a generic Box<T> class. Write a generic method to print all elements of any List.	3	
9	Programming Foundation	File I/O & Concurrency	**File I/O :** Files utility class for reading/writing. **Multi-threading (Basics):** Creating threads: extending Thread vs. implementing Runnable. Thread lifecycle.	Write a program to read a configuration file and write results to an output file. Create two threads: one to print even numbers, another to print odd numbers.	3	

10	Programming Foundation	Java Advanced Topics & Features	**Memory Management (Overview):** Stack vs. Heap memory. Introduction to Garbage Collection. **Java Features (8 to 21):** Highlights: var (Local Variable Type Inference), record, sealed classes, Text Blocks. **Annotations & Enums:** Built-in annotations (@Override). Defining and using enum types. **Internalization (I18N) - Overview:** Locale class. ResourceBundle for externalizing strings.		3	
11	DSA	Introduction to DSA	**Introduction to DSA:** What is DSA? Why DSA for backend engineers? Time and Space Complexity Analysis (Big O Notation)	Session Actitvity: Analyze the time and space complexity of a function that checks whether a given array contains a duplicate element.	3	
12	DSA	Arrays and Strings	**Arrays and Strings:** 1D & 2D Arrays Common Array Problems String Manipulation Techniques StringBuilder & StringBuffer Sliding Window & Two Pointer Techniques		3	
13	DSA	Recursion and Backtracking	**Recursion and Backtracking:** Introduction to Recursion Recursive Tree Patterns Backtracking Problems (e.g., N-Queens, Sudoku Solver) Memoization Basics		3	
14	DSA	Practice Session	Codekata & Leetcode Problems:Arrays,Strings,Recursion & Ba	Session Activity: LeetCode Problems: Two Sum Best Time to Buy and Sell Stock Move Zeroes Subsets Permutations N-Queens	3	
15	DSA	Searching and Sorting	**Searching and Sorting:** Linear & Binary Search Binary Search on Answers Bubble, Selection, Insertion Sort		3	
16	DSA	Searching and Sorting	**Searching and Sorting** Merge Sort & Quick Sort Counting Sort, Radix Sort (Basics) Comparator and Comparable Interfaces		3	
17	DSA	Practice Session	Codekata & Leetcode Problems:Searching & Sorting	Session Activity: LeetCode: Binary Search Search a 2D Matrix Top K Frequent Elements Median of Two Sorted Arrays	3	

18	DSA	Linked Lists	**Linked Lists** Singly and Doubly Linked List Fast & Slow Pointers Reversal Techniques Detecting and Removing Cycles Merge Two Sorted Lists, Middle Node		3	
19	DSA	Stacks and Queues	**Stacks and Queues** Stack using Arrays and LinkedList Queue and Deque Implementations Infix, Prefix, Postfix Expressions Cont. Monotonic Stack/Queue Problems LRU Cache (LinkedHashMap & Deque)		3	
20	DSA	Practice Session	Codekata & Leetcode Problems:Linked Lists,Stacks & Queues	LeetCode: Reverse Linked List Merge Two Sorted Lists Linked List Cycle Valid Parentheses Min Stack	3	
21	DSA	Hashing	**Hashing** HashMap, HashSet, Hashtable Frequency Counting Problems Grouping Anagrams, Two Sum, Subarray Sums Custom Hashing Techniques		3	
22	DSA	Trees and Binary Trees	**Trees and Binary Trees** Binary Tree Basics Tree Traversals (Inorder, Preorder, Postorder) Level Order & Zigzag Traversals Height, Diameter, Balanced Trees Constructing Trees from Traversals		3	
23	DSA	Binary Search Trees (BST)	**Binary Search Trees (BST)** BST Operations (Insert, Delete, Search) Inorder Successor & Predecessor BST Validation Kth Smallest Element Convert Sorted Array/List to BST		3	
24	DSA	Heaps and Priority Queues	**Heaps and Priority Queues** Min Heap and Max Heap Implementation PriorityQueue Class in Java Heap Sort Top K Elements Problems Median of a Stream		3	
25	DSA	Practice Session	Codekata & Leetcode Problems:Trees and Binary Trees,BST, Heaps and Priority Queues	Session Activity: Leetcode: Maximum Depth of Binary Tree Symmetric Tree Lowest Common Ancestor Kth Largest Element Merge K Sorted Lists	3	

26	DSA	Graphs	**Graphs** Graph Representations (Adjacency List/Matrix) BFS and DFS Traversals Detect Cycle in Undirected/Directed Graph Topological Sorting Shortest Path: Dijkstra & BFS-based Connected Components Union-Find (Disjoint Set Union)		3	
27	DSA	Greedy Algorithms	**Greedy Algorithms** Greedy Strategy Design Activity Selection Job Scheduling Fractional Knapsack Huffman Encoding (Intro)		3	
28	DSA	Practice Session	Codekata & Leetcode Problems:Graphs,Greedy Algorithms	Session Activity: LeetCode: Number of Islands Clone Graph Jump Game Gas Station	3	