JAVA FULLSTACK

Introduction to Java

- History & Evolution of Java
- Features of Java (Platform Independent, OOP, etc.)
- JDK, JRE, JVM
- Java Editions (SE, EE, ME)
- Java Program Structure
- How Java Works (Compilation & Execution)

Java Basics

- Data Types & Variables

-

- Operators
- Control Flow (if, switch, loops)
- Arrays
- Methods

  Input/Output in Java
- Packages & Access Modifiers

Object-Oriented Programming (OOPs) in Java

- Classes & Objects
- Constructors
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- Method Overloading & Overriding
- ‘this’ & ‘super’ Keyword
- Association, Aggregation & Composition
- Static & Final Keywords

Exception Handling

- Types of Exceptions (Checked & Unchecked)
- try-catch Block
- finally Block
- throw & throws Keyword
- Custom Exceptions
- Best Practices for Exception Handling

Java Collections Framework (JCF)

- Collection Hierarchy
- List (ArrayList, LinkedList)
- Set (HashSet, LinkedHashSet, TreeSet)
- Map (HashMap, TreeMap, LinkedHashMap)
- Queue & Deque (PriorityQueue, ArrayDeque)
- Iterator & ListIterator
- Comparable vs Comparator
- Collections Utility Class
- Best Practices & Performance Considerations

Multithreading & Concurrency

- Threads & Lifecycle
- Creating Threads (Extending Thread / Implementing Runnable)

-

- Thread Priorities
- Synchronization (synchronized keyword, blocks)
- Inter-Thread Communication (wait, notify)
- Deadlock, Starvation & Livelock
- Executor Framework
- Callable & Future
- Concurrency Utilities (Locks, Semaphores, CountDownLatch)

  Fork/Join Framework

Streams

- Introduction to Streams API
- Creating Streams
- Intermediate Operations (filter, map, sorted)
- Terminal Operations (forEach, collect, reduce)
- Parallel Streams
- Optional Class
- Best Practices for Stream Operations

Lambda Expressions & Functional Programming

- Introduction to Lambda Expressions
- Syntax & Functional Interfaces
- Built-in Functional Interfaces (Predicate, Function, Consumer, Supplier)
- Method References & Constructor References
- Streams with Lambdas
- Default & Static Methods in Interfaces
- Functional Programming Concepts in Java

Spring / Spring Boot (If Required for Project Development)

•

•

Introduction to Spring Framework
Inversion of Control (IoC) & Dependency Injection (DI)

- Spring Core & Bean Lifecycle
- Spring Boot Introduction & Setup
- Spring MVC (Controllers, REST APIs)
- Spring Data JPA
- Security (Spring Security Basics)
- Configuration (Properties, YAML, Profiles)
- Exception Handling in Spring
- Testing in Spring Boot
- Actuator & Monitoring
- Microservices Basics (Optional)

JPA / Hibernate

- Introduction to ORM
- JPA vs Hibernate
- Entities & Annotations
- CRUD Operations

-
-

- JPQL & Criteria API
- Relationships (OneToOne, OneToMany, ManyToOne, ManyToMany)
- Cascade & Fetch Types
- Transactions & EntityManager
  Caching & Performance Tuning
  Spring Data JPA Integration
- Native Queries
-

Java Deployment & Best Practices

- Packaging (JAR, WAR, EAR)
- Build Tools (Maven / Gradle)
- Environment Setup & Profiles
- Logging (Log4j, SLF4J)
- Code Quality & Linting Tools
- Unit Testing (JUnit, Mockito)
- Continuous Integration / Continuous
Deployment

-
-

(CI/CD) Basics

- Version Control (Git)
- Exception & Error Handling Strategies
- Secure Coding Practices
- Performance Optimization Tips
- Documentation & Code Comments

React Or Angular React:

- JSX
- Components (Functional & Class)

Props & State

Lifecycle Methods

- Hooks (useState, useEffect, etc.)
- Forms & Validation
- Event Handling
- Routing (React Router)
- Context API & Redux
- API Integration (Fetch / Axios)
- Custom Hooks
- Performance Optimization **Angular:**

-
-

- TypeScript Basics
- Components & Templates
- Modules & Services
- Dependency Injection
- Data Binding (One-way, Two-way)
- Directives & Pipes
- Routing & Navigation
- Forms (Template-driven & Reactive)
- HTTP Client & API Calls
- Observables & RxJS
- State Management (NgRx)
- Testing (Karma, Jasmine)