Java Developer Training

Basics of Java Programming(Brief Refresher)

- Overview of the Java Ecosystem
- Basic Syntax and Data Types
- Control Structures: If-Else, Loops, Switch
- Functions and Methods in Java
- Classes, Objects, and Constructors
- Inheritance and Polymorphism
- Encapsulation and Abstraction
- Interfaces and Abstract Classes

Comparison Between C++ and Java

- Key Differences Between C++ and Java
- Memory Management: Manual vs. Automatic
- Object-Oriented Programming: Similarities and Differences
- Comparing Data Structures in C++ and Java

Memory Management and Garbage Collection

- Basics of Java Memory Management
- Java's Memory Model: Heap and Stack
- Components of the JVM Memory
- Garbage Collection in Java
- Introduction to Garbage Collection
- Types of Garbage Collectors: Serial, Parallel, CMS, G1

- Garbage Collection Algorithms
- How Garbage Collection Works
- Generational Garbage Collection
- Tuning Garbage Collection
- Understanding Garbage Collection Metrics
- JVM Flags for GC Tuning
- Analyzing Garbage Collection Logs

Threads and Concurrency in Java

- Basics of Java Threads
- What is a Thread?
- Creating Threads: Thread Class vs. Runnable Interface
- Thread Lifecycle and Management
- Thread States: New, Runnable, Blocked, Waiting, Timed Waiting, Terminated
- Starting and Joining Threads
- Understanding Multithreading
- Creating and Managing Threads
- Synchronization and Locks
- Concurrency Best Practices

Java Collections Framework

- Overview and Importance of Collections
- Core Interfaces: **List**, **Set**, **Map**, **Queue**
- Implementations: **ArrayList**, **LinkedList**
- When to Use Each Implementation
- List Iteration Techniques

- Implementations: **HashSet**, **LinkedHashSet**, **TreeSet**
- Characteristics of Different Set Implementations
- Choosing the Right Set for Your Use Case
- Implementations: **HashMap**, **LinkedHashMap**, **TreeMap**
- Keys and Values in Maps
- Iteration Over Maps
- Implementations: **LinkedList**, **PriorityQueue**, **ArrayDeque**
- Differences Between FIFO and Priority-Based Queues
- Use Cases for Different Queue Types

Common Collection Operations

- Sorting and Searching within Collections
- Filtering and Transforming with Streams
- Using Comparator and Comparable Interfaces

Generics in Collections

- Understanding Generics in Java
- Benefits of Using Generics with Collections
- Common Generic Patterns and Restrictions

Tools for Java Development

- Integrated Development Environments (IDEs): IntelliJ, Eclipse, NetBeans, VSCode
- Build Tools: Maven and Gradle
- Version Control Systems: Git
- Other Useful Tools (e.g., Static Analysis Tools, Profiling Tools)

JVM Internals

- Java Virtual Machine (JVM) Basics
- Class Loading and Bytecode
- Just-In-Time (JIT) Compilation
- JVM Flags and Tuning

Code Review and Best Practices

- Importance of Code Reviews
- Conducting Effective Code Reviews
- Common Code Review Mistakes to Avoid
- Tools for Code Review and Collaboration

Introduction to JUnit

- Overview of Unit Testing
- Setting Up JUnit in a Java Project

Writing Unit Tests with JUnit

- Creating Test Classes and Test Methods
- Using Annotations: **@Test**, **@BeforeEach**, **@AfterEach**
- Assertions in JUnit: **assertEquals**, **assertTrue**, etc.

Testing Techniques with JUnit

- Parameterized Tests
- Exception Testing with **@Test(expected = Exception.class)**

- Nested Tests with **@Nested**

Advanced JUnit Concepts

- Lifecycle Management with **@BeforeAll** and **@AfterAll**
- Using Test Suites to Group Tests
- Conditional Test Execution with **@EnabledIf** and Similar Annotations

Introduction to Mockito

- What is Mockito?
- Importance of Mocking in Unit Testing
- Setting Up Mockito in a Java Project

Mocking with Mockito

- Creating Mocks with **Mockito.mock()**
- Mocking Method Calls with **Mockito.when()**
- Using **Mockito.verify()** to Verify Interactions

Mockito Advanced Techniques

- Argument Matchers with **Mockito.any()**, **Mockito.eq()**, etc.
- Mocking Static Methods with **Mockito.mockStatic()**
- Spying on Real Objects with **Mockito.spy()**

Performance Tuning and Profiling in Java

- Identifying Performance Bottlenecks
- Profiling Tools and Techniques

- Basic Performance Tuning Strategies
- JVM Flags for Performance Optimization

Introduction to Spring Boot

- What is Spring Boot?
- History and Evolution of Spring Framework
- Benefits of Using Spring Boot

Setting Up a Spring Boot Project

- Creating a Spring Boot Project
- Maven and Gradle for Spring Boot Projects
- Project Structure and Configuration

Core Spring Boot Components

- Main Application Class and **@SpringBootApplication**
- Auto-Configuration and Spring Boot Starters
- Application Properties and Configuration

Dependency Injection and Beans

- Basics of Dependency Injection
- Defining Beans with **@Bean** and Component Scanning
- Scope of Beans: Singleton, Prototype, etc.

Spring Boot with RESTful APIs

- Creating REST Controllers with **@RestController**
- Handling HTTP Requests: **@GetMapping**, **@PostMapping**, etc.

- Data Serialization and Deserialization with Jackson

Data Persistence in Spring Boot

- Introduction to Spring Data JPA
- Configuring Data Sources and Entity Relationships
- Working with Repositories and Custom Queries

Security in Spring Boot

- Introduction to Spring Security
- Configuring Basic Security Settings
- Authentication and Authorization with Spring Security

Advanced Spring Boot Topics

- Asynchronous Processing with **@Async**
- Building Event-Driven Applications with Spring Boot
- Using Spring Boot with Microservices Architectures