Enterprise Java Full Stack Development: Spring Boot, Angular, Microservices, Databases, Kafka, Redis & Redpanda Connect

Duration: 12 days

Prerequisites

- Basic computer literacy and familiarity with Windows or Linux operating systems.
- Understanding of programming fundamentals (variables, data types, operators, loops, and conditional statements) in any programming
- language.
- Basic knowledge of object-oriented programming concepts (classes, objects, methods, and inheritance) is preferred.
- Basic understanding of databases and SQL concepts is beneficial but not mandatory.
- Familiarity with HTML, CSS, and JavaScript fundamentals is an advantage for the Angular module.
- Ability to install and configure development tools such as JDK, IDE (IntelliJ IDEA/Eclipse/VS Code), and database software.

Programming in C (4)

S.No	Topics	Detailed contents
1	**Foundations of C**	
	Language Introduction	Setting up the compiler (GCC) and the structure of a standard C program The compilation process: Source code to executable object file
	Variables & Data Types	Primitive types: int, char, float, and double Formatted Input/Output using standard printf() and scanf() functions
	Operators & Expressions	Arithmetic, relational, logical, and assignment operations Evaluating operator precedence and implicit/explicit type casting
2	**Control Flow & Functions**	
	Conditional Statements	Decision making using if, else if, else, and nested conditions Multi-way branching execution via the switch-case statement
	Looping Structures	Iteration logic with bounded for loops and conditional while loops Controlling loop execution paths with break and continue keywords
	Modular Functions	Defining functions, return types, function prototypes, and arguments Understanding scope differences: Local variables versus Global variables

3 Data Structures & Pointers

Arrays & Matrices	Declaring, initializing, and indexing single-dimensional arrays Multidimensional layouts: Creating and traversing 2D matrices
Character Strings	Managing text using null-terminated (\0) character arrays Core string operations utilizing standard strlen, strcpy, and strcmp
Pointers Basics	Understanding memory addresses and using the reference operator (&) Declaring pointer variables and dereferencing them via the (*) operator

Programming in C++ (4)

1 C++ Foundations & Syntax

Language Overview	Setting up the C++ environment (GCC/Clang) and the main function structure Utilizing standard input/output streams via std::cout and std::cin
Core Data Types & Control	Declaring primitive types, working with std::string, and variable scoping Implementing conditional logic (if-else, switch) and iteration (for, while)

2 Object-Oriented Programming

Classes & Objects	Defining classes, constructing objects, and managing access modifiers (public, private) Working with constructors, destructors, and initialization lists
OOP Pillars in C++	Implementing single and multiple inheritance structures Achieving runtime polymorphism using virtual functions and abstract classes

3 Advanced Memory & Pointers

Pointers & References	Understanding memory addresses, pointer arithmetic, and dereferencing Using references (&) as aliases and passing parameters by reference
Dynamic Memory Management	Allocating and deallocating memory manually via new and delete operators Modern memory management using smart pointers (std::unique_ptr, std::shared_ptr)

Java Programming (16)

1 Java Fundamentals

Introduction & Environment Setup	JDK, JRE, and JVM architecture Writing, compiling, and running a Java program

	Language Basics	Variables, data types, and operators Control flow statements (if-else, switch, loops)
	Arrays & Strings	One-dimensional and multidimensional arrays String handling and core String methods

2 Object-Oriented Programming (OOP)

	Classes & Objects	Class definitions, object creation, and constructors Memory allocation and the this keyword
	The Pillars of OOP	Encapsulation and access modifiers Inheritance, polymorphism, and abstraction
	Interfaces & Packages	Defining and implementing interfaces Organizing code with packages

3 Core Advanced Features

	Exception Handling	try-catch-finally blocks Built-in and custom exceptions
	Java Collections Framework	Lists (ArrayList, LinkedList) Sets (HashSet) and Maps (HashMap)
	File I/O & Basic Streams	Reading and writing files Introduction to Stream API basics

Spring Boot (16)

1 Core Spring Boot and Microservice Foundations

	Introduction to Spring Boot and Microservices:	Why microservices? Benefits and challenges Overview of Spring Boot and its role in microservice development Key components of a Spring Boot application
	MVC Architecture Essentials:	Understanding the Model-View-Controller pattern Applying MVC principles to microservices Structuring a Spring Boot application for clarity and maintainability
	Microservice for Code Verification:	Requirements gathering: Defining expected criteria for code verification Designing the service: Input validation, business logic, and result evaluation Implementing RESTful APIs for microservice communication

2 Verification, Validation, and MVC Architecture

	Verification and Validation Techniques:	Writing unit tests for core functionalities Leveraging Spring Boot tools for input validation and error handling Monitoring and logging verification results

	Hands-on Development:	Building a microservice to validate generated codes Applying constraints and business logic checks Deploying and testing the microservice in a simulated environment
	Best Practices and Real-World Challenges:	Ensuring scalability and performance in Spring Boot microservices Addressing common pitfalls in MVC-based microservice design Evolving microservices to meet new verification requirements
3	**Introduction to REST API Principles**	
	Understanding REST Architecture	Principles of REST: Statelessness, client-server separation, uniform interfaces. HTTP methods: GET, POST, PUT, DELETE, PATCH, and their appropriate use cases. API design best practices: Resource naming, versioning, and HATEOAS.
	Building REST APIs with Spring Boot	Setting up a Spring Boot project. Creating RESTful endpoints and mapping HTTP methods to Java methods. Handling request parameters, path variables, and request bodies.
	Error Handling in REST APIs	Custom exception handling in Spring Boot. Designing standard error responses. Implementing global exception handling using @ControllerAdvice.
4	**Advanced REST API Development**	
	Securing REST APIs	Authentication and Authorization: OAuth2 and JWT basics. Implementing security in Spring Boot using Spring Security. Protecting APIs from common vulnerabilities (e.g., CSRF, XSS).
	Data Serialization and Validation	Using Jackson for JSON serialization and deserialization. Validating API requests with @Valid and custom annotations.
	Optimizing REST APIs	Pagination and filtering for large datasets. Caching responses to improve performance. Using asynchronous processing for long-running requests.
	Testing and Documentation	Writing unit tests for REST APIs using JUnit and Mockito.

		Automating API testing with Postman and REST Assured. Generating API documentation with Swagger/OpenAPI.

Databases (Oracle / PostgreSQL / MySQL) (12)

1	**Introduction and Installation**	
	Introduction and Installation	Introduction and architecture concepts
		Installing PostgreSQL and using pgAdmin / psql Installing MySQL Server, Workbench, and command-line shell Setting up SQL Developer and connecting to Oracle Database
	Fundamental SQL Syntax	Working with numeric, string, and date data types Executing CRUD operations (INSERT, SELECT, UPDATE, DELETE)
	Sorting & Filtering Data	Filtering records using WHERE, operators, and pattern matching
		Ordering query results and using LIMIT for pagination
2	**Database Architecture & Joins**	
	Schema Design & Constraints	Defining Primary Keys, Foreign Keys, Unique, and Not Null constraints Establishing relationships (One-to-Many, Many-to-Many)
		Combining tables via INNER, LEFT, RIGHT, and CROSS JOINs
	Aggregation & Grouping	Writing nested subqueries and derived tables Summarizing data with COUNT, SUM, AVG, MIN, and MAX
		Restricting aggregated results using GROUP BY and HAVING
3	**Performance & Optimization**	
	Indexing & Optimization	Creating B-Tree indexes to accelerate search performance
		Profiling slow queries with EXPLAIN statements
	Transactions & Storage Engines	Comparing InnoDB vs. MyISAM storage engines Managing transactions (START TRANSACTION, COMMIT, ROLLBACK)
	Views & Stored Procedures	Creating Views for data security and simplification
		Writing Stored Procedures and basic triggers

Apache Kafka (8)

1 Kafka Core Concepts and Architecture

Introduction to Event Streaming	Traditional messaging vs. Event Streaming Paradigms Core use cases(Log aggregation, strea processing, metrics)
Core Architectural Components	Understaing Brokers, Clusters and ZooKeeper Topics, Partitions and Segment files
Producers and Consumers	How producers publish data and handle partitioning strategies Consumer groups, oofsets and message delivery semantics

2 Stream Processing and Ecosystems

Kafka Stream API	Building real-time stream processing applications Stateless vs. Stateful transformations (Joins, Aggregations, Windowing)
KSQL/ksqlDB	Querying real-time event streams using familiar SQL syntax Creating materialized views and streaming pipelines
Performance Tuning & Security	Optimizing throughput, latency, and compression (Snappy, lz4) Securing clusters with SSL/TLS encryption, SASL authentication, and ACLs

Redis Cache (4)

1 Spring Boot Redis Integration

Spring Data Redis Basics	Connecting Spring Boot to Redis using Lettuce or Jedis clients Using RedisTemplate to perform simple CRUD operations Configuring basic text and JSON serializers for Java objects

RedPanda Connect (8)

1 Redpanda Connect Core Architecture

Pipeline Fundamentals	Declarative pipeline design processing architecture Deploying pipelines via CLI and Docker container runtimes
Data Ingestion & Egress	Configuring Inputs and Outputs from 300+ pre-built connectors Managing delivery gurantees using the in-process transaction model

2 Data transformation & Optimization

Stream Transformation	Processing payloads using Bloblang mapping and mutation language Setting up data enrichment, structures windowing, and data filtering

	Real-Time Integrations	Handling high-performance Change Data Capture (CDC) for major databases Schema validation and management using the integrated Schema Registry
3	**Operations and Observability**	
	Monitoring & Reliability	Exposing health check endpoints (/ping, /ready) for platform orchestration Integrating native OpenTelemetry tracking and Prometheus metrics pipelines
	Scaling & Security	Horizontal pipeline scaling patterns with lightweight compute footprints Securing sensitive configurations using access controls and secrets management

Angular (12)

1	**Angular Basics and MVC Architecture**	
	Introduction to Angular and Its Ecosystem:	Overview of Angular's features and benefits Key concepts: Components, Directives, and Services Setting up an Angular project with the Angular CLI
	Understanding MVC in Angular:	The role of Model, View, and Controller in Angular applications Structuring components and services to align with MVC principles Benefits of using MVC for clean and scalable applications
	Designing a Code Verification Application:	Gathering requirements: Expected inputs and verification criteria Creating a user-friendly interface for code input and results display Developing business logic for validation and error handling
2	**Building and Testing**	
	Implementing the Angular Application:	Working with Angular Forms for user input and validation Building reusable components for modularity and flexibility Integrating services for business logic and backend communication
	Testing and Debugging:	Writing unit tests for components and services Debugging and optimizing the application for better performance Ensuring error handling and validation are robust and user-friendly
	Hands-On Project:	Developing a complete Angular application to verify generated codes

		Incorporating real-time feedback and validation messages Testing the application in different scenarios to ensure reliability
		Golang (8)
1	Basics	
		Packages, variables and functions
		Flow Control Statements, for, if else, switch and defer
		More type structs, slices and maps
2	Methods and Interfaces	
		Methods and Interfaces
3	Concurrency	
		Concurrency
		Implementation (4)
1	**Real time Project works**	
2	**Final Assessment Test**	