Comprehensive Java Developer to DevOps Engineer Syllabus (114 Hours)
Training Mode: Instructor-led (3 hours/day)
Duration: 38 Days (141 Hours Total)
Audience: Java Developers aiming to master full-stack, DevOps, and microservices with Generative AI integration

Module 1: Object-Oriented Programming Concepts (Day 1,2)

- Introduction to OOPs
- Class and Object creation
- Constructors (Default, Parameterized, Constructor Overloading)
- Inheritance: Single, Multilevel, Hierarchical
- Method Overloading and Overriding
- Polymorphism: Compile-time vs Runtime
- Abstraction (Abstract classes and interfaces)
- Encapsulation and access modifiers
- Practical:
    - o Student class with constructor and methods
    - o Vehicle and Car inheritance example
    - o Shape polymorphism demo

Module 2: Exception Handling (Day 3)

- Exception hierarchy in Java
- Checked vs Unchecked Exceptions
- Try-Catch-Finally block flow
- Nested try blocks
- Throw and Throws keyword usage
- Creating custom exceptions
- Practical:
    - o Division and file handling with try-catch
    - o Custom exception for age validation

Module 3: Multithreading in Java (Day 4)

- Thread class and Runnable interface
- Thread lifecycle
- Creating and starting threads
- Thread priorities and sleep()
- Synchronization and deadlock
- wait(), notify(), notifyAll()
- Thread-safe collections
- Practical:
    - o Print even/odd using two threads

- o Bank withdrawal with synchronization
- o Producer-Consumer with inter-thread communication

Module 4: Java Collections Framework (Day 5, 6)

- • Collection Interface Overview
- • List: ArrayList, LinkedList
- • Set: HashSet, LinkedHashSet, TreeSet
- • Map: HashMap, TreeMap, LinkedHashMap
- • Queue: PriorityQueue, Deque
- • Stack class and legacy collections
- • Generics and Type safety
- • Comparable vs Comparator
- • Practical:
  - o Student sort by name and roll
  - o Word frequency counter using HashMap
  - o Priority task scheduler using PriorityQueue

Module 5: JDBC Database Integration (Day 7)

- • Introduction to JDBC and its role in Java applications
- • Core components of JDBC: DriverManager, Connection, Statement/PreparedStatement, ResultSet
- • JDBC Driver Types: Type 1-4
- • Adding MySQL JDBC Driver
- • Steps to connect (Class.forName, DriverManager, SQL, execution, closing)
- • Statement vs PreparedStatement
- • ResultSet navigation
- • Layered Architecture: JavaBean, DAO, Service, Controller, Main Class, Utility
- • CRUD Operations: Add, View, Update, Delete employees

Module 6: Java 8 New Features (Day 8, 9)

- Introduction to Java 8
- Lambda Expressions
- Functional Interfaces
- Method References
- Default and Static methods in Interfaces
- Streams API (Creation, Intermediate, Terminal operations)
- Optional Class and Avoiding NullPointerException
- Date and Time API (java.time package)
- Practical:
  - o Lambda with sorting custom object list

- Filtering employees using Stream API
- Using Optional to avoid null checks

Module 7: Java 11 Features Overview (Day 10)

- Introduction to Java 11 and LTS concept
- Running Java programs without compiling (Single File Source Code)
- var keyword for local variables (inferred typing)
- String enhancements (lines(), strip(), isBlank(), repeat())
- Files API Enhancements
- Optional Enhancements
- Practical:
  - Using var in loops and lambda
  - File reading and filtering lines with Files.lines()

Module 8 : Common Java Design Patterns (Day 11, 12)

- Introduction to Design Patterns
  - Why Design Patterns?
  - Types: Creational, Structural, Behavioral
  - Real-world need and impact in software architecture
- Creational Patterns (Focus: Object creation)
  - Singleton Pattern
    - Ensures one instance only (used in Logging, Config)
  - Factory Pattern
    - Used to create objects without specifying exact class
  - Builder Pattern
    - For creating complex objects with many fields
- Structural Patterns (Focus: Object structure)
  - Adapter Pattern
    - Allows incompatible interfaces to work together
  - Decorator Pattern
    - Add responsibilities to objects dynamically
  - Facade Pattern
    - Provides a simplified interface to a complex system
- Behavioral Patterns (Focus: Object interaction)
  - Observer Pattern
    - Publisher-subscriber model (UI updates, messaging)
  - Strategy Pattern
    - Select behavior/algorithm at runtime
  - Template Method Pattern

        - Define steps of an algorithm; allow subclasses to change some steps
    - Practical Examples
        - Singleton for Config Manager
        - Factory for Notification Service (Email/SMS)
        - Strategy for Payment Methods
        - Observer for Stock Ticker/Price Updates
        - Decorator for Pizza Toppings

Module 9: Introduction to Reactive Programming (Day 13)

- What is Reactive Programming
- Overview of Observables, Observers, and Subscriptions
- Why RxJS in modern applications (Angular, React, Node)
- Creating Observables using of, from, interval, timer
- Subscribing to Observables and teardown logic
- Practical:
    - Create observable from array and interval
    - Unsubscribe from timer observable

Module 10 : RxJS Operators - Creation & Transformation (Day 14)

- Creation Operators (of, from, interval, timer)
- Transformation Operators: map, pluck, filter, scan
- Pipeable operators and chaining
- Practical:
    - Filtering odd numbers from stream
    - Mapping API response using map and pluck

Module 11: Maven Build Tool (Day 15)

- What is Maven?
- Role in project lifecycle
- Advantages over Ant
- POM file, dependencies, repositories
- Lifecycle phases
- Plugin configuration
- Maven commands (install, package, dependency tree)

Module 12: Introduction to JEE / Jakarta EE (Day 16)

- JEE architecture: multi-tiered
- Key technologies: JSP, Servlets
- Messaging: JMS
- Transaction management: JTA
- Web vs Application Servers (Tomcat, WildFly)

Module 13 : Web Fundamentals (Day 17)

- HTML, Forms, Tables
- CSS Basics
- JavaScript DOM, Events
- Responsive Design basics
- Practical: Registration form with validation, Feedback page with CSS Grid

Module 14 : Servlet, JSP & MVC (Day 18)

- Servlet API, Deployment descriptor
- ServletConfig vs ServletContext
- RequestDispatcher, JSP tags, EL
- JavaBean in JSP
- MVC pattern using Servlet and JSP
- Practical: Login, Feedback form storing via MVC

Module 15: Spring Core & Context (Day 19-20)

- Spring Framework intro
- IoC, Bean lifecycle, DI (Constructor, Setter)
- Autowiring, Java/XML config
- Component Scanning, Stereotypes
- Practical: Annotation-based injection, Repository with @Component/@Autowired
- AI: Generate configuration for beans

Module 16-17: Spring DAO, ORM, JPA (Day 21-22)

- DAO best practices
- Spring JdbcTemplate
- ORM with Hibernate/JPA
- JPA CRUD, Entity Mapping
- Spring Data JPA
- Query Methods, Pagination
- Practical: CRUD using JpaRepository, Entity relationships

Module 18: Spring MVC (Day 23)

- Spring MVC Overview and Architecture
- DispatcherServlet, Request flow
- URL mapping, ModelAndView
- Entity creation and layer connection
- Dependency Injection in Controller/Service

Module 19: Spring Boot & Thymeleaf (Day 24)

- Spring Boot Starters, Initializr
- Devtools, Profiles
- Thymeleaf templates
- Form handling, validation
- Practical: CRUD UI with Thymeleaf

Module 20: Spring RESTful Web Services (Day 25)

- @RestController vs @Controller
- CRUD APIs, JSON handling
- Global Exception Handling
- Swagger Integration
- Practical: Book API with Swagger Docs
- AI: Use Copilot to scaffold REST controller

Module 21: Microservices Architecture (Day 26)

- Monolith vs Microservices
- DDD basics
- Microservice setup with Spring Boot
- API creation, config

Module 22: REST Communication & Spring Cloud (Day 27-28)

- RestTemplate & WebClient
- Spring Cloud: Eureka, Gateway, Config Server
- Load Balancing

Module 23: Eureka & Config Server (Day 29)

- Registering services with Eureka
- Centralized configs using Git/File System
- Config refresh with @RefreshScope

Module 24: API Gateway (Day 30)

- Routing, filtering with Spring Cloud Gateway
- Path rewriting, load balancing

Module 25: Spring Security (Day 31)

- Authentication/Authorization
- Filter chain
- RBAC
- Custom login form, Password encoding
- Custom UserDetailsService

Module 26-27: Docker & Compose (Day 32-33)

- Docker basics, CLI, Dockerfile
- Build/run containers
- Docker Compose setup
- Practical: Dockerize all services

Module 28: CI/CD Tools (Day 35)

- Jenkins pipelines
- GitHub Actions workflows
- Maven build, Docker deploy
- Practical: Auto pipeline for car booking project

Module 29-30: Final Project - Car Booking System (Day 35,36 and 37)

- User, Car, Booking Services
- API Gateway, Security
- Thymeleaf UI, Docker Compose
- CI/CD integration
- AI Support: Project scaffolding, testing

BONUS: Generative AI for Java Developers (Ongoing)

- GitHub Copilot: Scaffolding, refactoring
- ChatGPT: Debugging, architecture suggestions
- Uizard: Wireframes
- TestRigor, QA Wolf: AI Test Generation

Post-Course: Advanced Generative AI Tools

- ChatGPT Plugins, GitHub Copilot X
- AI for Testing: TestRigor, QA Wolf
- Uizard & Figma AI
- Mini Projects:
    - o AI chatbot for documentation
    - o DTO/Entity generator
    - o CI pipeline alert responder

Total Hours: 114 Hours

Outcome: Full-stack Java Developer with Microservices, DevOps & AI skills