**Backend Java Developer Roadmap: Week-by-Week Curriculum Breakdown**

Below is a structured **week-by-week study plan** that takes you from absolute beginner to advanced backend Java developer in one year. This curriculum is designed for a full-time schedule (about 8-12 hours/week), but you can adjust pace or combine weeks if learning part-time.

**Weeks 1-2: Getting Started with Java**

* Install JDK, IDE (IntelliJ IDEA/Eclipse), Git
* Basic Java syntax, variables, data types
* Operators, conditionals (if/switch), loops
* Arrays, strings, methods, type casting
* Build your first CLI project (calculator, todo-list)
* Version control basics (Git init, commit, push, clone)

**Weeks 3-4: Object-Oriented Programming (OOP) in Java**

* Classes, objects, constructors, attributes, methods
* Access specifiers, static keyword, nested classes
* Inheritance, polymorphism, encapsulation, abstraction
* Interfaces, abstract classes, enums, method chaining
* Pass by value/reference, object lifecycle
* Packages and modules

**Weeks 5-6: Advanced Java Concepts**

* Collections (List, Set, Map, Queue, Stack, Array vs ArrayList)
* Generics, iterators, Optionals
* Lambda expressions, functional interfaces, Stream API
* Exception handling (try-catch, custom exceptions)
* Annotations, working with dates/times
* File I/O, basic Networking, Regex

**Week 7: Build Tools, IDE, Version Control**

* Maven fundamentals: project structure, dependencies
* Gradle basic usage (contrast with Maven)
* Jenkins CI/CD basics
* Git branching, merge, pull requests
* Repo hosting (GitHub/Bitbucket/GitLab)

**Weeks 8-9: SQL & Database Fundamentals**

* Introduction to relational databases (PostgreSQL/MySQL)
* SQL syntax: SELECT, INSERT, UPDATE, DELETE
* Data filtering (WHERE), ORDER BY, GROUP BY, HAVING
* JOINs (INNER, LEFT, RIGHT, OUTER, CROSS), aggregate functions
* Data constraints (PK, FK, unique, not-null, check)
* Basic transactions & ACID
* NoSQL overview (MongoDB, Redis)

**Weeks 10-11: Database Access in Java**

* JDBC fundamentals, connecting Java to SQL
* Writing CRUD operations in Java/JDBC
* Introduction to ORM: JPA and Hibernate
* Mapping entities, relationships, entity lifecycle
* Transaction management, performance profiling (N+1)

**Weeks 12-14: Spring Core & Spring Boot Fundamentals**

* Spring architecture overview, terminology
* Configuration: XML, Java, annotations
* Bean lifecycle, dependency injection, IoC, Spring contexts
* Spring Boot: starters, auto-configuration, embedded servers
* Application properties (YAML/Properties), logging, profiles
* Create your first Spring Boot CRUD REST API

**Weeks 15-17: Web Development with Spring MVC**

* MVC fundamentals, controllers, request mappings
* Request/response handling, JSON/XML serialization
* Exception handling in Spring
* API documentation (Swagger/OpenAPI)
* Static resources, JSP/servlet introduction

**Weeks 18-19: ORM and Advanced Data Persistence**

* Spring Data JPA: repositories, pagination, sorting
* Advanced Hibernate: cascading, fetch types, lazy/eager loading
* Multi-datasource config, database migrations (Flyway/Liquibase)
* Caching and transactions in Spring

**Weeks 20-21: Application Security & Authentication**

* Spring Security overview, user authentication (form, basic, JWT, OAuth2)
* Password hashing (bcrypt, scrypt), CSRF, HTTPS/SSL/TLS
* CORS, server security basics
* API security best practices (roles/permissions, securing endpoints)
* Review OWASP top 10 risks

**Weeks 22-23: Unit & Integration Testing**

* JUnit 5: test structure, assertions, advanced features
* Mockito: mocking, stubbing, spies
* @SpringBootTest, integration tests, TestContainers
* MockMVC for controller testing
* REST Assured for API testing
* Behavior/load testing (Cucumber, JMeter)
* CI integration for testing

**Weeks 24-26: Microservices and Spring Cloud**

* Microservices vs monolithic architectures
* Domain-driven design, twelve-factor app principles
* Service discovery (Eureka/Consul)
* Spring Cloud Gateway/API gateway concepts
* Centralized config server (Spring Cloud Config)
* Circuit breaker pattern (Hystrix/Resilience4j)
* Messaging (RabbitMQ, Kafka intro)
* Distributed tracing (Sleuth, Zipkin)
* Build a basic microservice system with API gateway & two services

**Weeks 27-29: System Design & Architecture Patterns**

* High availability, horizontal/vertical scaling, load balancing
* Sharding, replication, CAP theorem
* Caching strategies (Redis, Memcached)
* Key system design patterns (Singleton, Factory, CQRS, Event Sourcing, Repository)
* Monolith vs microservices vs SOA; layered, DDD, clean architecture
* Event-driven messaging
* Scale simulation project: add caching, scale-up API, measure perf

**Weeks 30-31: DevOps and Production Ready Systems**

* Docker: containers, images, Compose basics
* Kubernetes fundamentals (pods, services, deployments)
* CI/CD pipelines: Jenkins, GitHub Actions basics
* Automated deployment and config
* App monitoring: Prometheus, Grafana, Spring Boot Actuator
* Logging, alerting, health checks

**Weeks 32-35: Advanced Topics & Specialization**

* Concurrency in Java: threads, virtual threads, synchronization
* JVM tuning, garbage collection, memory profiling
* Real-time systems: WebSockets, SSE, long/short polling
* NoSQL focus: MongoDB advanced, Neo4j, Elasticsearch/Solr
* Time-series databases (InfluxDB, Cassandra basics)
* Search APIs and indexing

**Weeks 36-38: Soft Skills and Coding Practice**

* DSA for coding interviews (arrays, strings, linked lists, sorting/searching, trees, graphs, recursion, DP, OOP)
* Problem solving daily (LeetCode/HackerRank/Edabit)
* System design interview practice
* Effective communication and documentation (GitHub READMEs, Swagger docs)
* Building and deploying a complete microservice system (e-commerce/blog/chat app)
* Portfolio polishing, interviews, applications, networking

**Weeks 39-52: Portfolio Projects & Review**

* Complete 3 showcase backend projects:
  + REST API (CRUD, users/auth, security, tests)
  + Distributed microservice system with messaging, tracing, auth, caching, monitoring
  + Real-time app (chat, notification service, or search API)
* Document code, write detailed READMEs
* Review all topics, focus on weak areas
* Final prep: mock interviews, system design challenges, refine portfolio

**Helpful Notes**

* **Each week**, review previous topics, practice coding assignments, and progress towards a project milestone.
* Adjust the schedule if you need more/less time for specific topics.
* Use official documentation, top books, online courses, and coding platforms.

If you want a printable checklist, topic-wise resource links, or a project guideline for each phase, let me know!