Java Developer Nanodegree Program Syllabus

Learn the de-facto language for building enterprise-scale applications

Program Overview

The ultimate goal of the Java Developer Nanodegree program is to equip students with the unique skills they need to build enterprise-scale applications with Java. A graduate of this program will be able to:

- Understand the fundamentals of Java, while being introduced to a Spring Boot framework and associated integrations and plugins.
- Describe the differences between web services, APIs, and microservices, develop REST and GraphQL APIs, and learn how to secure, consume, document, and test those APIs and web services.
- Work with relational and non-relational databases, use Java to read/write and build test cases for MySQL and MongoDB, and build persistence for Java applications.
- Learn about Git, version control, and best practices for authorization and authentication. Use Jenkins to build CI/CD pipeline to deploy code to production.

This program is comprised of 4 courses and 4 projects. Each project you build will be an opportunity to demonstrate what you've learned in the lesson, and will demonstrate to potential employers that you have skills in these areas.

Estimated Length of Program: 4 months
Frequency of Classes: Self-paced
Prerequisite Knowledge: Intermediate knowledge of any programming language, preferably an object-oriented language (e.g, Python, C++). Some web development experience desirable but not required.

Project 1: Create a Chat Room Application with Spring Boot

In this project, students will use the skills acquired in the first course to build a web-based chat room application using spring boot and websocket components. Students will implement the message model and controller to build an in-memory application. Students will also use Thymeleaf to create web pages to the build user login flow and chat room interface. Once the application is ready, students will create web driver tests to make sure the web application has sufficient test coverage.

Supporting Course Content: Java Basics

Lesson	Learning Outcomes
Introduction to Java	➔ Use different data structures and algorithms in Java and calculate time complexity of the code ➔ Implement exception handling, file IO, BufferReader
Spring Boot Servlet, Filter and Listener	➔ Describe how the model-view-controller (MVC) architecture works ➔ Configure Spring Boot and create Spring Boot servlet, filter and listener, from scratch ➔ Interact with static resources in a Spring Boot application
Spring Boot with Thymeleaf and Mybatis	➔ Manage spring boot application with Maven plugins ➔ Create web pages with Thymeleaf ➔ Integrate Spring Boot with Mybatis
Spring Boot Web Socket with spring boot tests	➔ Implement Spring Boot websocket structure and components. ➔ Create the message model and implement chat room controller. ➔ Create web driver tests to ensure test coverage.

Project 2: Build the Backend System for a Car Website

In this course, the student will build a backend system for a web site of cars. This backend will be composed of vehicles list services, pricing services, and location services as mentioned below:

- Vehicles API - a REST API to maintain vehicles data (CRUD)
- Pricing Service - a REST API to retrieve the price of a vehicle
- Location API - a HTTP client to retrieve the location of the vehicle

In the project students will use Java APIs and frameworks to integrate different services using different communication styles. Students will write the CRUD operations to store and retrieve vehicle data and implement an HTTP client to retrieve the address of the vehicle given the latitude and longitude. Students will also integrate the clients (Vehicle API) with pricing services to retrieve the price. Lastly, students will learn to use Swagger to efficiently create documentation for their APIs.

During the development of these steps, the student will be guided to write unit tests, error handling, logging, and other best practices.

Supporting Course Content: Web Services and APIs

Lesson	Learning Outcomes
Web Services & APIs Overview	➔ Describe web services and their advantages ➔ Describe how web services communicate ➔ Explore the differences between web services, APIs, and microservices
Develop REST APIs with Spring Boot	➔ Describe the REST architectural style and the importance of data formats ➔ Develop a REST API using Spring Boot and incorporate exception handling. ➔ Use proper HTTP response codes
Develop GraphQL APIs with Spring Boot	➔ Describe GraphQL and its advantages over REST ➔ Create a GraphQL schema ➔ Develop a GraphQL server and API using Spring Boot ➔ Use GraphQL to execute queries and operations on data
Develop Microservices with Spring Boot	➔ Describe the Microservices Architecture (MSA) ➔ Expose a microservice using Spring Boot ➔ Register a microservice
Secure API Endpoints with Spring Security	➔ Describe Spring Security ➔ Explain the differences between authentication vs authorization ➔ Incorporate Basic Authentication practices to secure an API
Consume Web Services and APIs	➔ Consume a REST API ➔ Consume a SOAP-based web service ➔ Fetch and process XML and JSON
Document REST APIs	➔ Describe Swagger, an open-source software framework to design, build, document, and consume RESTful web services ➔ Add Swagger annotations to model ➔ Generate API documentation
Test REST APIs	➔ Describe and explain unit and integration testing ➔ Incorporate unit and integration testing into a REST API

Project 3: Data Store for Customer Reviews

Students will build the polyglot persistence layer for a REST API that will support the customer reviews section of a product page in an ecommerce application. Students will begin with setting up MySQL & MongoDB and define classes for MongoDB model. Students will then update the persistence service to read/write from both MySQL and MongoDB, and write code to calculate aggregate rating for a product. Students will also build corresponding test cases for JPA & MongoDB repositories and wire the persistence service to the already provided REST Controller. By the end of this project, students will have built a fully functioning REST API with persistence in Relational Database Management System (RDBMS) that can be inspected via a tool like Postman.

Supporting Course Content: Data Stores & Persistence

Lesson	Learning Outcomes
RDBMS & JDBC	➔ Describe the structure of relational databases (MySQL) and explain ACID compliance. ➔ Write SQL queries to operate on relational databases. ➔ Use Java Database Connectivity (JDBC) API to build persistence for a Java application. ➔ Use Flyway to perform database migrations.
Java Persistence API	➔ Describe the benefits and uses for Object Relational Mapping and Java Persistence API (JPA). ➔ Explain the concepts behind Spring Data project. ➔ Develop a Spring Boot app that defines a Repository. ➔ Use Spring and H2 for testing database code.
NoSQL & MongoDB	➔ Describe NoSQL, non relational databases and flexible schema. ➔ Describe the use cases for the document oriented database MongoDB. ➔ Use Mongo shell to perform different operations such as create/modify collections, select/create/update/delete documents and aggregations. ➔ Describe the pros & cons of nested documents and document references.
MongoDB for Java	➔ Connect to MongoDB from Java. ➔ Use Spring Data MongoDB to connect to MongoDB. ➔ Define document relationships. ➔ Define Repository and Repository methods that use filters, sort and pagination. ➔ Use Embedded MongoDB for testing Repositories.

Project 4: Implement Authorization for an eCommerce Application

In this project, students will add authorization using Spring Security with OAuth and username/password combinations to an eCommerce web application created in Spring Boot. Proper security and hashing will need to be implemented to store this data as well. Students will identify the right metrics for an effective analytics environment and use either Splunk or ELK to analyze the metrics. Students will also automate the configuration and deployment of these systems and the application. Students will use Jenkins to integrate with their version control and deploy their application to AWS.

Supporting Course Content: Security and DevOps

Lesson	Learning Outcomes
Git	➔ Learn the basics of git such as branching, pull requests, and merging ➔ Describe what version control is and means
Authorization and Authentication	➔ Identify the need for security in modern day web applications ➔ Describe best practices for authorization and authentication ➔ Implement modern authorization and authentication technologies such as password hashing and JWT
Testing	➔ Learn and use testing frameworks such as junit ➔ Describe the concept of code coverage and its importance ➔ Implement negative testing as well as happy path testing
Logging and Analytics	➔ Identify important application metrics and log them ➔ Send logs to Splunk ➔ Create visualizations and dashboards in Splunk to display those metrics
Jenkins and CI/CD	➔ Describe and explain CI/CD ➔ Create a build pipeline using Jenkins ➔ Build a Docker Image ➔ Create a CI pipeline for a Docker Image ➔ Deploy Docker container in production