Index

JEE Full Stack with Angular (60 Days)

JEE Full Stack with Angular variant provides exposure to the entire spectrum of Java technologies starting from Core Java to Spring. It focuses on Web Application development using Angular and Spring Technology. The following table lists the course structure.

Sr. No.	Course	Duration	Remarks
1	Programming Foundation with Java	6	
2	Problem Solving and Data Structure with Java	6.5	HackerRank Mock assessment on 4th day
3	Sprint 1 Evaluation (Automated) (Problem Solving)	0.25	HackerRank test on problem solving
4	Advance Concepts in Java	4.5	
5	Sprint 2 Evaluation (Automated) (Java Basics)	0.25	HackerRank test on java basics
6	Database and SQL	2	
7	JDBC + JPA with Hibernate	2.5	Sprint Case Study will be given
8	Maven Lifecycle	0.5	
9	Spring 5.0	6	
10	Spring Microservices	1.5	

11	Web Basics	3	
12	Angular	5	
13	Angular Integration with Spring Boot Application	1	
14	Micro FrontEnd	3	
15	Sprint 3 Back End Implementation of Case Study	5	
16	Sprint 4 Front End Implementation of Case Study	5	
17	Sprint 4 Back End and Front End Evaluation	1	
18	Power Skill Sessions (to be distributed across the training duration)	5	
19	L1 Preparation + Test	2	
		60	

Programming Foundation with Java

Duration : 6 Days

Contents:

- **Datatypes, Variables and Arrays**
  - Literals, Assignments, and Variables
  - Literal Values for All Primitive Types
  - Using a Variable or Array Element That Is Uninitialized and Unassigned
  - Local (Stack, Automatic) Primitives and Objects
  - Passing Variables into Methods
  - Does Java Use Pass-By-Value Semantics?
  - Passing Primitive Variables
  - Array Declaration, Construction, and Initialization
  - Declaring an Array
  - Constructing an Array
  - Initializing an Array
  - Initialization Blocks

- **Operators**
  - Java Operators
  - Assignment Operators
  - Relational Operators
  - Arithmetic Operators
  - Conditional Operator
  - Logical Operators
  - Ternary Operator

- **Flow Control, Exceptions**
  - if and switch Statements
  - if-else Branching
  - switch Statements
  - Loops and Iterators
  - Using while Loops

- Using do Loops
- Using for Loops
- Using break and continue
- The String Class
- Important Methods in the String Class
- Identifiers
- Sun's Java Code Conventions
- JavaBeans Standards
- Declare Classes
- Source File Declaration Rules
- Modularity
- Legal Return Types
- Return Type Declarations
- Returning a Value
- Class Declarations and Modifiers
- Declare Class Members
- Constructor Declarations
- Variable Declarations
- Declaring Enums
- Constructors and Instantiation
- Default Constructor
- Overloaded Constructors
- Coupling and Cohesion
- Passing Objects/Returning Objects to/from Methods

- **I/O, Formatting, and Parsing**
  - StringBuilder, and StringBuffer
  - The StringBuffer and StringBuilder Classes
  - Important Methods in the StringBuffer and StringBuilder Classes
  - Dates, Numbers, and Currency
  - Working with Dates, Numbers, and Currencies
  - Parsing, Tokenizing, and Formatting
  - Locating Data via Pattern Matching
  - Abstraction
  - Encapsulation
  - Inheritance, Is-A, Has-A
  - Polymorphism
  - Overridden Methods
  - Overloaded Methods
  - Reference Variable Casting
  - Implementing an Interface
  - Declaring an Interface
  - Declaring Interface Constants
  - Static Variables and Methods
  - Access Modifiers
  - Using Wrapper Classes and Boxing
  - An Overview of the Wrapper Classes
  - Creating Wrapper Objects
  - Using Wrapper Conversion Utilities
  - Autoboxing

- Overloading
- Garbage Collection
- Overriding hashCode() and equals()
- Overriding equals()
- Overriding hashCode()
- Collections
- What Do You Do with a Collection?
- List Interface
- Set Interface
- Map Interface
- Queue Interface
- Using the Collections Framework
- ArrayList Basics
- Autoboxing with Collections
- Sorting Collections and Arrays
- Navigating (Searching) TreeSets and TreeMaps
- Other Navigation Methods
- Backed Collections
- Generic Types
- Generics and Legacy Code
- Mixing Generic and Non-generic Collections
- Polymorphism and Generics
- Layered Architecture
    - Business Delegate Pattern
    - Data Access Object Pattern
    - Transfer Object Pattern
    - Iterator Pattern

Problem Solving and Data Structure with Java

Duration : 6.5 Days

Contents

Arrays and Big-O Notation

Sort Algorithms

Lists

Stacks

Queues

Hashtables

Search Algorithms

Trees

Heaps

HackerRank Preparation : 2 Days

Advance Concepts in Java

Duration :4.5 Days

- **Lambda Expressions**
  - Introduction
  - Writing Lambda Expressions
  - Functional Interfaces
  - Types of Functional Interfaces
  - Method reference
- **Stream API**
  - Introduction
  - Stream API with Collections
  - Stream Operations

- **Threads**
  - Defining, Instantiating, and Starting Threads
  - Defining a Thread
  - Instantiating a Thread
  - Starting a Thread
  - Thread States and Transitions
  - Thread States
  - Preventing Thread Execution
  - Sleeping
  - Thread Priorities and yield( )
  - Synchronizing Code
  - Synchronization and Locks
  - Thread Deadlock
  - Thread Interaction
  - Using notifyAll( ) When Many Threads May Be Waiting

- **File Handling and Exception Handling**

  - File Navigation and I/O
  - Types of Streams
  - The Byte-stream I/O hierarchy
  - Character Stream Hierarchy
  - RandomAccessFile class
  - The java.io.Console Class
  - Serialization
  - Handling Exceptions
  - Catching an Exception Using try and catch
  - Using finally
  - Propagating Uncaught Exceptions
  - Defining Exceptions
  - Exception Hierarchy
  - Handling an Entire Class Hierarchy of Exceptions
  - Exception Matching
  - Exception Declaration and the Public Interface
  - Rethrowing the Same Exception

    - Common Exceptions and Errors

- **TDD with Junit 5**
    - Types of Tests
    - Why Unit Tests Are Important
    - What's JUnit?
    - JUnit 5 Architecture
    - IDEs and Build Tool Support
    - Lifecycle Methods
    - Test Hierarchies
    - Assertions
    - Disabling Tests
    - Assumptions
    - Test Interfaces and Default Methods
    - Repeating Tests
    - Dynamic Tests
    - Parameterized Tests
    - Argument Sources
    - Argument Conversion
    - What Is TDD?
    - History of TDD
    - Why Practice TDD?
    - Types of Testing
    - Testing Frameworks and Tools
    - Testing Concepts
    - Insights from Testing
    - Mocking Concepts
    - Mockito Overview
    - Mockito Demo
    - Creating Mock Instances
    - Stubbing Method Calls

DevOps Concepts

Contents

- DevOps/ CI CD concepts (GitHub, CI Jenkins, Sonar)

Contents:

- Introduction to DevOps :
    - What is DevOps
    - Evolution of DevOps
    - Agile Methodology
    - Why DevOps
    - Agile vs DevOps
    - DevOps Principles
    - DevOps Lifecycle
    - DevOps Tools
    - Benefits of DevOps

- Continuous Integration and Delivery pipeline
- Use-case walkthrough

- GitHub
  - What is DevOps
  - Introduction to Git
  - Version control
  - Repositories and Branches
  - Working Locally with GIT
  - Working Remotely with GIT

- Jenkins
  - Introduction to CI
  - Jenkins Introduction
  - Creating Job in Jenkins
  - Adding plugin in Jenkins
  - Creating Job with Maven & Git

- Jenkins With TDD(Junit testing)
  - Integration of jUnit testing with Jenkins

- Sonar

Database and SQL

Duration : 2 Days

Contents:

- **Introduction**
  - The Relational Model
  - What is PostgreSQL?
- **Understanding Basic** PostgreSQL **Syntax**
  - The Relational Model
  - Basic SQL Commands - SELECT
  - Basic SQL Commands - INSERT
  - Basic SQL Commands - UPDATE
  - Basic SQL Commands – DELETE

- **Querying Data with the SELECT Statement**
  - The SELECT List
  - SELECT List Wildcard (*)
  - The FROM Clause
  - How to Constrain the Result Set
  - DISTINCT and NOT DISTINCT

- **Filtering Results with the Where Clause**
  - WHERE Clause
  - Boolean Operators
  - The AND Keyword

- The OR Keyword
- Other Boolean Operators BETWEEN, LIKE, IN, IS, IS NOT

- **Shaping Results with ORDER BY and GROUP BY**
  - ORDER BY
  - Set Functions
  - Set Function And Qualifiers
  - GROUP BY
  - HAVING clause

- **Matching Different Data Tables with JOINs**
  - CROSS JOIN
  - INNER JOIN
  - OUTER JOINs
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
  - SELF JOIN

- **Creating Database Tables**
  - CREATE DATABASE
  - CREATE TABLE
  - NULL Values
  - PRIMARY KEY
  - CONSTRAINT
  - ALTER TABLE
  - DROP TABLE

JDBC + JPA with Hibernate

Duration : 2.5 Days

Contents :

- **Introduction to JDBC**
  - Database Drivers
  - Establishing Connection
  - Statement, Prepared Statement
  - ResultSet
  - SQLException
- **Introduction to JPA**
  - Introduction & overview of data persistence
  - Overview of ORM tools
  - Understanding JPA
  - JPA Specifications
- **Entities**
  - Requirements for Entity Classes
  - Persistent Fields and Properties in Entity Classes
  - Persistent Fields
  - Persistent Properties
  - Using Collections in Entity Fields and Properties
  - Validating Persistent Fields and Properties

- Primary Keys in Entities

- **Managing Entities**
  - The EntityManager Interface
  - Container-Managed Entity Managers
  - Application-Managed Entity Managers
  - Finding Entities Using the EntityManager
  - Managing an Entity Instance's Lifecycle
  - Persisting Entity Instances
  - Removing Entity Instances
  - Synchronizing Entity Data to the Database
  - Persistence Units
- **Querying Entities**
  - Java Persistence query language (JPQL)
  - Criteria API
- **Entity Relationships**
  - Direction in Entity Relationships
  - Bidirectional Relationships
  - Unidirectional Relationships
  - Queries and Relationship Direction
  - Cascade Operations and Relationships

Maven Fundamentals

Duration : 0.5 Days

- Introduction
- Folder Structure
- The pom.xml
- Dependencies
- Goals
- Scopes
- The Compiler Plugin
- Source Plugin
- Jar Plugin
- Maven Life Cycle

Spring 5.0

Duration : 6 Days

Contents:

- **Spring Core**

  Spring Core Introduction / Overview

  - Shortcomings of Java EE and the Need for Loose Coupling
  - Managing Beans, The Spring Container, Inversion of Control
  - The Factory Pattern
  - Configuration Metadata - XML, @Component, Auto-Detecting Beans
  - Dependencies and Dependency Injection (DI) with the BeanFactory
  - Setter Injection

    Creational Design Pattern

- Factory Pattern
- Singleton Pattern
- Prototype Pattern

Spring Container

- The Spring Managed Bean Lifecycle
- Autowiring Dependencies

Dependency Injection

- Using the Application Context
- Constructor Injection
- Factory Methods

Metadata / Configuration

- Annotation Configuration @Autowired, @Required, @Resource
- @Component, Component Scans.
- Life Cycle Annotations
- Java Configuration, @Configuration, XML free configuration
- The Annotation Config Application Context

- **Spring Boot**

Spring Boot Introduction

- Spring Boot starters, CLI
- Application class
- @SpringBootApplication
- Dependency injection, component scans, Configuration
- Externalize your configuration using application.properties

Using Spring Boot

- Build Systems, Structuring Your Code, Configuration, Spring Beans and Dependency Injection, and more.

Spring Boot Essentials

- Application Development, Configuration, Embedded Servers, Data Access, and many more
- Common application properties
- Auto-configuration classes
- Spring Boot Dependencies

- **Spring MVC (via Spring Boot)**

JSP

i. Writing Java Server Page
    1. Developing a Simple Java Server Page

ii. JSP Scripting Elements
    1. Forms of Scripting Elements
    2. Predefined Variables
    3. Examples using Scripting Elements

iii. JSP Directives
    1. Page directive
    2. Include directive
iv. JSP Actions
    1. jsp:include Action
    2. jsp:forward Action
v. JSP Standard Template Library (JSTL)
    1. What is JSTL?
    2. Installing JSTL
    3. Using the Expression Language
    4. Using JSTL Core Libraries

Introduction / Developing Web applications with Spring MVC

- Model View Controller
- Front Controller Pattern
- DispatcherServlet Configuration
- Controllers, RequestMapping
- Working with Forms
- Getting at the Request, @RequestParam, @RequestHeader
- ModelAndView
  Presentation Layer Design Pattern
    - Intercepting Filter Pattern
    - Front Controller Pattern
    - View Helper Pattern

Spring Controllers

- Using @ResponseBody
- JSON and XML data exchange

RESTful Web Services

- Core REST concepts
- REST support in Spring 5.x
- REST specific Annotations in Spring
- @PathVariable, @RequestParam
- JSON and XML data exchange
- @RequestMapping

- **Spring Data JPA**
    - Spring Data JPA Intro & Overview
    - Core Concepts, @RepositoryRestResource
    - Defining Query methods
    - Query Creation
    - Using JPA Named Queries
    - Defining Repository Interfaces
    - Creating Repository instances
    - JPA Repositories

- Persisting Entities
- Transactions

- **Spring Data REST**
  - Introduction & Overview
  - Adding Spring Data REST to a Spring Boot Project
  - Configuring Spring Data REST
  - Repository resources, Default Status Codes, Http methods
  - Spring Data REST Associations
  - Define Query methods
  - Postman/Swagger

- **Global Exception Handler**

  @Controller Advice

  @Exception Handler

Spring Security

- Spring Security
- Spring Security with Spring boot

Spring Microservices

Duration : 1.5 Days

Microservices Overview

- Microservices architecture
- Core characteristics of microservice
- Use cases and Benefits
- Design standards
- Monolithic Architecture
- Distributed Architecture
- Service oriented Architecture
- Microservice and API Ecosystem
- Microservices in nutshell
- Point of considerations
- SOA vs. Microservice
- Microservice & API

Locating Services at Runtime Using Service Discovery

- Role of service discovery in microservices
- Describing spring cloud Eureka
- Creating Eureka Server
- Registering Services with Eureka
- Configuring health information
- Actuator & Profiles

HTML, CSS , BootStrap, Javascript ES 6

Program Duration : 3 Days

HTML 5:

- HTML Basics
  - Understand the structure of an HTML page.
  - New Semantic Elements in HTML 5
  - Learn to apply physical/logical character effects.
  - Learn to manage document spacing.
- Tables
  - Understand the structure of an HTML table.
  - Learn to control table format like cell spanning, cell spacing, border
- List
  - Numbered List
  - Bulleted List
- Working with Links
  - Understand the working of hyperlinks in web pages.
  - Learn to create hyperlinks in web pages.
  - Add hyperlinks to list items and table contents.
- Image Handling
  - Understand the role of images in web pages
  - Learn to add images to web pages
  - Learn to use images as hyperlinks
- Frames
  - Understand the need for frames in web pages.
  - Learn to create and work with frames.
- HTML Forms for User Input
  - Understand the role of forms in web pages
  - Understand various HTML elements used in forms.
  - Single line text field
  - Text area
  - Check box
  - Radio buttons
  - Password fields
  - Pull-down menus
  - File selector dialog box
- New Form Elements
  - Understand the new HTML form elements such as date, number, range, email, search and datalist
  - Understand audio, video, article tags

CSS 3

- **Introduction to Cascading Style Sheets 3.0**
  - What CSS can do
  - CSS Syntax
  - Types of CSS
- **Working with Text and Fonts**
  - Text Formatting
  - Text Effects
  - Fonts
- **CSS Selectors**
  - Type Selector
  - Universal Selector
  - ID Selector
  - Class selector
- **Colors and Borders**
  - Background
  - Multiple Background
  - Colors RGB and RGBA
  - HSL and HSLA
  - Borders
  - Rounded Corners
  - Applying Shadows in border

BootStrap

- **Introduction to Bootstrap**
  - Introduction
  - Getting Started with Bootstrap
- **Bootstrap Basics**
  - Bootstrap grid system
  - Bootstrap Basic Components
- **Bootstrap Components**
  - Page Header
  - Breadcrumb
  - Button Groups
  - Dropdown
  - Nav & Navbars
- **JavaScript Essentials**
- **ES6**
  - Var, Let and Const keyword
  - Arrow functions, default arguments
  - Template Strings, String methods
  -
  - Object de-structuring
  - Spread and Rest operator
  - Typescript Fundamentals
  - Types & type assertions, Creating custom object types, function types
  - Typescript OOPS - Classes, Interfaces, Constructor, etc

Angular

Duration : 6 Days

Contents:

- **Introduction to Angular Framework**
  - Introduction to Angular Framework, History & Overview
  - Environment Setup, Angular CLI, Installing Angular CLI
  - NPM commands & package.json
  - Bootstrapping Angular App, Components, AppModule
  - Project Setup, Editor Environments
  - First Angular App & Directory Structure
  - Angular Fundamentals, Building Blocks
  - MetaData

- **Essentials of Angular**
  - Component Basics
  - Setting up the templates
  - Creating Components using CLI
  - Nesting Components
  - Data Binding - Property & Event Binding, String Interpolation, Style binding
  - Two-way data binding
  - Input Properties, Output Properties, Passing Event Data

- **Templates, Styles & Directives**
  - Template, Styles, View Encapsulation, adding bootstrap to angular app
  - Built-in Directives,
- **Pipes, Services & Dependency Injection**
  - In-built Pipes
  - Services & Dependency Injections
  - Creating Data Service

- **Template-Driven and Reactive Forms**
  - Template-Driven vs Reactive Approach
  - Understanding Form State
  - Built-in Validators & Using HTML5 Validation
  - Grouping Form Controls
  - FormGroup, FormControl, FormBuilder
  - Forms with Reactive Approach
  - Predefined Validators & Custom Validators
  - Showing validation errors

- **Components Deep Dive / Routing**
  - Component Life Cycle Hooks
  - Reusable components in angular using <ng-content>
  - Navigating with Router links
  - Understanding Navigation Paths
  - Navigating Programmatically
  - Passing Parameters to Routes
  - Passing Query Parameters and Fragments

- **Http Requests / Observables**
  - HTTP Requests
  - Sending GET Requests
  - Sending a PUT Request
  - Using the Returned Data
  - Catching Http Errors
  - Basics of Observables & Promises

- **Authentication and Route Protection**
  - How Authentication works in SPA
  - JWT Module
  - JSON Web Tokens
  - Signup, Login and logout application
  - Router Protection, Route Guards
  - CanActivate interface
  - Checking and using Authentication Status
  - Router Protection and Redirection
  - Angular Integration with Spring Boot Application

Micro FrontEnd

Duration - 16 hours

1 Introduction — 1 hr

Understand the basic principles and concepts of micro frontends.	Benefits and challenges of adopting micro frontend architecture.

2 Webpack & Design Pattern — 5 hrs

Explore Webpack 5's Module Federation, a powerful tool for building micro frontends.	Share and consume modules between different applications
Investigate various communication techniques between micro frontends	Custom Events, Pub/Sub patterns, or shared state management
Create applications that use JavaScript Module Federation to share and consume modules	

3 Containerization — 10 hrs

containerization tools like Docker to package and deploy micro frontend applications.	Set up continuous integration and continuous deployment pipelines for your micro frontend applications
Routing in Micro Frontends	Best practices while developing Micro Front End Architecture
custom routing strategies	Demo to Orchestrating multiple containers

Sprints

Would be implemented using Agile approach.

Udemy clean code course (6 hours) to be learnt on Saturday before beginning with Sprint

- **Sprint 1 implementation with code reviews of L&D and BU SME**
  Developing a Spring Boot application following Layered Architecture, Spring Boot with Maven, Spring REST, Spring Data JPA, Database and Postman/Swagger

  - Project Kick off – Expectation setting by BU Mentor
  - Sprint Planning, Group formation, Case study sharing by BU team
  - Class Design, Interface Design
  - Identifying Unit Test Cases
  - Implementing Spring JPA into project
  - Test case reviews
  - Code reviews
  - Performance monitoring during the sprint implementation and sharing the feedback
  - **Sprint - 1 Evaluation 15-20 mins/participant**

- **Sprint 2 implementation with code reviews of L&D and BU SME**
  Developing Angular Application with all the best practices and integrating same with the Application developed in Sprint 1
  - Creating front end for the project using Angular
  - Code reviews
  - Performance monitoring during the sprint implementation and sharing the feedback
  - **Sprint - 2 Evaluation 15-20 mins/participant**