Index

JEE FULL STACK 2.0 WITH ANGULAR - (8 WEEKS)

JEE Full Stack 2.0 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	Discover (Induction)	1	
2	Soft Skills Foundation – Part 1	1	
3	Core Java 8 + Database & MySQL with DevOps (Git, Sonarqube, Gradle, Jenkin)	12.5	Project kick off-Individual project use cases has to be implemented
4	Core Java 8 Test	0.5	Coding and MCQ Test
5	Soft Skills Foundation – Part 2	1	
6	JPA with Hibernate 3.0	2	
7	Spring 5.0 (Core + MVC + REST + Data JPA + Data REST)	6	Wireframe diagram to be created for case study
8	Soft Skills Foundation – Part 3	1	
9	Sprint 1 + JPA and Spring MCQ Test	5	Sprint 1 - Backend implementation using Spring REST and Spring Data JPA
10	Sprint 1 Evaluation	1	
11	Soft Skills Foundation – Part 4	1	
12	HTML 5, CSS 3 with bootstrap 4, Javascript, Typescript	4	
13	Angular 7	6	
14	Sprint 2 + HTML 5, CSS 3 with bootstrap 4, Javascript, Typescript & Angular 7 MCQ	4	Sprint 2 - Front end implementation using Angular
15	Sprint 2 Evaluation	1	
16	L1 Preparation + L1 Test	2	
	Total Training Duration	49	

Curriculum

Agile SCRUM

Execution:

- **Sprint 1 implementation with code reviews of L&D and BU trainer**
    - Implementing Spring into the project
    - Test case reviews
    - Code reviews
    - Performance monitoring during the sprint implementation and sharing the feedback
    - **Sprint – 1 Evaluation 30min/participant**

- **Sprint 2 implementation with code reviews of L&D and BU trainer**
    - **Creating front end for the project using Angular**
    - **Code reviews**
    - **Performance monitoring during the sprint implementation and sharing the feedback**
    - **Sprint - 2 Evaluation 30min/participant**

Core Java 8

Program Duration: 12.5 days

Contents:

- **Declarations and Access Control**
    - Identifiers & JavaBeans
    - Legal Identifiers
    - Sun's Java Code Conventions
    - JavaBeans Standards
    - Declare Classes
    - Source File Declaration Rules
    - Class Declarations and Modifiers
    - Concrete Subclass
    - Declaring an Interface
    - Declaring Interface Constants
    - Declare Class Members
    - Access Modifiers
    - Nonaccess Member Modifiers
    - Constructor Declarations
    - Variable Declarations
    - Declaring Enums

- **Object Orientation**
    - Encapsulation
    - Inheritance, Is-A, Has-A
    - Polymorphism
    - Overridden Methods
    - Overloaded Methods
    - Reference Variable Casting
    - Implementing an Interface

- Legal Return Types
- Return Type Declarations
- Returning a Value
- Constructors and Instantiation
- Default Constructor
- Overloaded Constructors
- Statics
- Static Variables and Methods
- Coupling and Cohesion

- **Assignments**
  - Stack and Heap—Quick Review
  - Literals, Assignments, and Variables
  - Literal Values for All Primitive Types
  - Assignment Operators
  - Casting Primitives
  - Using a Variable or Array Element That Is Uninitialized and Unassigned
  - Local (Stack, Automatic) Primitives and Objects
  - Passing Variables into Methods
  - Passing Object Reference Variables
  - 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
  - Using Wrapper Classes and Boxing
  - An Overview of the Wrapper Classes
  - Creating Wrapper Objects
  - Using Wrapper Conversion Utilities
  - Autoboxing
  - Overloading
  - Garbage Collection
  - Overview of Memory Management and Garbage Collection
  - Overview of Java's Garbage Collector
  - Writing Code That Explicitly Makes Objects Eligible for Garbage Collection

- **Operators**
  - Java Operators
  - Assignment Operators
  - Relational Operators
  - instanceof Comparison
  - Arithmetic Operators
  - Conditional Operator
  - Logical Operators

- **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
- Unlabeled Statements
- Labeled Statements
- 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

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

- **TDD with Junit 5**
  - Types of Tests
  - Why Unit Tests Are Important
  - What's JUnit?
  - JUnit 5 Architecture
  - IDEs and Build Tool Support
  - Setting up JUnit with Maven
  - 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

- **Strings, I/O, Formatting, and Parsing**
  - String, StringBuilder, and StringBuffer
  - The String Class
  - Important Facts About Strings and Memory
  - Important Methods in the String Class
  - The StringBuffer and StringBuilder Classes
  - Important Methods in the StringBuffer and StringBuilder Classes
  - 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
  - Dates, Numbers, and Currency
  - Working with Dates, Numbers, and Currencies
  - Parsing, Tokenizing, and Formatting
  - Locating Data via Pattern Matching
  - Tokenizing

- **Generics and Collections**
  - Overriding hashCode() and equals()
  - Overriding equals()
  - Overriding hashCode()
  - Collections
  - So 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

- **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

- **Concurrent Patterns in Java**
  - Introducing Executors, What Is Wrong with the Runnable Pattern?
  - Defining the Executor Pattern: A New Pattern to Launch Threads
  - Defining the Executor Service Pattern, a First Simple Example
  - Comparing the Runnable and the Executor Service Patterns
  - Understanding the Waiting Queue of the Executor Service
  - Wrapping-up the Executor Service Pattern
  - From Runnable to Callable: What Is Wrong with Runnables?
  - Defining a New Model for Tasks That Return Objects
  - Introducing the Callable Interface to Model Tasks
  - Introducing the Future Object to Transmit Objects Between Threads
  - Wrapping-up Callables and Futures, Handling Exceptions

- **Concurrent Collections**
  - Implementing Concurrency at the API Level
  - Hierarchy of Collection and Map, Concurrent Interfaces
  - What Does It Mean for an Interface to Be Concurrent?
  - Why You Should Avoid Vectors and Stacks
  - Understanding Copy On Write Arrays
  - Introducing Queue and Deque, and Their Implementations

- Understanding How Queue Works in a Concurrent Environment
- Adding Elements to a Queue That Is Full: How Can It Fail?
- Understanding Error Handling in Queue and Deque
- Introducing Concurrent Maps and Their Implementations
- Atomic Operations Defined by the ConcurrentMap Interface
- Understanding Concurrency for a HashMap
- Understanding the Structure of the ConcurrentHashMap from Java 7
- Introducing the Java 8 ConcurrentHashMap and Its Parallel Methods
- Parallel Search on a Java 8 ConcurrentHashMap
- Parallel Map / Reduce on a Java 8 ConcurrentHashMap
- Parallel ForEach on a Java 8 ConcurrentHashMap
- Creating a Concurrent Set on a Java 8 ConcurrentHashMap
- Introducing Skip Lists to Implement ConcurrentMap
- Understanding How Linked Lists Can Be Improved by Skip Lists
- How to Make a Skip List Concurrent Without Synchronization

Database & SQL

Contents:

- **Introduction**
  - The Relational Model
- **Understanding Basic SQL 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

    - **Introduction to JDBC**
      - **Connection, Statement, PreparedStatement, ResultSet**

Introduction to Design Pattern

Self learning with online links and explanation by Trainer with Demos

- Creational Design Pattern
  - Factory Pattern
  - Singleton Pattern
  - Prototype Pattern
- Structural Design Pattern
  - Decorator Pattern
  - Facade Pattern
- Behavioral Design Pattern
  - Chain of Responsibility Pattern
  - Iterator Pattern
- Presentation Layer Design Pattern
  - Intercepting Filter Pattern
  - Front Controller Pattern
- Business Layer Design Pattern
  - Business Delegate Pattern
  - Transfer Object Pattern
- Integration Layer Design Pattern
  - Data Access Object Pattern

JPA with Hibernate 3.0

Program Duration: 2 days

Contents:

- **Introduction**
  - 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

Spring 5.0

Program Duration: 6 days

Contents:

1. **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

Spring Container

- The Spring Managed Bean Lifecycle
- Autowiring Dependencies

Dependency Injection

- Using the Application Context
- Constructor Injection
- Factory Methods
- Crucial Namespaces 'p' and 'c'
- Configuring Collections

Metadata / Configuration

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

2. **Spring Boot**

SPRING BOOT Introduction

- Spring Boot starters, CLI, Gradle plugin
- Application class
- @SpringBootApplication
- Dependency injection, component scans, Configuration
- Externalize your configuration using application.properties
- Context Root and Management ports
- Logging

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

3. **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

4. **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

HTML 5, CSS 3 with Bootstrap, Javascript, TypeScript

Program Duration: 4 days

Contents:

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 & Typescript**
  - 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 7

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, Creating Attribute Directive
  - Using Renderer to build attribute directive
  - Host Listener to listen to Host Events
  - Using Host Binding to bind to Host Properties

- **Pipes, Services & Dependency Injection**
  - In-built Pipes, Creating a Custom Pipes
  - Services & Dependency Injections
  - Creating Data Service
  - Understanding Hierarchical Injector

- **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
- Setting up Child (Nested) Routes
- Outsourcing Route Configuration (create custom module)

- **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