Index

JEE FULL STACK WITH DEVOPS & CLOUD(AWS)

JEE with DevOps & Cloud(AWS) variant provides exposure to the entire spectrum of Java technologies starting from Core Java to Spring. It focuses on Web Application development using DevOps & AWS and Spring Technology. The following table lists the course structure.

Course	Duration (Days)
Power Skills Touch Point - Foundation – session 1	1
Core Java 8 , Database PostgreSQL , DevOps, JDBC	3
Power Skills Touch Point Session 2	0.2
Core Java 8 , Database PostgreSQL , DevOps, JDBC	4.8
Module 1 MCQ Part 1 Test	**0.1**
Power Skills Touch Point Session 3	0.2
Core Java 8 , Database PostgreSQL , DevOps, JDBC-(Total duration:7.25)	0.7
Core Java 8 , Database PostgreSQL , DevOps, JDBC ..contd	6.5
Module 1 MCQ Part 2 Test + Coding Assessment	**0.5**
Power skills (Behavioural) -Foundation – session 4	1
JPA with Hibernate 3.0	2
Spring 5.0	5
Power skills (Behavioural) -Foundation – session 5	1
Spring 5.0	4
Power skills (Behavioural) -Foundation – session 6	0.2
Spring 5.0	0.8
Spring 5.0 + JPA MCQ	0.2
Docker	2.8
Power skills (Behavioural) -Foundation – session 7	0.2
Kubernetes	2.8
AWS with Container Deployment (EKS)	4
Power skills (Behavioural) -Foundation – session 8	1
Docker + Kubernetes + AWS MCQ	0.2
Sprint Implementation	5.8
Power skills (Behavioural) -Foundation – session 9	0.2
Sprint Implementation	1.8
Sprint Evaluation(Coding Assessment)	**1**
L1 Preparation	1
L1 Assessment (MCQ - Concept & Code-based Qs)	**1**
Total Training Duration	**53**

Agile SCRUM

Execution:

Agile Software Development Coursera Course on NEXT platform – 11 Hr

Agile SME connect for 1 Hr. – Day 1 of Sprint 1

Agile Lab 1 Hr – Day 1 of Sprint 1

- **Sprint 1 implementation for backend with code reviews of L&D and BU trainer**
  - Implementing Spring into the project
  - Test case reviews
  - Code reviews
  - Performance monitoring and review during the sprint implementation and sharing the feedback
- **Sprint implementation for Deployment on cloud.**
  - **Code reviews**
  - **Performance monitoring and review during the sprint implementation and sharing the feedback**

- **Complete Project evaluation - 30min/participant**

Core Java 8

Program Duration: 12 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

- **Gradle Fundamentals**
  - Introduction
  - Folder Structure
  - Install and Setup Gradle on Windows

  - Dependencies in Build Scripts
  - Gradle Wrapper
  - Lifecycle Tasks: The Base Plug In
  - Using Project Info and the check command
  - Creating Variables and external properties
  - Creating a Build Scan
  - Dependencies

- **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~~

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

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

DevOps (Git, SonarQube, Maven, Jenkins)

Duration : 1 day

Contents:

- **Introduction to DevOps**
    - Introduction of DevOps
    - Dev And Ops
    - Agile Vs DevOps
    - Continuous Integration & Delivery pipeline
    - Tools For DevOps
    - Use-case walkthrough
- GIT Hub
    - Working locally with GIT
    - Working remotely with GIT
    - Branching, merging & rebasing with GIT
    - Use Case walkthrough
- Jenkins:
    - Introduction to Jenkins
    - Jenkins Objective
    - Introduction to continuous integration deployment & Jenkins-ci
    - Continuous Deployment & distribution builds with Jenkins
- Sonar
    - Introduction to Sonar
    - Code quality Monitoring- Sonar
    - Use Case walkthrough

Database Using PostgreSQL and JDBC

Duration : 2.5 days

Contents:

- **Introduction**
  - The Relational Model
  - What is PostgreSQL?
  - PostgreSQL – Data Types
  - Arrays Functions and Operatorssss
- **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**
  - Wildcards (%, _)
  - The SELECT List
  - SELECT List Wildcard (*)
  - The FROM Clause
  - How to Constrain the Result Set
  - DISTINCT and NOT DISTINCT
- **Arrays Functions and Operators**
  - array_append
  - array_cat
  - array_lower
  - array_to_string
  - array_agg
  - every,Count,sum,avg
  - Array Operators
- **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**
  - Table Aliases
  - CROSS JOIN
  - INNER JOIN
  - OUTER JOINs

- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- SELF JOIN
- Natural Join

- **Creating Database Tables**
  - CREATE DATABASE
  - CREATE TABLE
  - NULL Values
  - PRIMARY KEY
  - CONSTRAINT
  - ALTER TABLE
  - DROP TABLE
- **PostgreSQL Transactions**
  - BEGIN, COMMIT, ROLLBACK

- **PostgreSQL Constraints**
  - CHECK, UNIQUE, NOT NULL

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

JPA Using PostgreSQL

Program Duration: 2 days

Contents:

- **Introduction to JDBC**
- **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: 10 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

- JSP
  - Writing Java Server Page
    - Developing a Simple Java Server Page
  - JSP Scripting Elements
    - Forms of Scripting Elements
    - Predefined Variables
    - Examples using Scripting Elements
  - JSP Directives
    - Page directive
    - Include directive
  - JSP Actions
  - JSP Standard Template Library (JSTL)
    - What is JSTL?
    - Installing JSTL
    - Using the Expression Language
    - 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
- ModelAndView
- Spring form tags and Model Binding, @ModelAttribute
- Data Validation

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. **Microservices**

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

Environment Management with Centralized Configuration

- Role of Configuration in microservices
- Spring cloud config
- Creating a configuration server
- Consuming configurations in apps

Performance Issues Using Distributed Tracing

- Role of tracing in microservices
- What is Spring Cloud Sleuth?
- Adding Spring Cloud Sleuth to a project
- Visualizing latency with Zipkin
- Adding Zipkin to a solution

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

Protecting Systems with Circuit Breakers

- Role of circuit breakers in microservices
- Describing Resilience4j
- Creating a Resilience4j service

Routing Your Microservices Traffic

- Role of routing in microservices
- Describing Spring Cloud LoadBalancer
- Configuring Spring Cloud LoadBalancer
- Describing Spring Cloud Gateway

Spring Security & Outh2

- Spring Security
- Spring Security with Spring boot
- Outh2
- Outh2 with Spring Boot

Docker

Program Duration: 3 days

Contents

- Introduction to Docker
  - Limitation of VM
  - Introduction to Container
  - Container Vs VM
  - What is Docker
  - Docker Community
  - ocker Architecture
  - Docker Installation

- Docker Platform overview
  - Docker Platform
  - Docker Engine
  - Docker Images
  - Docker containers
  - Registry
  - Repositories
  - Docker Hub
- **Deploying a Containerized App**

Module Overview
Warp Speed Run-through
Containerizing an App
Hosting on a Registry
Running a Containerized App
Managing a Containerized App
Multi-container Apps with Docker Compose
Taking Things to the Next Level with Docker Swarm
Microservices and Docker Services
Multi-container Apps with Docker Stacks
Docker Swarm

- Introduction to images and Reopository naming , Automated build, Private distribution

Kubernetes

Program Duration: 3 days

Contents

- **Introduction of Kubernetes**
  - What Is Kubernetes?
  - Kubernetes What and Why

- **Kubernetes Architecture**
  - Module Overview
  - Kubernetes Big Picture View
  - Kubernetes Masters
  - Kubernetes Nodes
  - The Declarative Model and Desired State
  - Kubernetes Pods
  - Stable Networking with Kubernetes Services
  - Game Changing Deployments
  - The Kubernetes API and API Server
  - Api Server
  - Scheduler
  - Controller Manager
  - etcd - the cluster brain

- **Getting Kubernetes**
  - Module Overview
  - Getting kubectl
  - Getting K8s in the Cloud

- **Working with Pods**
  - Module Overview
  - App Deployment Workflow

- Creating a Pod Manifest
- Deploying a Pod
- Deployment vs StatefulSet
- Pod Identity
- Scaling database applications: Master and Worker Pods
- Pod state, Pod Identifier
- 2 Pod endpoints

- **Kubernetes Deployments**
  - Module Overview
  - Kubernetes Deployment Theory
  - Creating a Deployment YAML
  - Deploying a Deployment
  - Self-healing and Scaling
  - Rolling Updates and Rollbacks

- **ClusterIP Services**
  - Service Communication
  - Multi-Port Services
  - Headless Services
  - NodePort Services
  - LoadBalancer Services

- Helm - Package Manager
  - Package Manager and Helm Charts
  - Templating Engine
  - Use Cases for Helm
  - Helm Chart Structure
  - Values injection into template files

AWS

Program Duration: 4 days

Contents:

- Cloud Basics
  - What is and Why Cloud?
  - Why Cloud Computing
  - Key characteristics of Cloud
  - Cloud Computing Architecture
  - Cloud Deployment and Service Model Selection criteria
  - Cloud APIs
  - Cloud benefits and Challenges
  - Different Cloud implementer
  - Latest trend
- Cloud Native Concepts

- Cloud technology
- Cloud Native Approach
- Purpose of Cloud Native
- What are Cloud Native companies doing differently to improve IT agility Benefits of Cloud native
- Hybrid cloud

- AWS Basics of different services
  - AWS history
  - Cloud Computing and Amazon Web Services
  - Functionality offered by AWS
  - The Differences that Distinguish AWS
  - Features of AWS service
  - Different AWS web services in Cloud
  - AWS global infrastructure
- Compute services
  - Amazon EC2

Container Deployment Service EKS

- Creation of an EKS cluster
- Configure kubectl using AWS CLI
- Serverless pods
- Scaling the cluster