JAVA

Trainer's name: [Banu] [Prakash]

Program objective:

Learn OOP using Java as programming language. Persist data using JDBC and ORM frameworks. Build RESTful web services using Spring and Spring Boot

Pre-requisite:

Knowledge of RDBMS, any Programming knowledge with basic understanding of programming constructs, using conditional statements and loops.

Duration of program

5 Days

H/w – S/w required for Lab Setup

Type of hardware	List of software
NA	**Operating System: Windows/MAC/ Linux/ Unix** **Web Browsers: Chrome, Firefox and IE.** **Editors: Sublime Text / Visual Studio Code.**

Target Audience

This tutorial is designed for anyone who will be using Java as programming language for developing Secure traditional web applications and RESTful web service with relational database and want to use Spring framework then this course covers them all. This course also will be useful to build modules using Java 8 features, handling concurrency issues and implementing good exception handling techniques

Day 1:

- Thinking Object-Oriented
  - A way of Viewing the World
  - Messages and Methods
  - Real world usage of interfaces to send messages
  - Real world understanding of Abstraction and Encapsulation
- Introduction
  - Features of Java
  - Java Architecture
  - JVM, JRE, JDK
- OO Programming in Java
  - Object
  - Class
  - Class Specification
  - Characteristics of an Object
  - Basic Principles of Object Orientation
  - Abstraction
  - Encapsulation
  - Modularity
  - Hierarchy
  - Polymorphism and overloading/overriding
  - Constructors
  - Static attributes and Static Methods
  - Packaging in Java
- Unit tesing
  - What is unit testing?
  - What is Junit?
  - Features of Junit
    - Fixtures
    - Test Suites
    - Test Runners
    - Junit classes

Day 2:

- Relationship between objects
  - Generalization and Specialization Relationship using inheritance
  - Uses of Abstract class and methods
  - Understanding Polymorphism and why we need it.
- Interfaces
  - What is an interface?
  - Uses of programming to interface
    - DESIGN, IMPLEMENTATION, TESTING and INTEGRATION
    - Implement OCP using interfaces

        - Writing loosely coupled applications using Interface
        - Unit Testing using Mockito
    - Inner Classes and Anonymous classes
    - Functional Interface
- Java 8
    - Lambdas: intro, motivation, and big idea
    - Lambda expression interpretation
    - Most basic form of lambdas
    - Type inferencing
    - Expression for body
    - Omitting parens
    - Using effectively final variables
    - @FunctionalInterface
    - Default and static methods in interface
    - Method references
    - java.util.function package
    - Lambda building blocks in java.util.function: overview
    - Predicate
    - Function and BiFunction
    - Consumer
    - Supplier
    - High Order functions, Function Composition and Currying
    - Variable scoping: Local variables, instance variables, "this"
    - Method references: details -- Class::staticMethod, variable::instanceMethod Class::instanceMethod, Class::new
    - New features in Java 8 interfaces: Default methods (also called "virtual extension methods" or "defender methods") and static methods
- Error Handling with Exceptions
        - Basic exceptions
        - Exception arguments
        - Catching an exception
        - The try block
        - Exception handlers
        - Creating your own exceptions
        - Catching any exception
        - Re-throwing an exception
        - Standard Java exceptions
        - Exception guidelines
- Annotations
    - Retention: SOURCE, CLASS and RUNTIME
    - Writing your own annotations and processing them using Reflection API

Day 3:

- Collections
    - Sorting and searching Lists Advanced Array Operations
    - Sorting arrays and the Equals Implementation
    - Containers
    - Filling Containers
    - Disadvantages of Object Containers
    - Making a type conscious Array List
    - Parameterized types
    - Iterators
    - Container Taxanomies
    - Collection Functionality
    - List Functionality
    - Stack and Queue Implementations
    - Set Functionality
    - Sorted Set
    - Map functionality
    - Generic Collections
- Java 8 Streams
    - Overview of Streams
    - Building Streams
    - Outputting Streams into arrays or Lists
    - Core Stream methods:
    - Intermediary methods: map, filter, skip, limit, flatMap
    - Terminal methods: reduce, forEach, collect, count
    - Lazy evaluation and short-circuit operations
    - Parallel Streams
    - Infinite Streams (really unbounded streams with values that are calculated on the fly)

Day 4:

- Maven
    - What and Why Maven?
    - The pom.xml file
    - Dependency Management
    - Configure plugins
    - Maven vs Gradle
- Introduction to JDBC
- Introduction to Servlets and web applications
- Spring Boot and JPA
    - Dependency Injection and IoC Container
    - ApplicationContext
    - Using a container
    - Bean Overview
        - Bean Scope
        - LifeCycle

    - Stereotype Annotations
        - @Component
        - @Repository
        - @Service
        - @Configuration
        - @Controller
        - @RestController
        - @Bean
- Spring Boot Configuration
    - Reducing Code with Custom Auto Configurations
    - Tuning your Auto Configuration
    - Enabling the Auto configuration Report
    - Excluding unnessessary Auto configurations
    - Tune auto configurations with application.properties
    - Overriding default configuration with command line arguments
- Spring Profiles
- Spring Data JPA
    - Introduction to ORM and entity mapping
    - Creating a Spring Data JPA Repository
    - Making Crud Operations with Repository
    - Adding Spring Data JPA
        - JPQL
        - Query Creation
            - Query creation from method names
            - Special parameter handling
                - Using Pageable, Slice, and Sort in query methods
            - Limiting Query Results
            - @Query
            - Streaming query results
    - Declarative Transaction Management
    - Domain Driven Design

Day 5:

- RESTful Web Services using Spring BOOT
    - Introduction to RESTful web services
    - Spring MVC Auto-configuration
    - @RestController
    - Handling HTTP methods
    - @RequestBody, @ResponseBody
    - @PathVariable, @QueryParam
    - Spring HATEOAS
    - Using Swagger to document RESTful web services
- Spring Boot: Unit Testing Rest Services

- Unit testing the GET Rest Service
  - @RunsWith: using SpringJUnit4ClassRunner to launch Spring TestContext Framework
  - @WebMvcTest: Unit testing Spring MVC Framework
  - MockMvc and MockBean: adding mocks to Spring ApplicationContext
  - Mockito: Mocking behaviour