Core Java & Spring

Program Duration: 10 days

Day-1,2

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

- **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

Day-3

- **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

Day-4

- **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()
- Synchronization and Locks
- Using notifyAll() When Many Threads May Be Waiting

- **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

Day-5

- **Lambda Expressions**
  - Introduction
  - Writing Lambda Expressions
  - Functional Interfaces
  - Types of Functional Interfaces
  - Method reference

- **Stream API**
  - Introduction
  - Stream API with Collections
  - Stream Operations

- **MYSQL DB**
  - DML, DDL, DRL Queries
  - Constraints
  - Joins

SPRING MODULES

Day-6

- **Spring Framework**
  - Introduction to Service Layer
  - Need of Spring Framework
  - Introduction to Spring Framework
  - Introduction to Spring IoC Container
  - Configuring and using Spring IoC Container
  - Java-Annotation based Configuration
  - Introduction to Auto wiring,
  - Using @Autowired annotation and other annotations

Day-7

- **Spring Boot**
  - What is Spring Boot?
  - Spring starter Maven Dependencies
  - Understanding @SpringBootApplication
  - Example of Spring MVC-based RESTful Web Service
  - Project Structure
  - Externalized Configuration application.properties and YAML
- **Actuators**
  - Exposing Information about your services
  - Customize Health and Info Endpoints
  - Custom Metrics and Custom Actuator
  - Prometheus Monitoring
- **Building Web Applications (MVC)**
  - Controllers and Model Attributes
  - Template Views
  - Using Embedded and External Databases
  - Spring Boot Initializers and Command Line Runners
  - Active Profiles

- Form Submissions

Day-8

REST Services

- What is REST?
- Restful Controllers
- JSON and XML Data Exchange
- Content Negotiation
- Working with exception handling
- Exception handling
- Angular JS Accessing your Services

- **Java Clients using RestTemplate**
  - ResponseEntity
  - Headers
  - Status codes
  - RequestEntity
  - Posting JSON to a Service
  - Integration with JMS and other backend services
- **Persistence with JPA Repositories**
  - JPA, EntityManagers, Entity Classes, Annotation mappings
  - Spring JPA Data, CrudRepository, PagingAndSortingRepository
  - Spring Data Rest, Exposing Databases as Endpoints
  - JSON
  - @Projections and Excerpts
    - Restrict the data sent back to the client
  - Spring Unit Testing

Day-9,10

- **Introduction to Microservices**
  - What are Microservices?
  - Decentralized Governance, Scalability, Fault Tolerance
  - Cloud Computing
  - Spring Cloud
  - Microservice Tradeoffs
  - Key Principles – DDD/Bounded Context
  - Key Patterns – API Gateway and Security
  - Third party Communication using Spring
  - Service and Client Discovery

- Netflix
- Microservices Design patterns
    - **Database per service pattern**
    - Saga pattern
    - Api gateway
    - Circuit breaker pattern
    - CQRS
    - Event Sourcing (Kafka)

- **Core Microservice Patterns using Spring Cloud and Netflix OSS**
    - Where are my Services?
        - Using Service Discovery
        - Eureka Servers and Clients
    - Scale Services
        - Load Balancing and Service Discovery
        - A LoadBalanced RestTemplate
    - Circuit Breaker, when services fail
        - Hystrix
        - Callbacks
    - Gateway/Edge Services – Providing an API
        - Zuul services
        - Filtering the Request and Response
    - Create Feign Clients to your Services
    - Messaging frameworks: Kafka
        - Keeping Services synchronized with each other
        - Creating Message Queues (Kafka)
        - Project development and deployment on Spring
        - Deploying to Docker container