Java with Framwork Development

Core Java

- Object Oriented Programming Concepts
    - Object
    - Characteristics of an Object
    - State
    - Behavior
    - Identity
    - Responsibility
    - Major Pillars
    - Abstraction
    - Encapsulation
    - 'IS A' relationship – Inheritance
    - 'HAS A' relationship – Containment
    - Polymorphism

- Introduction to Java
    - Features of Java
    - Java Compiler
    - Java Runtime Environment
    - Bytecode Verifier
    - Class Loader
    - JiT Compiler
    - JDK, JRE, JVM

- Java Language Fundamentals
    - Classes and Object
    - Class Syntax
    - Primitive Data Types
    - Implicit and Explicit Conversion
    - Access specifiers
    - Constructor
    - Creating object
    - Memory Allocation
    - Garbage Collection
    - Method Overloading
    - Parameterized Constructor
    - this keyword
    - static keyword – variable & method
    - main method

    - Instance init block & Static init block
    - Array – 1D, 2D
    - Enhanced for loop - forEach loop
    - Variable Argument
    - Package
    - Importing from packages
    - static import

- Containment, Inheritance and Polymorphism
    - Implementing 'HAS A' relationship
    - Container Object and Contained Object
    - Implementing 'IS A' relationship
    - extends keyword to achieve Inheritance
    - Super class and Sub class
    - super keyword
    - protected keyword
    - Polymorphism
    - Method Overriding
    - Super class reference to create Sub class object
    - Reference Type Casting
    - Static Type and Dynamic Type of reference
    - Covariant return type
    - java.lang.Object class and it's methods – toString()

- Abstract Class and Interface
    - abstract keyword
    - Abstract class vs Concrete class
    - Abstract method vs Concrete method
    - final keyword – variable, method, class
    - final method vs abstract method
    - final class vs abstract class
    - Interface
    - Role based inheritance
    - Multiple Inheritance
    - implements keyword
    - Implementing interface in a class
    - Interface extends interfaces
    -
- Annotations and Enum
    - Annotation
    - Creating user defined annotation
    - Use of annotation at different levels
    - Meta-annotation
    - java.lang.annotation.Annotation interface
    - Built-in Annotations - @Override, @SupressWarnings, etc
    - Enum

    - Creating user-defined Enum
    - Enum Constants
    - java.lang.Enum class

- Functional-Style Programming
    - Functional Interface
    - Lambda Expression
    - @FunctionalInterface
    - Method References
    - Built-in Functional interface
    - java.util.function package
    - Predicate, Consumer, Supplier, Function
    -
- Utility Classes (java.lang) & Inner classes
    - Wrapper classes
    - Autoboxing & Autounboxing
    - String, StringBuffer, StringBuilder
    - String constant pool
    - Inner classes
    - Simple Inner class
    - Static nested class
    - Method Local Inner class
    - Regex

- Exception Handling
    - What is exception?
    - Type of Errors
    - Exception class hierarchy
    - Exception Handling Mechanism
    - try keyword
    - catch keyword
    - Checked and unchecked exceptions
    - throw vs throws keyword
    - finally block
    - multicatch
    - final re-throw
    - try-with-resources (ARM)
    - User-defined exception
    -
- Multithreading & Thread Synchronization
    - Multitasking
    - Multiprocessing vs Multithreading
    - What is thread?

    - Thread Life Cycle
    - Creating thread using java.lang.Thread class
    - Creating thread using java.lang.Runnable interface
    - Thread class and it's methods
    - Inter thread communication
    - Thread Synchronization
    - synchronized keyword
    - synchronized method & synchronized block

- Collection Framework & Generics
    - Collection and it's types
    - java.util package
    - Generics Syntax
    - List – ArrayList, Vector
    - Set – HashSet, SortedSet, NavigableSet – TreeSet
    - Map – HashMap, SortedMap, NavigableMap – TreeMap
    - Hashtable and Properties class
    - Importance of equals() & hashCode() methods
    - Searching & Sorting Algorithm – Arrays & Collections classes
    - Comparable vs Comparator interfaces
    - Generic Collections

- Stream API
    - What is Stream?
    - Types of Stream – Sequential & Parallel Stream
    - java.util.stream package
    - Stream operations – intermediate and terminal operations
    - map(), reduce(), filter(), forEach(), limit(), skip() methods
    - Collectors – toCollection(), toList(), toSet(), toMap() methods
    -
- File IO & Object Serialization
    - java.io package
    - File class & it's method
    - File Reading, Writing and Appending Operation
    - Byte Stream – FileInputStream & FileOutputStream
    - Character Stream – FileReader & FileWriter
    - BufferedReader & BufferedWriter, PrintWriter
    - Autocloseable
    - Using try-with-resources
    - Non-blocking IO (nio)
    - java.nio.file package
    - Path interface
    - Files class, Paths class
    - FileSystem
    - Object Serialization
    - ObjectInputStream & readObject() method

    - ObjectOutputStream & writeObject() method
    - transient keyword
    -
- Unit Testing
    - Introduction
    - What is Testing?
    - Why Unit Testing?
    - Testing Terminologies
    - Junit Framework
    - Junit Annotations
    - Creating Test Cases
- Design Pattern
    - What is a Design Pattern?
    - Type of Design Patterns – Creational, Structural and Behavioral
    - Singleton Design Pattern
    - Factory Design Pattern
    - Facade Design Pattern

- Java9 version features
- Java11 version features

Adv Java

- Maven
    - What is Maven?
    - Features of Maven
    - Environment Setup
    - Build Life Cycle
    - Phases
    - Goals
    - Plugins
    - POM xml file
    - Dependency Management
    - Maven Dependencies
    - Maven Repositories
    - Maven Archetypes
    - Creating Maven Project using Eclipse

- JDBC
    - Java Database Connectivity
    - java.sql package
    - Connection interface
    - Driver interface

- Types of JDBC Drivers
- DriverManager class
- Connecting to MySQL using JDBC API
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- Types of ResultSet – Forward Only, Scrollable
- ResultSetMetaData
- DatabaseMetaData

- Introduction to Web Technologies
  - Distributed Architecture
  - Two-tier Architecture
  - Three tier (N-tier) Architecture
  - Web Application
  - Web Application Architecture
  - Web Container
  - Web Server
  - Web Browser
  - HTTP – Stateless Protocol
  - HTTP Methods – GET, POST, PUT, DELETE, etc
  - HTTP Message Format
  - Request Message Format – Request Header, Request Body
  - Response Message Format – Response Header , Response Body
  - Client Side Programming
  - Server Side Programming

- Servlet
  - Introduction to Servlet
  - Servlet API – Servlet, ServletConfig interfaces
  - GenericServlet
  - HttpServlet
  - Servlet Life Cycle
  - ServletContext vs ServletConfig
  - Servlet Configuration - web.xml
  - @WebServlet Annotation
  - Creating Servlet
  - Configuring Tomcat Server in Eclipse IDE
  - Deploying and Running Servlet on Tomcat via Eclipse

- JSP
  - Introduction to Java Server Pages
  - JSP API
  - JSP Life Cycle
  - Creating and Running JSP Page on Tomcat Server

- Building Blocks of JSP
- JSP Directives
- JSP Implicit Objects
- JSP Scripting Elements
- Scriptlet
- JSP Expression

- Web Service
  - What is Web Service?
  - Web Service vs Web Application
  - SOA – Service Oriented Architecture
  - Service Provider
  - Service Consumer
  - Types of Web Service
  - SOAP Web Service
  - RESTful Web Service
  - ROA – Resource, URI and Representation
  - Uniform Interfaces
  - Creating REST API
  - Running RESTful Web Service on Tomcat
  - Consuming REST API

Java Framework - Hibernate JPA

- Object Relationship Mapping
  - ORM Principle
  - Drawbacks of JDBC
  - Benefits of ORM
  - Paradigm Mismatch
  - Types of ORM Frameworks

- Hibernate JPA Introduction
  - What is Hibernate?
  - Features of Hibernate
  - Advantages of Hibernate
  - Hibernate in Layered Application Design
  - Hibernate Architecture
  - Components of Hibernate
  - Configuration
  - SessionFactory
  - Session and it's methods – get(), load(), save(), update(), delete()
  - What is JPA?
  - Hibernate and JPA

- Simple Class Mapping
  - Entity Class
  - POJO class and it's advantages
  - Hibernate JPA Mapping
  - Mapping POJO class with Database Table
  - JPA Annotations - @Entity, @Table, @Id, @Column
  - Hibernate Configuration
  - hibernate.cfg.xml file
  - Configuring JDBC Driver, JDBC URL
  - Hibernate Dialect
  - Creating Hibernate Application
  - Performing CRUD operations

- Hibernate JPA Entity Life Cycle
  - Transient State
  - Persistent State
  - Detached State

- JPQL
  - Java Persistent Query Language
  - From clause
  - Where clause
  - Order By
  - Select clause
  - Using Criteria

- Component Mapping
  - Mapping 'HAS A' relationship
  - Creating Contained class
  - Creating Container class
  - Using JPA Annotations - @Embedded, @Embeddable

- Inheritance Mapping
  - Types of Inheritance Mapping
  - Table Per Class Hierarchy
  - JPA Annotations - @Inheritance
  - Inheritance Strategy – InheritanceType.SINGLE_TABLE
  - @DiscriminatorColumn, @DiscriminatorValue
  - Table Per Concrete Class
  - Inheritance Strategy – InheritanceType.TABLE_PER_CLASS
  - Table Per Joined Subclass
  - Inheritance Strategy – InheritanceType.JOINED
  - @PrimaryKeyJoinColumn

- Relationship Mapping
  - One-to-One Mapping

        - @OneToOne, mappedBy
        - @JoinColumn
    - One-to-Many Mapping
        - @OneToMany
    - Many-to-One Mapping
        - @ManyToOne
    - Many-to-Many Mapping
        - @ManyToMany
        - @JoinTable

- Hibernate Caching Mechanism
    - First-Level Cache
    - Second Level Cache

Java Framework – Spring & Spring Boot

- Introduction to Spring
    - What is Spring?
    - Features of Spring Framework
    - IoC – Inversion of Control
    - DI – Dependency Injection
    - AOP – Aspect Oriented Programming

- Spring Architecture
    - Types of Spring Module
    - Spring IoC Container
    - Spring Core - BeanFactory
    - Spring Context - ApplicationContext
    - Spring AOP
    - Spring DAO - JdbcTemplate
    - Spring ORM - HibernateTemplate
    - Spring Web
    - Spring Web MVC

- Spring Bean and Configuration
    - Creating Spring Bean
    - Spring Bean Configuration
    - Xml Based Configuration
    - Annotation Based Configuration
    - Stereotype Annotation - @Component
    - @Configuration
    - @ComponentScan
    - JavaConfig - @Bean
    - Bean Scoping
    - Bean Scopes – singleton, prototype, request, session

- @Scope Annotation

- Spring Bean Life Cycle
  - Spring Aware Interfaces
  - BeanNameAware
  - BeanFactoryAware
  - ApplicationContextAware
  - InitializingBean – afterPropertiesSet()
  - @PostConstruct
  - Custom Init Method
  - DisposableBean
  - Custom Destroy Method

- Dependency Injection
  - Constructor Injection
    - constructor-arg
  - Setter Injection
    - property
  - Bean Autowiring
    - @Autowired
    - @Qualifier
  - Method Injection
  - Collection Injection
    - List Injection
    - Set Injection
    - Map Injection
    - Properties Injection

- Spring Boot
  - Introduction to Spring Boot
  - Features of Spring Boot
  - Spring Boot Starter Dependencies
  - @SpringBootApplication
  - AutoConfiguration
  - Creating Spring Boot Application
    - Spring Initializr – https://start.spring.io/
    - STS - Spring Tool Suite
    - Selecting starter dependencies

- Spring Data
  - What is Spring Data?
  - Features of Spring Data
  - Spring Data JDBC
    - Repository interface and it's hierarchy
    - Creating custom Data Repository interface

        - Performing CRUD Operations
        - Custom methods
        - @Query, @Param
    - Spring Data JPA
        - JpaRepository interface
        - Creating custom Data JPA Repository interface
        - Performning CRUD Operations
        - Customer methods
        - JPQL Queries
        - Transaction Configuration
- Spring Web
    - Spring Web MVC
        - MVC Architecture
        - Spring Web Architecture
        - DispatcherServlet
        - HandlerMapping
        - ViewResolver
        - Controller - Request Handler
        - @Controller Stereotype Annotation
        - @RequestMapping
    - RESTful Web Service
        - @RestController
        - @GetMapping
        - @PostMapping
        - @PutMapping
        - @DeleteMapping
        - Creating REST API

- Spring Security
    - Authentication
    - Authorization
    - Roles
    - Permissions
    - Features of Spring Security
    - Authentication Models
    - Password Encoders
    - User Store - Types
    - OAuth2

- Spring Microservices:
    - Overview of Microservices Architecture
    - Definition of Microservices
    - Benefits and challenges
    - Monolithic vs. Microservices architecture

- API Gateway Pattren
- Service discovery tools (e.g., Eureka)
- Resilience and Fault Tolerance
- Circuit Breaker Pattern

DevOps

- Introduction to DevOps
    - Git and Github
    - CI/CD tools with jenkins
    - Docker and kubernetes
    - Sonarcube