Java SE 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