Lara Technologies
Java Fullstack Course Content
https://lara.co.in
7975938871

Core Java:

1. Introduction to Java:

-> Historical Development of Computers over the years

-> Assembler and Compiler

-> Machine Level Language (MLL), Assembly Level Langauge (ALL) and High Level Langauage (HLL)

-> Historical Development of High Level Languages

-> Java's origin and history: Developed by James Gosling at Sun Microsystems (now owned by Oracle)

-> Java's Platform Independency

-> Java's "Write Once, Run Anywhere" philosophy through the Java Virtual Machine (JVM)

-> Java's main components: Java Development Kit (JDK), Java Runtime Environment (JRE)

-> Different Vesions of Java and Latest version of Java

-> Differences between JDK, JRE and JVM

-> Memory Management

-> Compiler vs Interpreter

-> Java Features / Java Buzz words

-> JVM Architecture

-> Major Differences between C++ and Java

-> Case studies

2. Java Installation :

-> Downloading and Installing Java

- -> Setting up Environment variables
- -> Different platforms available to prepare Java Applications
  - -> Notepad
  - -> Edit Plus
  - -> IDE's like Eclipse, STS, IntelliJ
- -> Descriptions of .java file, .class file, .obj file, .exe file
- -> Writing First JAVA program.
- -> Complete flow of Execution of a Java program
- -> Compiling and executing Java code from the command line
- -> Understanding the structure of a Java program
  - -> Comment Section
  - -> Package declarations
  - -> import statements section
  - -> classes / interfaces / abstract classes / enums section
  - -> main class section which would contain main( ) method
- -> Java naming conventions for
  - -> classes, abstract classes, interfaces, enums
  - -> methods
  - -> variables
  - -> constants
  - -> packages
- -> Java .class files generating process
- -> Saving a java program
- -> public classes concept in Java
- -> Command Line Arguments in Java
- -> Case Studies

3. Operators :

- -> Different Number Systems followed by Java
  - -> Octal, Decimal, Binary and Hexa Decimal
  - -> Case Studies
- -> Definitions of Unary, Binary and Ternary Operators

- -> Arithmetic operators
  - -> +, -, *, /, %
  - -> Increment and Decrement operators(++ and --)
  - -> Differences between pre-increment and post-increment operators
  - -> Detailed Case Studies
- -> Assignment operators
  - -> Simple Assignment operator and Compound Assignment Operators
  - -> =, +=, -=, *=, /=, %=
- -> Relational (Comparison) operators
  - -> ==, !=, <, >, <=, >=
- -> Logical operators
  - -> Logincal AND (&&) ,
  - -> Logical OR (||)
  - -> NOT operator (!)
- -> Bitwise operators
  - -> &, |, ^, <<, >>
- -> Difference between & and &&, | and ||
- -> Operator precedence and associativity

4. Data Types and Variables :

- -> Primitive data types
  - -> byte, short, int, long, float, double, char
  - -> Range of each data type
  - -> Memory occupation
  - -> Default values of each type
  - -> Detailed Case Studies
- -> Typecasting
  - -> Definition of Primitive Level and Object level Typecasting
  - -> Implicit and Explicit Typecasting

- -> Reference data types
- -> Overview of Object orientation
- -> Objects and references
- -> Variable naming conventions and rules
- -> Instance variables and Local variables
- -> Initializing variables
- -> Differences between declaring and defining
- -> Default values and explicit initialization

5. Control Statements :

- -> Definition of Control statements
- -> Different Control constructs
  - -> simple if
  - -> if-else
  - -> nested if-else
  - -> else-if ladder
- -> Case Studies
- -> switch control construct
- -> Changes wrt swtich happened in different versions of Java
- -> Case Studies
- -> Loops in Java
  - -> for loop
  - -> while loop
  - -> do-while loop
  - -> Differences between whild and do-while
  - -> for-each loop
- -> Detailed Case Studies
- -> break and continue statements within loops
- -> labelled break and continue
- -> Case Studies

6. Arrays in Java :

- -> Object creation process

- -> Detailed description about JVM memory
    - -> Method area
    - -> Heap area
    - -> Stack area
    - -> Native methods stack area
    - -> PC Register area
- -> Activation Records
- -> Memory allocation for local variables, instance variables and reference variables
- -> Problems associated with traditional approach of storing data by creating variables
- -> Advantages of using array approach for storing data
- -> 1-D array
    - -> Declaring, Defining and Accessing the data
- -> 2-D array
    - -> Declaring, Defining and Accessing the data
    - -> Memory allocation
- -> Jagged Arrays in Java
    - -> 2-D jagged array
    - -> Memory allocation
- -> 3-D Regular array and jagged array
    - -> Memory allocation
- -> Disadvantages associated with array apporach and how to overcome
- -> Case Studies

7. String Manipulation in Java :

- -> Definition of string
- -> Mutable and immutable strings
- -> Various ways of comparing two strings
- -> Memory allocation for string
    - -> Detailed discussion about Heap Area and SCP area
    - -> Detailed case studies
- -> String class to create immutable string objects
- -> Important constructors supported by String class

-> Important built-in methods present in String class (35+)

-> String.join( ) method

-> Creation of customized immutable class

-> Importance of SCP area

-> StringBuffer and StringBuilder classes

-> Need of introducing StringBuilder class

-> Differences between StringBuffer and StringBuilder classes

-> Behaviour of equals( ) method in String and StringBuffer class

-> Important built-in methods present in StringBuffer and StringBuilder classes

-> Case Studies

8. Methods in Java :

-> Definition of method.

-> Description about method signature

-> Different categories of methods

    -> method with no parameter, no return type

    -> method with parameter but no return type

    -> method with no parameter but return type

    -> method with parameter and return type

-> Passing primitive varialbes as the parameters

-> Passing objects as the parameters

-> Case Studies

9. Method overloading and var-args methods in Java :

-> Definition of method overloading

-> Handling multiple methods in C and C++

-> Handling multiple methods in Java

-> Advantages of method overloading

-> Definition and examples for polymorphism

-> Why method overloading is compile time polymorphism?

-> Why the name "method overloading" ?

-> Method overloading with numeric promotion

-> Problems associated with method overloading with numeric promotion

-> Introduction to var-args method

-> Case Studies

-> Rules associated with var-args method

-> Different chances of getting java.lang.NullPointerException

10. OOP's Principles :

-> Pillars of Object Orientation

-> Differences between Procedure Oriented Programming (POP) and Object Oriented Programming (OOP)

-> Encapsulation

- -> Data hiding
- -> Abstraction
- -> How to achieve encapsulation
- -> Tightly encapsulated classes
- -> Detailed case studies

-> setters and getters methods

-> accessors and mutators

-> Shadowing problem

-> "this" keyword description

-> Constructor - a specialized setter

- -> Definition and detailed description about constructors
- -> zero-param and param constructors
- -> Rules associated wrt constructors

-> Execution of constructors during object creation

-> Differences between constructors and ordinary methods

-> Instance variables, Instance Intializer Block (IIB) and Instance methods

-> Detailed flow of execution of Instance context

-> Inheritance Definition, overview and Advantages (Detailed discussion in upcoming chapter)

-> Polymorphism Definition, overview and Advantages (Detailed discussion in upcoming chapter)

-> Object class and its methods

-> Case Studies

11. static keyword and static import :

-> static context

-> static variables

-> static blocks

-> static methods

-> class loading process

-> Class.forName( ) method

-> Memory allocation for static variables

-> Detailed flow of execution of static context

-> Multiple static blocks

-> Advantages of using static variables

-> Instance block and static blocks in combination

-> Rules associated to access instance context and static context

-> Differences between instance variables and static variables

-> Real time application development to demostrate the usage of static context

-> Program to count the number of objects created for any particular class

-> static or utility methods

-> When to use and when not to use static context

-> Introduction to static import

-> Normal import statements

-> Implicit and explicit import statements

-> static import statements

-> Implicit and explicit static import statements

-> built-in static import and customized static import

-> static import statement resolvation in different cases

-> Problems associated with static import

-> Detailed Case Studies

12. Access Modifiers in Java :

-> Introduction to packages in java

-> Definition

-> Advantages of using packages

- -> Types of packages in java
- -> How to create user-defined packages
- -> Detailed case studies using user-defined packages
- -> Jar files and War files in Java
- -> Detailed descriptions about
    - -> public
    - -> protected
    - -> default
    - -> private
    - -> abstract
    - -> final
    - -> static
    - -> synchronized
    - -> native
    - -> strictfp
    - -> volatile
    - -> transient
- -> class level modifiers
- -> method level modifiers
- -> field level modifiers
- -> Detailed Case Studies
- -> Valid and invalid combination access modifiers

13. Inheritance, Method overriding and Polymorphism in Java :

- -> Definition of inheritance
- -> Rules associated wrt inheritance
- -> Types of Inheritance
- -> Execution of constructors in case of inheritance
- -> Diamond shaped problem
- -> super( ) method and this( ) method
- -> Constructor chaining and local chaining
- -> super keyword

-> Differences between super( ) method and this( ) method

-> Differences between "super" keyword and "this" keyword

-> Inherited, overridden and specialized methods

-> Method overriding

-> Runtime Polymorhism or late binding

-> Parent reference to child object

-> Rules associated with parent reference to child object

-> Rules associated with method overriding

-> Method Hiding

-> Differences between Method overriding and Method hiding

-> Differences between Method overloading and Method overriding

-> Delegation Model in Java

-> Has-A Relationship in Java

-> Detailed Case Studies

14. Abstract classes and Interfaces :

-> Introduction to Abstract Classes

-> Understanding the need for abstraction in object-oriented programming

-> Differences between concrete classes and abstract classes

-> Declaring Abstract Classes Using the "abstract" keyword

-> Defining abstract methods within abstract classes.

-> Abstract Class Members

-> Implementing instance variables in abstract classes

-> Defining concrete (non-abstract) methods within abstract classes.

-> Access modifiers and abstract class members.

-> Inheritance with Abstract Classes

-> Extending abstract classes with concrete subclasses.

-> Overriding abstract methods in concrete subclasses.

-> Inheriting and utilizing concrete methods from abstract classes.

-> Partial Implementation of Abstract classes

-> Constructors in Abstract Classes

-> Constructor chaining between abstract and concrete classes.

-> Initializing instance variables in abstract class constructors.

-> Abstract Classes and Interfaces Relationship

-> Comparing abstract classes and interfaces.

-> Choosing between abstract classes and interfaces for different scenarios.

-> Combining abstract classes and interfaces in class hierarchies.

-> final keyword

-> Detailed Case Studies

-> Introduction to Interfaces

-> Defining the concept and purpose of interfaces.

-> Exploring the role of interfaces in achieving multiple inheritance.

-> Declaring Interfaces

-> Using the interface keyword to declare interfaces.

-> Defining method signatures within interfaces.

-> Implementing Interfaces in concrete classes using the implements keyword

-> Providing implementations for methods declared in interfaces

-> Rules associated wrt Interfaces

-> Default Methods and static methods in interfaces

-> Accessing default methods and static methods present in interfaces

-> Case Studies

-> Constant Values in Interfaces

-> Differences between constants in interfaces and other classes.

-> Functional Interfaces

-> Defining functional interfaces and their characteristics.

-> Marker Interfaces

-> Explaining the concept of marker interfaces.

-> Role and use cases of marker interfaces in Java.

-> Notable examples of marker interfaces in the Java API.

-> Inheritance and Interfaces

-> Extending multiple interfaces in a single class.

-> Handling method conflicts when implementing multiple interfaces.

-> Diamond shaped problem and ways to mitigate it using interfaces.

-> Abstract Classes vs. Interfaces

15. Wrapper classes in Java :

-> Introduction to Wrapper Classes

-> Concept of wrapper classes and their purpose in Java

-> Need to represent primitive data types as objects

-> Discussing scenarios where wrapper classes are useful

-> Primitive Data Types and Their Limitations

-> Creating Wrapper Objects

-> Eight wrapper classes for primitive data types.

-> Relationship between primitive types and their corresponding wrapper classes.

-> Creation of instances of wrapper objects.

-> Autoboxing and Unboxing

-> Wrapper Class Constructors

-> Discussing various constructors available in wrapper classes

-> Exploring constructor overloads and how to initialize wrapper objects

-> Common Methods in Wrapper Classes

    -> valueOf( ) method

    -> toString( ) method

    -> equals( ) method

    -> hashCode( ) method

-> Number and Character Wrapper Classes

-> Exploring the Number class hierarchy : Integer, Double, Float, etc.

-> Demonstrating methods like intValue( ), doubleValue( ) and compareTo( ) in Number

-> Introducing the Character wrapper class and its methods

-> Boolean Wrapper Class:

    -> Exploring the Boolean wrapper class and its methods

    -> Concept of autounboxing in boolean wrappers

-> Wrapper Classes for Conversions

-> Methods for converting strings to primitive types using wrapper classes

    -> parseXXX( ) i.e. parseInt( ), parseDouble( ) etc.

    -> valueOf( )

-> Wrapper Classes and Collections

-> Usage of wrapper classes in collections and generics

16. Design Patterns in Java :

-> Introduction to Design Patterns:

-> Understanding the need for design patterns.

-> Advantages of using design patterns in software development.

-> Three categories of design patterns

- -> Creational Design Pattern
- -> Structural Design Pattern
- -> Behavioral Design Pattern

-> Creational Design Patterns

- -> Singleton Pattern purpose
- -> Implementing the singleton pattern to ensure a single instance of a class
- -> Handling issues like thread safety, lazy initialization, and eager initialization

-> Factory Method Pattern

- -> Understanding the factory method pattern
- -> Implementing factory methods to create objects without specifying the exact class
- -> Comparing factory methods to constructors and abstract factories.

-> Prototype Pattern

- -> Prototype pattern purpose and advantages
- -> Implementing prototypes to create new objects by cloning existing ones
- -> shallow cloning and deep cloning

-> Strategy Design Pattern

-> Comparing the strategy pattern to inheritance and encapsulation

17. Garbage Collection in Java :

-> Introduction to Garbage Collection

-> Understanding the role of memory management in programming languages.

-> Concept of Garbage Collection (GC) and its significance in Java.

-> Need for automatic memory management to prevent memory leaks.

-> Memory Management in Java

- -> Java memory model stack vs. heap

-> Object allocation and deallocation in Java

-> Object Lifecycle:

-> Object creation, reference, dereference, and destruction.

-> Concept of reachability and how it relates to garbage collection.

-> Types of Garbage Collectors

-> Serial Garbage Collector

-> Parallel Garbage Collector

-> G1 (Garbage-First) Garbage Collector

-> Z Garbage Collector

-> Goals of the Z Garbage Collector

-> Reachability Analysis

-> Identifying root objects

-> local variables, static variables, and method call stacks

-> The role of object references in establishing reachability

-> Memory Leak Detection and Prevention

-> Garbage Collection in Modern Java Versions

-> Updates and improvements to Garbage Collection in recent Java versions

18. Regular Expressions (Regex) in Java :

-> Introduction to Regular Expressions

-> Purpose and benefits of regular expressions

-> Introducing the concept of pattern matching in text data.

-> Basic Regular Expression Syntax

-> Basic regex metacharacters and their meanings.

-> Exploring character literals, character classes, and quantifiers

-> Special characters like ^, $, ., *, +, ?, etc.

-> Pattern and Matcher Classes in the java.util.regex package

-> Exploring the process of compiling regex patterns and matching against input text

-> Character Classes and Quantifiers

-> Understanding character classes like \d, \w, \s, and their negations

-> Exploring quantifiers like {n}, {n,}, {n,m} for repetition

-> Anchors and Boundary Matchers

-> Exploring anchors like ^ (start of line) and $ (end of line).

-> Introducing word boundaries \b and non-word boundaries \B.

-> Capturing and Non-Capturing Groups

-> Introducing capturing groups for extracting matched portions

-> Exploring non-capturing groups using (?:...)

-> Discussing the role of captured groups in replacement operations

-> Alternation and Pipe Symbol:

-> Explaining alternation using the | (pipe) symbol

-> Discussing how to use parentheses with alternation

-> Escaping Special Characters:

-> Need to escape special characters in regular expressions

-> Backslash (\ ) as escape character

-> Predefined Character Classes

-> Predefined character classes like \d, \w, and \s

-> Exploring their negations \D, \W, and \S

-> Greedy and lazy quantifiers

-> Differences between *, *?, +, +?, ?, and ??.

-> Exploring the replaceAll() and replaceFirst() methods

-> Lookahead and Lookbehind

-> Introducing positive and negative lookahead assertions

-> Explaining positive and negative lookbehind assertions

-> Regular Expressions in Java API

-> Exploring regex-related methods in the String class (e.g., matches(), replaceAll(), split()) etc.

<u>19. Assertions in Java :</u>

-> Introduction to Assertions

-> Purpose and benefits of assertions in Java

-> Runtime checks for debugging and validation

-> Enabling and Disabling Assertions

-> How to enable and disable assertions in Java

-> The -ea and -da command-line options for enabling and disabling assertions

-> "assert" Keyword for creating assertions

-> Using assert with boolean expressions to ensure expected behavior

-> Assertion Failure and Error Messages

    -> Understanding what happens when an assertion fails

-> Providing custom error messages for better debugging

-> Evaluating Expressions in Assertions

-> Discussing method preconditions

    -> conditions that must be satisfied before a method executes

-> Introducing method postconditions

    -> conditions that must be true after a method's execution

-> Appropriate ways of using assertions

-> Avoiding side effects in assertions

-> Ensuring assertions are enabled during development and testing

20. Generics and Inner classes in Java :

-> Introduction to Generics

-> Need for generics to create flexible and type-safe code

-> Type parameters and generic classes/methods

-> Generic Classes and Type Parameters

    -> Defining generic classes using type parameters

    -> Creation of instances of generic classes with different types

    -> Use of type parameters in class members and methods

-> Type Inference and Diamond Operator (<>)

-> Bounded Type Parameters

-> Upper and lower bounds for type parameters

-> Wildcards and Unbounded Generics

    -> Introducing wildcards (?) in generics to represent unknown types

-> Generic Methods

    -> Defining generic methods that accept and return generic types

-> Erasure and Type Safety

    -> Type erasure in Java's generics during compilation

-> Generics and Subtyping

-> Relationship between generic types and subtyping

-> Covariance and contravariance in relation to generics

-> Introduction to Inner Classes

-> Concept of inner classes and their role in encapsulation

-> Different types of inner classes such as member, local, anonymous, and static

-> Member Inner Classes

-> Member inner classes and their relationship with outer classes

-> Creation of instances for member inner classes

-> Access rules and visibility of member inner classes

-> Local Inner Classes

-> Local inner classes defined within method bodies

-> Scope and lifetime of local inner classes

-> Demonstrating how local inner classes can access local variables

-> Anonymous Inner Classes

-> Anonymous inner classes used for creating one-time implementations

-> Syntax and creation of anonymous inner classes

-> Static Nested Classes

-> Static nested classes defined within a class

-> Differences between static nested and member inner classes

-> Inner Classes and Encapsulation

-> How inner classes facilitate encapsulation by providing private access

-> Use of inner classes to model more complex relationships

-> Outer Class Access from Inner Classes

-> Demonstrating how inner classes can access members of their outer classes

-> Exploring how inner classes maintain a reference to their enclosing instance

-> Local Inner Classes and Final Variables:

-> Requirement for local variables accessed by local inner classes to be final or effectively final

-> Anonymous Inner Classes and Interfaces

-> Creating instances of anonymous inner classes implementing interfaces

21. Annotations in Java :

-> Introduction to Annotations

-> Concept and purpose of annotations in Java

-> Advantages of using annotations for metadata

-> Built-in Annotations

-> @Override

-> Understanding the @Override annotation for indicating method overriding

-> How the @Override annotation helps prevent accidental method name changes

-> @Deprecated

-> @Deprecated annotation to mark elements as obsolete

-> How the @Deprecated annotation affects compilation and warnings

-> @SuppressWarnings

-> @SuppressWarnings annotation to suppress compiler warnings

-> How to apply the @SuppressWarnings annotation to different elements

-> Custom Annotations

-> Declaring Custom Annotations

-> Process of creating custom annotations

-> Syntax of declaring custom annotations

-> meta-annotations used to annotate annotations

-> Element Types and Values

-> Different types of elements that can be annotated

-> Defining default values for annotation elements

-> Annotation Retention and Targets

-> Retention policies (@Retention) for annotations

-> Different target types (@Target) for annotations

-> Meta-Annotations

-> Annotations used to annotate other annotations

-> Annotation Processors

-> Annotation processors and their role in code generation and analysis

22. Exception Handling in Java :

-> Introduction to Exception Handling

-> Need for exception handling in Java programming

-> Computer Architecture

  -> Stand alone architecture

  -> Client-Server architecture

  -> N-tier architecture

-> Concept of exceptions and their classifications

-> Scenarios where exception handling is essential

-> Checked and Unchecked Exceptions

-> Differences between checked and unchecked exceptions

-> Role of the throws clause in method signature

-> The try, catch, and finally blocks

  -> try block for enclosing code that might throw exceptions

  -> catch block for handling exceptions

  -> finally block for code that should execute regardless of exceptions

-> Handling Exceptions with catch

  -> Syntax of the catch block and parameterized exception types

  -> Handling different types of exceptions using multiple catch blocks

-> Order of catch blocks and best practices

-> finally block

  -> Introducing the finally block and its role in resource cleanup

  -> How to use the finally block in various scenarios

  -> How finally works with return statements and exceptions

-> Valid and invalid combinations of try-catch-finally

-> Throwing Exceptions

  -> How to manually throw exceptions using the throw keyword

-> When and how to throw exceptions to indicate exceptional conditions

-> Exception Chaining and getCause( ) method

-> Runtime Errors

-> try-with-resources concept

  -> try-with-resources statement for automatic resource management

-> How to use the AutoCloseable interface with try-with-resources

-> Demonstrating the benefits of try-with-resources for resource cleanup

-> Custom Exception Classes

-> Concept of custom exception classes

-> When and how to create custom exception classes for specific scenarios

-> Providing meaningful error messages

-> Throwable class hierarchy

-> Differences between final, finally and finalize( ) method

-> Exploring real-world scenarios where exception handling is crucial

23. Multithreading in Java :

-> Introduction to Multithreading

-> Single Tasking and Multi Tasking Operating Systems

-> Concept of multithreading and its advantages

-> Process based multitasking and thread-based multitasking

-> Threads as lightweight execution units within a process

-> Scenarios where multithreading is useful

-> Creating and Running Threads

-> Thread class for creating and managing threads

-> Different ways to create threads

-> Extending Thread class

-> Implementing Runnable interface

-> How to start and manage threads using the start( ) method

-> Thread Lifecycle and different states of a thread

-> Synchronization and Race Conditions

-> Concept of race conditions and thread interference

-> Need for synchronization to ensure data consistency in multithreaded programs

-> How to use the synchronized keyword to synchronize methods and blocks

-> Thread Safety and Shared Resources

-> Thread Priorities

-> Thread priorities and their significance

-> Thread priority levels and their relationship with the thread scheduler

-> Thread.MIN_PRIORITY, Thread.MAX_PRIORITY, and Thread.NORM_PRIORITY

-> Thread Coordination

-> To achieve thread coordination using methods like join(), sleep(), wait(), notify(), and notifyAll()

-> Producer-Consumer problem

-> Inter-thread communication

-> Deadlock and Starvation

-> Common scenarios that lead to deadlock and starvation

-> Callable interface

-> Differences between Callable call( ) method and Runnable run( ) method

-> Thread Pools and Executors

-> Thread pools and the Executor framework

-> ExecutorService, ThreadPoolExecutor, and ScheduledExecutorService

-> How to create and manage thread pools to improve resource usage

-> Thread Local Variables

-> volatile Keyword

24. Collections Framework in Java :

-> Introduction to the Collection Framework

-> Disadvantages associated with arrays to store the data

-> Advantages of using Collection Framework

-> Collection interface and Collections class

-> Collection Interfaces

-> List Interface

-> ArrayList, LinkedList, and Vector implementing the List interface

-> Demonstrating methods for adding, retrieving, and modifying elements in a list

-> Set Interface

-> Set interface for collections without duplicates

-> HashSet, LinkedHashSet, and TreeSet implementing the Set interface

-> Demonstrating methods for adding, retrieving, and verifying elements in a set

-> Queue and Deque Interfaces

-> Queue and Deque interfaces for FIFO and LIFO collections

-> ArrayDeque, LinkedList, and PriorityQueue implementing these interfaces

-> Demonstrating methods for adding, retrieving, and removing elements in queues and deques

-> Cursors in Collections

-> Iterator interface

-> hasNext( ) method and next( ) methods

-> ListIterator interface for bi-directional traversing

-> hasPrevious( ) and previous( ) methods

-> descendingIterator( ) for reverse traversal

-> forEach( ) method which accepts Consumer and BiConsumer as a parameter

-> Collections Hierarchy

-> Map Interface:

-> Map interface for key-value pair associations

-> HashMap, LinkedHashMap, and TreeMap implementing the Map interface

-> Demonstrating methods for adding, retrieving, and updating entries in a map

-> Map Hierarchy

-> SortedMap and NavigableMap Interfaces

-> SortedMap and NavigableMap interfaces for sorted maps

-> TreeMap which implements these two interfaces

-> Demonstrating methods for navigating and manipulating sorted maps

-> Collection views

-> keySet( ) method

-> values( ) method

-> entrySet( ) method

-> Utility Classes

-> Collections Class

-> Exploring the Collections class for utility methods related to collections

-> Discussing methods for sorting, searching, and modifying collections

-> Arrays Class

-> Arrays class for utility methods related to arrays

-> Discussing methods for sorting, searching, and converting arrays

-> Legacy and Thread-safe classes in collections

-> Vector

-> Stack

-> HashTable

-> Properties

-> Legacy Enumeration Interface for traversing collections

-> Performance considerations when selecting collection classes

-> Time complexity for common operations (e.g., adding, retrieving, searching) etc.

-> Comparable and Comparator Interfaces

-> Comparable Interface

-> Comparable interface for defining natural ordering of objects

-> How to implement Comparable for user-defined classes

-> Comparator Interface

-> Comparator interface for custom sorting logic

-> To implement Comparator for user-defined classes

25. Java 1.8 Features :

-> Introduction to Java 8 Features

-> Significance of Java 8 and its impact on programming

-> Major features and improvements in Java 8

-> Interface related changes

-> default methods

-> static methods

-> invoking default methods and static methods

-> Functional Interface

-> @FunctionalInterface annotations

-> Working of @FunctionalInterface wrt inheritance

-> Lambda Expressions

-> Lambda expressions as a new way to write anonymous functions

-> Syntax of lambda expressions

-> Invoking lambda expressions

-> Use of lambda expressions for functional programming

-> Pre-defined Functional Interfaces for objects

-> Predicate

-> Consumer

-> Supplier

-> Function

-> BiPredicate

-> BiConsumer

-> BiFunctions

-> Pre-defined Functional Interfaces for primitive datatype

-> IntPredicate

-> FloatPredicate

-> DoublePredicate etc.

-> Alternative to lambda expressions

-> Method References and Constructor References using double colon (: :) operator

-> Stream API to process sequences of elements present in arrays or collections

-> of( ), stream( ), filter( ), map( ) and so many methods

-> Optional class to avoid NPE

-> Optional class for handling null values

-> Explaining the concept of avoiding null pointer exceptions

-> Demonstration on how to use Optional for safer coding

-> StringJoiner class

-> Date and Time API

-> Earlier Date class

-> New Date and Time API (java.time package)

-> Exploring classes like LocalDate, LocalTime, LocalDateTime etc.

-> Period and Duration

-> SplIterator

-> Nashorn JavaScript Engine

-> Nashorn JavaScript engine for executing JavaScript on the Java Virtual Machine

Logical coding:

1. Display petterns:

square, rectangle and triangles and much more ....

almost 145+ patterns

2. Number System:

prime, fibonacci, perfect, reverse, palindrom and much more ...

almost 60+ logical questions.

3. Strings:

every type of String manupulation....

almost 100+ logical questions

4. Arrays:

every type of arrays manupulation....

almost 100+ logical questions

5. DataStructures:

More than 20 operations in the datastructures

singular, circular, double

binary search tree

6. Algorithms:

1. Recursive
2. Bubble sort
3. Insertion sort
4. Selection Sort
5. Quick sort
6. Merge sort

SQL::

Oracle uninstallation including registry clean

Downloading Oracle

Installing Oracle

Downloading SQL developer

Installing SQL developer

Configuring Connection from SQL Developer to Oracle

Play around with one simple table

Understanding the importance of Databases

Diffenet available databases in the industry

Knowing difference between database server and database

Knowing the role of SQL in the databases.

Table creation

Inserting records in various ways

Updating records in various ways

deleting records in various ways

droping table

select command

columns with aliasing

boolean conditions with where cluase

order by

built-in functions like count, min, max, avg

finding out 2nd max and 2nd min

inner query

group by

dept wise 2nd max, 2nd min

rank with over and finding any ranked record

rownum

pagination

distinct

reading unique records with distinct

reading unique records with group by

rowid

deleting duplicate records

not null

unique

constraint

composite unique

primary

composite primary

diff between unique and primary

foreign key

conditions to be followed in the foreign key

one-to-one mapping

conditions to be followed to achieve one-to-one mapping

inner join in one-to-one

left outer join in one-to-one

right outer join in one-to-one

full outer join in one-to-one

one-to-many mapping

conditions to be followed to achieve one-to-many mapping

inner join in one-to-many

left outer join in one-to-many

right outer join in one-to-many

full outer join in one-to-many

many-to-many mapping

conditions to be followed to achieve many-to-many mapping

inner join in many-to-many

left outer join in many-to-many

right outer join in many-to-many

full outer join in many-to-many

date column

timestamp column

to_date,to_timestamp and to_char functions

self join

alter table name

drop column, add new column

add constraint

disable constraint

enable constraint

drop constraints

normalization

1st normal form (1nf)

2nd normal form (2nf)

3rd normal form (3nf)

Project database design: LMS Database design

Project database design: e-commerce Database design

Project database design: LinkedIn application Database design

Restful webservices (Rest APIs) in Springboot

1. Configuring and Downloading Springboot application

   with spring web dependency
2. importing into eclipse and configuring in application.properties
3. starting and stopping the application
4. starting application in different port numbers (multiple instances).
5. knowing about embedded Tomcat
6. understanding 404
7. making application class as a RestController
8. developing helloworld restful webservice in the application class
9. CommandLineRunner
10. Developing a separate RestController
11. all diff return types to the restful webservices

    1. void
    2. primitive data types
    3. wrapper classes types
    4. array types
    5. collections type

12. understanding Bean class development
13. returning Bean type from restful webservice
14. returning Bean type with has-a from restful webservice
15. returning Bean type with collections from restful webservice
16. input through PathVariable
16. multiple PathVariables
17. understanding bad input (400 error code)
18. required, name, value attributes of PathVariable
19. input through RequestParam
20. understanding query string
21. multiple RequestParams
22. defaultValue, required, name, value attributes of RequestParam
23. RequestBody
24. Advanced Rest Client (ARC)
25. JSON object format
26. File Upload
27. Same request with multiple Files upload
28. Same request with Files upload and text parts
29. POI integration
30. Excel file upload
31. Excel file reading with POI
32. mail sender dependency
33. sending mail from restful webservice
34. ResponseEntity importance
35. ResponseEntity through static methods
36. ResponseEntity through constructors

37. ResponseEntity with diff status codes(404, 400, 500)
38. ResponseEntity with Response Headers (content-type, refresh, content-disposition)
39. File download and auto reload
40. diff types of http methods (post, get, put, delete)

Spring Data JPA:

1. Downloading Springboot application with Spring web, Data JPA and H2 dependencies
2. inmemory databases
3. Configuring H2 database
4. Entity development.
5. Repository development.
6. CRUD operations.
   1. save
   2. saveAll
   3. findById
   4. findAllById
   5. findAll
   6. count
   7. deleteById
   8. deleteAll
   9. deleteAllById
   10. delete
7. unique, not null
8. custom finder methods
   1. findByField methods
   2. findAllByField methods
   3. JQL with @query
   4. SQL with @query
9. auto generates
10. Lombok

10. composite unique key
11. composite primary key
12. service layer
13. one-to-one mapping
    1. uni-directional
    2. bi directional
       2.1. mappedBy
       2.2. jsonIgnore
       2.3. service layer
14. one-to-many mapping
    1. uni-directional
    2. bi directional
       2.1. mappedBy
       2.2. jsonIgnore
       2.3. service layer
15. many-to-many mapping
    1. uni-directional
    2. bi directional

Spring Security:

1. Security Setup
2. Inmemory Credentials and Password encoding
3. Reading Credentials from DB
4. Authorization Patterns
5. Tesing Authentication through ARC
6. Setting up Angular
7. Tesing Authorization through Angular
8. Interception Filters
9. Guard Services
10. JWT implementation in Spring Boot app and calling from Angular app

Spring Microservices:

1. Monolithic
2. Disadvantages of Monolithic
3. Microservices
4. Advantages of Microservices
5. Developing helloworld Microservice
6. Open feign
7. Eureka server
8. Feign client with Eureka server
9. Spring cloud load balancer
10. Spring cloud API Gateway
11. Fault Tolerance
12. Circuit Breaker
13. Resilience4j & Spring AOP
14. Spring cloud config server
15. Sleuth
16. Zipkin
17. Rabbit MQ
18. Docker
19. Kubernetes

HTML

1. HTML INTRODUCTION
2. HTML BASIC

Html structure

3. HTML FORMATTING

<heading>

<paragraphs>

<font>

<b>/<I>/<u>/<mark><ins><del>

<em><sub><sup>

4. HTML LIST

<ul>

<ol>

<dl>

5. HTML MARQUEE

6. HTML PRE AND HORIZONTAL RULE

7. HTML IMAGE

<src>

<alt>

<width>

<height>

8. HTML ANCHOR

Internalpage link

<target>

9. HTML TABLE

<THEAD>

<TH>

<TBODY>

<TR>,<TD>

<TFOOT>

<ROWSPAN><COLSPAN>

<BORDER>

10. HTML IFRAME

11. HTML AUDIO

<CONTROL>

<SOURCE>

12. HTML VIDEO

<CONTROL>

<SOURCE>

<WIDTH>

<HEIGHT>

<POSTER>

13. HTML FORM

ACTION

METHOD

14. HTML FORM ELEMENT

<INPUT>

<LABEL>

<SELECT>

<TEXTAREA>

<OPTION>

<BUTTON>

<FIELDSET>

<LEGEND>

15. HTML INPUT TYPES

TYPE=INPUT

TYPE=BUTTON

TYPE=CHECKBOX

TYPE=RADIO

TYPE=EMAIL

TYPE=SUBMIT

TYPE=TIME/DATE/MONTH

TYPE=TEL

TYPE=PASSWORD

16.HTML <ARTICLE>

CSS:

- CSS Introduction
- CSS Implementation
- CSS Basic Selectors
- CSS Color & Background Color
- CSS Borders
- CSS Outline
- CSS Padding
- CSS Margin
- CSS Height/Width
- CSS Min-Height/Max-Height
- CSS Min-Width/Max-Width
- CSS Box-Sizing
- CSS Overflow
- CSS Border-Radius
- CSS Box-Shadow
- CSS Float
- CSS Clear
- CSS Font
- CSS Text Formatting
- CSS Text-Decoration
- CSS Word
- CSS Text-Shadow
- CSS White-Space

- CSS Text-Overflow
- CSS Writing-Mode
- CSS Column-Count
- CSS @font-face Rule
- CSS with Google Fonts
- CSS List-Style
- CSS Background-Image
- CSS Background-Attachment
- CSS Background-Size
- CSS Background-Origin
- CSS Background-Clip
- CSS Color Modes
- CSS Gradient Background
- CSS Opacity
- CSS Background-Blend-Mode
- CSS Mix-Blend-Mode
- CSS Display
- CSS Visibility
- CSS Html Basic Layout Design
- CSS Position
- CSS Z-index
- CSS Media Queries
- CSS Table Properties
- CSS Resize
- CSS Cursor
- CSS3 Text-Decoration-Thickness
- CSS3 Text-Underline-Offset
- CSS Display Table

CSS Advanced

- CSS Units
- CSS Variables - var()

CSS Calc() function

CSS Clip-Path

CSS Shape-Outside

CSS Filter

CSS Transition

CSS Transform 2D

CSS Transform 3D

CSS Perspective

CSS Transform-Style

CSS Backface-Visibility

CSS Animation

CSS Animation-Fill-Mode

CSS Object-Fit

CSS User-Select

CSS Box-Decoration-Break

CSS Quotes

CSS Border-Image

CSS Advance (Combinator Selectors)

CSS Advanced (Attribute Selectors)

CSS Pseudo-Classes Selectors

CSS Pseudo Element Selectors

CSS Before and After Pseudo Elements

CSS Attr() function

CSS Counter

CSS Caret-Color

CSS @import Rule - Import StyleSheet

CSS with Icon Fonts

CSS Scrollbar Styling

CSS Display Flow-Root

CSS CurrentColor

CSS3 ::Marker Pseudo Element

- CSS !important
- CSS3 @supports
- CSS : Placeholder-Shown
- CSS List Style Type : String
- CSS Scroll-Behavior
- CSS Scroll-Snap
- CSS File-Selector-Button Pseudo Element
- CSS Backdrop Filter
- CSS3 :is() Pseudo Class
- CSS Text-orientation
- CSS :Focus-within Pseudo Class
- CSS Accent-Color
- CSS Prefers-color-scheme
- CSS3 Aspect-Ratio
- CSS Display-Mode @media rule
- CSS Fullscreen Pseudo class
- CSS Min() Max() Clamp() Function
- CSS Orientation Media Rule
- CSS :has Pseudo Class
- CSS Print Media
- CSS @page at-rule
- CSS Break-Before Break-After & Break-Inside
- CSS Inset
- CSS Hypens & Hypenate-Characters
- CSS Object-View-Box
- CSS Nesting

CSS Grid

- CSS Grid Layout Introduction
- CSS Grid Building with Rows & Columns
- CSS Grid-Gap
- CSS Grid Items Positioning

CSS Grid Items Spanning

CSS Grid Lines Naming

CSS Grid Area Naming

CSS Grid MinMax function

CSS Grid Implicit & Explicit Grid

CSS Grid Items Alignment

CSS Grid Tracks Alignment

CSS Grid Auto-Fill & Auto-Fit

CSS Grid Fit-Content

CSS Grid Order Property

CSS Grid - Nested Grids

CSS Grid - Overlapping Grid Items

CSS Flex

CSS Flexbox Introduction

CSS Flexbox Flex-Direction

CSS Flex-Wrap & Flex-Flow

CSS Flexbox Justify-Content

CSS Flexbox Align-Items

CSS Flexbox Align-Content

CSS Flexbox Align-Self

CSS Flexbox Order

CSS Flexbox Flex-Grow

CSS Flexbox Flex-Basis

CSS Flexbox Flex-Shrink

CSS Flexbox with Margin Auto

CSS Flexbox - Nested Flex

CSS Flexbox - Align Form Elements

Create Website Layout with CSS Flexbox

Java Script:

JS Introduction

JS Implementation

HTML Tags in Javascript

JS Comments

JS Variables

JS Variables (Let & Const)

JS Data Types

JS Arithmetic Operators

JS Assignment Operators

JS with Google Chrome Console

JS Comparison Operators

JS If Statement

JS Logical Operators

JS If Else Statement

JS If Else If Statement

JS Conditional Ternary Operator

JS Switch Case

JS Alert Box

JS Confirm Box

JS Prompt Box

JS Functions

JS Functions with Parameters

JS Functions with Return Value

JS Global & Local Variable

JS Events

JS While Loop

JS Do While Loop

JS For Loop

JS Break & Continue Statement

JS Even & Odd with Loops

JS Nested Loop

JavaScript Nested Loop - II

JS Arrays

JS Create Arrays Method - II

JS Multidimensional Arrays

JS Modify & Delete Array

JS Array Sort & Reverse

JS Array Pop & Push

JS Array Concat & Join

JS Array Slice & Splice

JS isArray

JS Array index

JS Array Includes

JS Array Some & Every

JS Array find & findIndex

JS Array Filter

JS Array Methods

JS forEach Loop

JS Objects

JS Objects - II

JS Array of Objects

JS Const Variable with Array & Objects

JS For in Loop

JS Map Method

JS String Methods

JS String Methods - II

JS Number Methods

JS Math Methods

JS Date Methods

JS Button

JS HTML DOM

JS DOM Introduction

JS DOM Targeting Methods

JS DOM Get & Set Value Methods

JS DOM querySelectors

JS DOM CSS Styling Methods

JS addEventListener Method

JS Forms

JS Form Events

JS Form Events - II

JS Effects

JS Interval

JS Timeout

JS Browser BOM

JS History Object

Angular:

1. Installation

2. Unins tallation

3. Creating a project and understanding default code

4. Creating a component

5. Creating a component with different options

6. Creating a component without ng command

7. typescript datatypes and fields declaration

8. typescript methods declaration

9. typescript methods body development

10. access specifiers and constructor

11. bindings

12. directives

13. template variable

14. template driven forms

15. reactive forms

16. http client
17. routing
18. modular development
19. tyscript compiler
20. type assertion
21. arrow functions
22. advanced directives
23. http with template variables
24. one-to-one mapping
25. one-to-many mapping
26. many-to-many mapping

GIT:

Version control

Introduction

Uses

Need

Types

Examples

Realtime why when how we use.

Difference between types of version control

Introduction to git

Features of git/ why git/ benefits of git

Diagrammatic explanation of git

Introduction to git hub

Features of git hub/ why github/ benefits of github

Branching statagy in realtime environment

Instalation of git, git hub, tortoise git

Introduction to repository

Creating repository in github

Basic commands of git

How to execute commands in git

When to use which command

Flow of realtime work life with commands

Detail discussion on daily using commands

Conflicts in git

When we will get

How to avoid

How to resolve

Raising pull request

Review comments and how to reply or address

How to merge codes

Create branches

Switch branches

Full understanding of realtime work flow with commands

Flow of jira interems of git

Realtime use of github and git.

How to practice to make familiar with these tools.

Internship:

LMS Proposal

1. Course Package CRUD operations (Like Java full stack)
2. Skill CRUD operations (Like Core Java)
3. Topic CRUD operations (OOPs concepts)
4. Subtopic CRUD Operations (Like Encapsulation)
5. Skills Mapping to Packages
6. Video Upload
7. Material Upload
8. MCQ Questions Upload
9. Students Registration (Signup and detailed form submission, view my profile, change my profile)

10. General admin activities

   1. Login
   2. Forgotten Password
   3. Remember me
   4. Change Password
   5. Logout

11. Students mapping to packages

12. Student Dashboard

   1. My Courses
   2. My Notifications
   3. My Unit exams
   4. My Cumulative exams
   5. My Certificates

13. Payment gate way integration.
14. Video player integration
15. Exams/Tests Platform

Technologies :

Server side: Springboot with Data JPA and Spring Security and Backend as MySQL

Client Side : Angular