Java Developer Interview Questions & Answers

1. Explain 4 pillers in Oops

   ➔ Oops stands for object-oriented programming. In oops everything comes under class and objects.

   There are 4 pillers-

   A] Abstraction

   B] Encapsulation

   C] Polymorphism

   D] Inheritance

   - **Abstraction**: Abstraction means hiding internal details and exposing the essential and relevant details to the users.

     Real world example

     For example- phone call, we don't know the internal processing.
     But we know how to make call and communicate with friends.
     In java, we use abstract class and interface to achieve abstraction.

   - **Encapsulation**: The wrapping up of a data into a single unit. Encapsulation refers to combining data and associated functions as a single unit. In OOP, data and functions operating on that data are combined together to form a single unit, which is referred to as a class.

     Real world example

     Capsule, it is wrapped with different medicines.
     A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private.
     declare class variables/attributes as private
     provide public getter and setter methods to access and update the value of a private variable

   - **Polymorphism:** Polymorphism means the ability to take more than one form. Polymorphism is Greek word; Poly means many, and morphism means form- many form. And form means function, many functions it means many methods.

Polymorphism uses methods to perform different tasks. This allows us to perform a single action in different ways.

2 types of Polymorphism:

A] Compile time[Method Overload]

I. Method overload means same method name, but different parameters.
II. Method overload possible with in single class
III. Static method allows to overload
IV. Final method allows to overload

Both methods must have different arguments
The number of arguments is different
The sequence of arguments is different
Type of argument is different

The Java Compiler handles Compile-time Polymorphism.

B] Run time[Method Override]

i. Method override means same method name and same parameters.
ii. Method override never possible with in single class, we must need to use Inheritance concept.
iii. Static method cant allows to override
iv. Final method cant allows to override

Both methods must have the same arguments
The number of arguments is the same
The sequence of arguments is the same
Type of argument is the same

JVM handles the Runtime polymorphism.

- Inheritance- One class can acquire the properties of another class. Inheritance promotes the reusability of code. If you want to create a new class and that class has some fields or code common to

another class, you can use that class as the parent class and can inherit that class. So in this way, you don't have to write that same piece of code again and can use that code in the child class or class which you are creating. Hence that code is reused.

subclass (child) - the class that inherits from another class
superclass (parent) - the class being inherited from

To inherit from a class, use the **extends** keyword.

In Oops there are 5 types of Inheritance-

1) Single Inheritance
2) Multilevel Inheritance
3) Multiple Inheritance
4) Hybrid Inheritance
5) Hierarchical Inheritance

Java does not support to Multiple and Hybrid Inheritance.

Because Ambiguity problem, we can't extends 2 class at a same time. Instead of multiple inheritance Java support to interfaces and by using interface will implements 2 or more interfaces with in single class by "," separated.

Benefits of Inheritance-

- Code reusability
- We can achieve Runtime Polymorphism using Inheritance.

2. What are access modifiers in Java?
   ➔ Access modifiers it means scope of visibility
   In Java 4 types of access modifiers-

   1) Private- Scope with in class
   2) Default- Scope with in Package
   3) Protected- Scope with in Package and outside package but with in subclass
   4) Public- Access everywhere

Access modifiers have a significant role in the security of the Java code. It can restrict the visibility of specific codes to certain types of users and give access to certain types of users.

3. Differentiate Abstract Class and Interface

➔ **Interface:**

i. All methods are abstract till jdk 7. But from JDK 8 interface supports to static and default methods and both are non-abstract methods.
ii. In interface all methods are public
iii. All variables are public, final & static
iv. Interface does not support to constructor
v. Interface can't instantiate
vi. Keyword- interface
vii. Interface - Interface=> extends
viii. With in class will implements n no of interfaces by "," separated

Abstract Class:

i. Abstract class support to abstract & non-abstract methods
ii. In abstract class methods are may/not be public, final & static
iii. In abstract class variables are may/not be public, final & static
iv. Abstract class support to constructor
v. Abstract class can't instantiate
vi. Keyword- abstract
vii. Interface- Class=> implements | Class- Class => extends
viii. With in class will extends only one class at a time

4. What is static block?

➔ Static blocks are executed before the main() method.
Static blocks executed automatically when the class is loaded in memory. A class can have any number of static initialization blocks.
The execution of multiple static blocks will be in the same sequence as written in the program.
Note: If you want to execute code before main method then such code/functionality you can add in static block.
Static blocks are mainly used to initialize the static variables.

5. Explain Constructors in Java

➔ Constructor itself initialize at the time of object creation

In Java Constructor have 2 types-

1) Default Constructor- No Parameter
   In this Constructor we are not passing any parameters
2) Parameterized Constructor- Passing Parameters
   In this Constructor we are going to pass some parameters

Note:

i. Constructor name same name as class name
ii. Constructor does not have any return type
iii. Constructor Overload possible in Java
iv. Constructor Override never possible in Java
v. Java does not support to copy constructor, instead of copy constructor Java support to object cloning concept.

6. What is Enum in Java?

➔ In Java, an enum (short for enumeration) is a type that has a fixed set of constant values. We use the enum keyword to declare enums. It was introduced with the release of Java 5.

Enum represents a group of constants (unchangeable variables, like final variables).

To create an enum, use the enum keyword and separate the constants with a “,” separated.

Note- Enum is by default public, static & final- Will always write enum in capital letters.

Example-

```
enum Size {
  SMALL, MEDIUM, LARGE, EXTRALARGE
}
```

7. What is Type Casting in Java?

➔ Type casting means will convert one datatype type into another datatype

In Java there is 2 types of Type Casting-

i. UpCasting[Widening]- lower to higher. Ex- int to double
ii. DownCasting[Narrowing]- higher to lower. Ex- double to int

8. Explain SOLID Principles

➔ SOLID principles are an object-oriented approach to software design & development that is used in Java.

1. Single Responsibility
2. Open/Closed
3. Liskov Substitution
4. Interface Segregation
5. Dependency Inversion

Single Responsibility Principle-

One class should have one and only one responsibility

Open Closed Principle-

Software components should be open for extension, but closed for modification

Liskov's Substitution Principle-

Derived types must be completely substitutable for their base types

Interface Segregation Principle-

Clients should not be forced to implement unnecessary methods which they will not use

Dependency Inversion Principle-

Depend on abstractions, not on concretions

Benefits-

Clean: SOLID principles make code clean and standard code.

Maintainable: with the help of SOLID principles our code becomes more manageable and easier to maintain.

Scalable: Easy to refactor or change code.

Redundancy: SOLID principles avoid redundant code.
Testable: can be easily unit tested.
Readable: SOLID principles make the code easy and readable.
Independent: code becomes independent by reducing dependencies.
Reusable: code becomes reusable.

9. What is Marker Interface and How many marker interface available in Java
   ➔ An interface that does not have any methods, i.e, an empty interface in java is known as Marker Interface
   In Java 3 Marker Interfaces-
   i. Serializable-
      The Serializable interface coming from java.io package
      It is used for Convert object into byte stream

   ii. Cloneable-
      The Cloneable interface coming from java.lang package
      It is used for Clone the object

   iii. Remote-
      The Remote interface coming from java.rmi package
      It is used for Network Programming

      Uses of Marker Interface-
      The main use of the Marker Interface in Java is to convey to the JVM that the class implementing some interface of this category has to be granted some special behavior. e.g, when a class implements the Serializable interface, which is a marker interface, then this is an indication to the JVM that the objects of this class can be serialized. Similarly, when a class implements Cloneable Interface, then it indicates to the JVM that the objects of this class can be cloned.
      When a class implements Remote Interface, then it indicates to the JVM there should be use for Network- Socket Programming.

10. What is Exception and How to handle it?
    ➔ Exception is an abnormal condition which is interrupt normal flow of program.

Exception is a class coming from java.lang package, And Throwable is a immediate parent class of Exception.

In Java 2 types of Exception

A. Compile Time[Checked] Exception-
   Those exceptions that are checked at compile-time comprises checked exceptions.
   The program will not compile if they are not handled.
   They are child classes of Exception except for RuntimeException.
   Example: IOException, SQLException, etc.
B. Run Time(Unchecked) Exception-
   Those exceptions that are checked at runtime comprises unchecked exceptions.
   They give runtime errors if not handled explicitly.
   They are child classes of RuntimeException.
   Example: ArithmeticException, NullPointerException, etc.

   There are 5 blocks for Exception Handling-

   I. Try- Exceptional code will add in try block
   II. Catch- It is used for catch the Exception
   III. Throw- It is user defined Exception
   IV. Throws- It is System generated Exception
   V. Finally- It always execute

11. Differentiate throw and throws

Throw-

1. It is used for Custom/User Defined Exception
2. Throw will declare inside method
3. In throw will handle only one Exception at a time
4. Will call throw explicitly

Throws-

1. It is System Generated Exception
2. Throws will declare at the time of method declaration
3. In throws will handle multiple Exception by “,” separated
4. Throws calls implicitly

12. Differentiate final, finally and finalize

➔ Final-

Final Variable: Can't reinitialize
Final Method: Can't override
Final Class: Can't extends

Finally-
Finally is the block. It always executes.
Realtime Use- Will add Database Connection close, Hibernate Session close code in finally block.

Finalize-
finalize() is a method it help us to achieve garbage collection in Java.

Java does not supports to destructor, In Java by default System.gc() method available.

By referencing object as null and call to System.gc();
It will help us to perform clean up.

13. What is try-with-resources?

➔ try-with-resources introduced in Java 7.
It help us to close object automatically. And we are able to use here try without catch and finally.

14. Why String class is Immutable?

➔ The five advantages String class in Java being immutable are security, caching, synchronization, class loading and performance.
Security: parameters are typically represented as String in network connections, database connection urls, usernames/passwords etc. If it were mutable, these parameters could be easily changed.

Caching: when compiler optimizes your String objects, it sees that if two objects have same value (a="Java", and b="Java") and thus you need only one string object (for both a and b, these two will point to the same object).

Synchronization and concurrency: making String immutable automatically makes them thread safe thereby solving the synchronization issues.

Class loading: String is used as arguments for class loading. If mutable, it could result in wrong class being loaded (because mutable objects change their state).

Performance: The hashcode of string is frequently used in Java. For example, in a HashMap. Being immutable guarantees that hashcode will always the same, so that it can be cached without worrying the changes.That means, there is no need to calculate hashcode every time it is used.

15. Differentiate String, StringBuffer and StringBuilder

➔ String:

I. String is Immutable
II. Comes in JDK 1.0 version

StringBuffer:

I. StringBuffer is mutable
II. Thread Safe
III. Thread Synchronized
IV. Comes in JDK 1.2 version
V. Performance wise slower than StringBuilder

String Builder:

I. StringBuilder is mutable
II. Not Thread Safe
III. Not Thread Synchronized
IV. Comes in JDK 1.5 version
V. Performance wise faster than StringBuffer

16. What is String Pool?

➔ String pool is a storage space. All string literals/variables are stored in String Pool. It is also known as String Constant Pool or String Intern Pool.
It is privately maintained by the Java String class.
By default, the String pool is empty.

17. Differentiate == and .equals()

➔ == and equals()

==

1. This operator is used for comparing addresses (or references), i.e checks if both the objects are pointing to the same memory location.
2. It is a binary operator in Java.

Equals()-

1. It will compare actual values
2. This is a method defined in the Object class.

18. Write a java program to find repeated character with number of occurrences in String

➔

```
import java.util.Arrays;

public class RepeatedCharacterWithNumberOfOccurences {

    public static void main(String[] args) {
        String name = "Swara";

        char str[] = name.toLowerCase().toCharArray();

        Arrays.sort(str);
        for (int i = 0; i < str.length; i++) {
            int count = 1;
            char temp = str[i];

            for (int j = i + 1; j < str.length; j++) {
                if (str[i] == str[j]) {
                    i++;
```

```
                        count++;
                    }
                }
                if (count > 1) {
                    System.out.println(temp + " : " + count);
                }

            }

        }

    }
```

Output: a : 2

19. How to make custom immutable class in Java?

➔ Immutable means we can't modify it

Steps to create Custom Immutable class in Java-

I. Declare class with final
II. All variables are private
III. Don't use setter, use Constructor
IV. Make all mutable fields final so that its value can be assigned only once.

20. Differentiate Array List and Linked List

➔ Both are the classes under java.util package comes under List interface. And List allows to duplication of data.

ArrayList:

I. ArrayList follows DDL- Data Definition Language
II. ArrayList is more suitable for read data
III. ArrayList follows insertion order
IV. ArrayList RealTime Example- In Google Pay Check Account Balance, will prefers to use ArrayList

LinkedList:

I. LinkedList follows DML- Data Manipulation Language
II. LinkedList is more suitable for Insert/Update/Delete data
III. LinkedList follows insertion order
IV. LinkedList RealTime Example- In Google Pay Transfer Funds(Send Money), will prefers to use LinkedList

21. When ConcurrentModificationException come?

➔ In iteration(for each, Iterator) if there is any modification happens like add, remove then ConcurrentModificationException comes at Run Time.

22. What is Fail fast and Fail safe iterator in Java?

➔ Iterator in Java is used to traverse over a collection's objects. The collections return two types of iterators, either it will be Fail Fast or Fail Safe.

Fail Fast: It means in iteration(for each/Iterator) if we are doing add/remove operations then ConcurrentModificationException occurs.

Once Exception occurs it will stopping the whole operation.

These types of iterators do not allow modifying the collection while iterating over it.

No extra memory is required in this case.

Fail Safe: It avoid the Exception- ie. ConcurrentModificationException.

A fail-safe iterator does not throw any exceptions, if the collection is modified during the iteration process.

These types of iterators allow modifying the collection while iterating over it.

Extra memory is required in this case.

23. How Hash Map internally works?

➔ HashMap is a class comes under Map interface.

Map is an interface coming from java.util package.

In Map no duplicate key but it allows duplicate values

In Map there are 2 classes

a) HashMap- Does not maintain order
   HashMap allows one null key, and multiple null values
b) TreeMap- Follows order
   TreeMap does not allow any null key, but it allows multiple null values.

Internal Working of HashMap-

In HashMap there is bucket and its size is 16 block- 0 to 15

Buckets: Array of the node is called buckets. Each node has a data structure like a LinkedList. More than one node can share the same bucket. It may be different in capacity.

HashMap uses Hashing Principle.

HashMap internally use LinkedList data structure for storing Key and Value

There are 3 methods- equals(), hashcode() and collision()

equals()- If two hashmap objects are equals then it must be same hascode

hashcode()- If two hashcode are same then collision will be arrived

collision()- Once collision arrived it will allocate the same block to store elements.

For adding elements in HashMap put() method available

24. What is LinkedHashMap?

➔ The LinkedHashMap interface extends the HashMap class to store its entries in a hash table. It internally maintains a doubly-linked list among all of its entries to order its entries.

- LinkedHashMap follows insertion order.
- LinkedHashMap is a Key, Value pair

- LinkedHashMap allows one null key and multiple null values.
- LinkedHashMap is not synchronized.
- The initial default capacity of Java HashMap class is 16 with a load factor of 0.75

25. Differentiate HashMap and HashTable

➔ HashMap-

I. HashMap allows one null key and multiple null values.
II. HashMap is not Thread Safe and not Thread synchronized.
III. HashMap is fast.
IV. Iterator in HashMap is fail-fast.
V. HashMap is a new class introduced in JDK 1.2

HashTable-

I. Hashtable does not allow any null key or value.
II. Hashtable is Thread Safe and Thread Synchronized
III. Hashtable is slow
IV. Enumerator in Hashtable is not fail-fast.
V. Hashtable is a legacy class

26. Differentiate Comparable and Comparator

➔ Both are interface in Java

Comparable:

I. Comparable coming from java.lang package
II. Comparable have only one method- compareTo()
III. Comparable it is used for single sequence sort
IV. The Collections.sort(List) method can be used to sort Comparable type list members.

Comparator:

I. Comparator coming from java.util package
II. Comparator have two methods- compare() and equals(), but equals() is not mandatory to override.
III. Comparator is used for multiple sequence sort
IV. The Collections.sort(List, Comparator) method can be used to sort the list components of the Comparator type.

27. Explain Collection Framework in detail

➔ Collection itself interface and Collections is a class in Java
Collection means group of object
In Collection Framework 3 important interface-

I. List- It allows duplication of data and follows insertion order
In list there are 2 classes-
1] ArrayList- Follows DDL(Data Definition Language)
ArrayList more suitable for Read data
Example- In Google Pay check Account Balance

2] LinkedList- Follows DML(Data Manipulation Language)
LinkedList more suitable for Insert/Update/Delete
Example- In Google Pay transfer/send money

II. Set- Uniqueness, does not allows duplication of data
In set there are 2 classes
1] HashSet- HashSet does not maintain order
2] TreeSet- Follows order

III. Map- It is Key, Value pair like JSON. Key never allows duplicate, but values allows as duplicate
In Map there are 2 classes
1] HashMap- HashMap does not maintain order
HashMap allows one null key, and allows multiple null values
2] TreeMap- Follows order
TreeMap doesn not allows any null key, and allows multiple null values.

28. Differentiate between ArrayList and Vector.

➔ **ArrayList**-

- ArrayList is not a legacy class
- ArrayList is Non-synchronized
- Increases size by 1/2 of the ArrayList
- ArrayList is not thread-safe

Vector-

- Vector is a legacy class
- Vector is Synchronized
- Increases size by double of the ArrayList
- Vector is thread-safe

29. What is Thread and Explain Thread Lifecycle

➔ Thread- It is a Lightweight part of process.
Thread is a class coming from java.lang package

We are able to create Thread in 2 ways-
A] extends Thread class
B] implements Runnable interface

There are 5 states- New, Ready, Running, Waiting & Terminate

Initially Thread is in New state then it goes Ready for execution. Execution it means in running state. In running state if any interrupt occurs then Thread goes to waiting state. Once interrupt fullfill then it goes from waiting to Ready for Execution. Once execution done finally goes to Terminate/Dead state.

1. New: In this state, a Thread class object is created using a new operator, but the thread is not alive. Thread doesn't start until we call the start() method.
2. Ready: In this state, the thread is ready to run after calling the start() method. However, the thread is not yet selected by the thread scheduler.
3. Running: In this state, the thread scheduler picks the thread from the ready state, and the thread is running.
4. Waiting/Blocked: In this state, a thread is not running but still alive, or it is waiting for the other thread to finish.
5. Dead/Terminated: A thread is in terminated, halt or dead state when the run() method exits.

30. Can we start same Thread Twice?
    ➔ No. It will give us exception- IllegalThreadStateException

31. What is Thread Synchronization?
    ➔ Two or more Thread simultaneously comes into execution but only one Thread execute at a time that's called Thread Synchronization
    Why we use Synchronization-
    Synchronization helps in preventing thread interference.
    Synchronization helps to prevent concurrency problems.

    Thread Synchronization can be achieved in the following ways.

    - Synchronized Method
    - Synchronized block
    - Static Synchronization

Lock Concept in Java-

Synchronization Mechanism developed by using the synchronized keyword in java language. It is built on top of the locking mechanism, this locking mechanism is taken care of by Java Virtual Machine (JVM). The synchronized keyword is only applicable for methods and blocks, it can't apply to classes and variables. Synchronized keyword in java creates a block of code is known as a critical section. To enter into the critical section thread needs to obtain the corresponding object's lock.

32. Differentiate wait() and sleep() method
    ➔ Wait()-
    1. Wait() method used for Interthread communication
    2. It is defined in the object class
    3. Get wait Thread in execution state then need to call notify or notifyAll() method.
    4. The wait() method releases the lock.

    Sleep()-

1. Sleep() method used for stops the execution of the current thread for some specified period.
2. It is defined in thread class
3. After some time interval Thread will comes into execution state
4. The sleep() method doesn't release the lock.

33. What's the use of the join() method?

➔ join() method is used to pause the execution of a current thread unless and until the specified thread on which join is called is dead or completed.

To stop a thread from running until another thread gets ended, this method can be used. It joins the start of a thread execution to the end of another thread's execution. It is considered the final method of a thread class.

34. What is inter-thread communication?

➔ Inter-thread communication is a process or mechanism using which multiple threads can communicate with each other.

It is especially used to avoid thread polling in java.

Using wait(), notify(), and notifyAll() methods we are able to achieve inter-thread communication.

35. What is Serialization?

➔ Serialization means convert object into byte stream.

Transient variable avoids to serialize

Serialization is used in this case which translates Java object's state to byte-stream to send it over the network or save it in file.

Serialization in java can be implemented using java.io.Serializable interface

Serializable itself Marker Interface there is no methods

36. What are the different ways to create object in Java?

➔ Object Createion Ways-

1. Using new keyword → constructor get called

Employee emp1 = new Employee();

2. Using newInstance() method of Class → constructor get called

```
Employee emp2 = (Employee) Class.forName("com.csi.Employee")
                          .newInstance();
```

It can also be written as

```
Employee emp2 = Employee.class.newInstance();
```

3. Using newInstance() method of Constructor → constructor get called

```
Constructor<Employee> constructor = Employee.class.getConstructor();
Employee emp3 = constructor.newInstance();
```

4. Using clone() method → no constructor call

```
Employee emp4 = (Employee) emp3.clone();
```

5. Using deserialization → no constructor call

```
ObjectInputStream in = new ObjectInputStream(new FileInputStream("data.obj"));
Employee emp5 = (Employee) in.readObject();
```

37. Differentiate NoClassDefFoundError & ClassNotFoundException

➔ ClassNotFoundException is an exception that occurs when you try to load a class at run time using Class.forName() or loadClass() methods and mentioned classes are not found in the classpath.

NoClassDefFoundError is an error that occurs when a particular class is present at compile time, but was missing at run time**.**

ClassNotFoundException-

1. It is an exception. It is of type java.lang.Exception.
2. It occurs when an application tries to load a class at run time which is not updated in the classpath.
3. It is thrown by the application itself. It is thrown by the methods like Class.forName(), loadClass() and findSystemClass().

4. It occurs when classpath is not updated with required JAR files.

NoClassDefFoundError -

1. It is an error. It is of type java.lang.Error.
2. It occurs when java runtime system doesn't find a class definition, which is present at compile time, but missing at run time.
3. It is thrown by the Java Runtime System.
4. It occurs when required class definition is missing at runtime.

38. List out JDK 1.8 Features

➔ JDK Features:

1. Lambada Expression- A function that can be shared or referred to as an object.
2. Functional Interface- It have Single Abstract Method
3. Static Method in Interface- It allows implementation with in interface
4. Default Method in Interface- It allows implementation with in interface
5. Date Time API- Local Date & LocalDateTime
6. For Each- Iterate records and print with in single line
7. Stream API- It is used with collection framework to sort, filter, collect data, etc
8. Collectors- It help us to count the records
9. String Joiner- It will help us to add delimiter
10. Parallel Sort- It help us to sort integer type of data
11. Optional Class- It help us to avoid null pointer exception
12. Method Reference- Uses function as a parameter to invoke a method.

39. What is Lambda Expression?

➔ Lambda Expression is an anonymous function which accepts a set of input parameters and returns results. Lambda Expression is a block of code without any name, with or without parameters and with or without results. This block of code is executed on demand.

A Lambda Expression contains 3 parts:

Parameter List A Lambda Expression can contain zero or one or more parameters. It is optional.- Lambda Arrow Operator "->" is known as Lambda Arrow operator. It separates the parameters list and body.- Lambda Expression Body

40. Write a Java program and fetch record as those employees have salary equals or more than 50000.00 using stream API

➔

```
employees.stream().filter(emp -> emp.getEmpSalary() >= 50000.00).forEach(System.out::println);
```

41. Sort Record by Name using Stream API

➔

```
employees.stream().sorted((e1,e2)-> e1.getEmpName().compareTo(e2.getEmpName())).forEach( System.out::println);
```

42. Sort Record by Age using Stream API

➔

```
employees.stream().sorted(Comparator.comparingInt( Employee::getEmpAge)).forEach(System.out::println);
```

43. Sort Record by Salary using Stream API

➔

```
employees.stream().sorted(Comparator.comparingDouble( Employee::getEmpSalary)).forEach(System.out::println);
```

44. What is a Stream API?

➔ The Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

We can use Stream to filter, collect, print, and convert from one data structure to another, etc.

Stream does not store elements. It simply conveys elements from a source such as a data structure, an array, or an I/O channel, through a pipeline of computational operations.

Stream is functional in nature. Operations performed on a stream do not modify its source. For example, filtering a Stream obtained from a collection produces a new Stream without the filtered elements, rather than removing elements from the source collection.

45. What is Functional interfaces?

➔ Functional Interfaces are an interface with only one abstract method. Due to which it is also known as the Single Abstract Method (SAM) interface. It is known as a functional interface because it wraps a function as an interface or in other words a function is represented by a single abstract method of the interface.
Functional interfaces can have any number of default, static, and overridden methods. For declaring Functional Interfaces @FunctionalInterface annotation is optional to use. If this annotation is used for interfaces with more than one abstract method, it will generate a compiler error.
Functional interfaces from previous Java versions are Runnable, Callable, Comparator, and Comparable.

46. What are the features of a lambda expression?

➔ Less Coding and enable functional programming
Lambda expressions can be passed as a parameter to another method.
Lambda expressions can be standalone without belonging to any class.

47.What is an Optional class?

➔ It help us to avoid NullPointerException

Optional is a container type which may or may not contain value i.e. zero(null) or one(not-null) value. It is part of java.util package. There are pre-defined methods like isPresent(), which returns true if the value is present or else false.

48. Differentiate map() and flatMap()

➔ Map()-

1. It processes stream of values.
2. The function passed to map() operation returns a single value for a single input.
3. One-to-one mapping occurs.
4. It only performs the mapping.
5. map() is used only for transformation.

flatMap()-

1. It processes stream of stream of values.
2. The function you pass to flatmap() operation returns an arbitrary number of values (zero or more) as output for a single value.
3. One to many mapping occurs.
4. It performs mapping as well as flattening.
5. flatMap() is used both for transformation and mapping.

49. Write Java code for Summation using Stream API

➔

```
public class SummationNumber {
    public static void main(String[] args) {

            List<Integer> intList = Arrays.asList(8, 9, 1, 10);

            Integer sum =
intList.stream().collect(Collectors.summingInt(Integer::intValue));

            System.out.println(sum);

//or
            /*
            Integer sumation = intList.stream()
```

```
                .reduce(0, Integer::sum);

        System.out.println(sumation);*/

    }
}
```

50. Write a Java Code for HashMap sort by Key/Value-

➔

```
Map<String, Integer> hm = new HashMap<>();
        hm.put("IT", 1);
        hm.put("ELECTRONICS", 5);
        hm.put("MECHANICAL", 3);
        hm.put("COMPUTER", 2);

        Map<String, Integer> finalSortbyValue =
hm.entrySet().stream().sorted(Map.Entry.comparingByKey())
                .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue, (v1, v2) -> v2, LinkedHashMap::new));

        finalSortbyValue.forEach((key, value) ->
System.out.println(key + "\t : " + value));
```

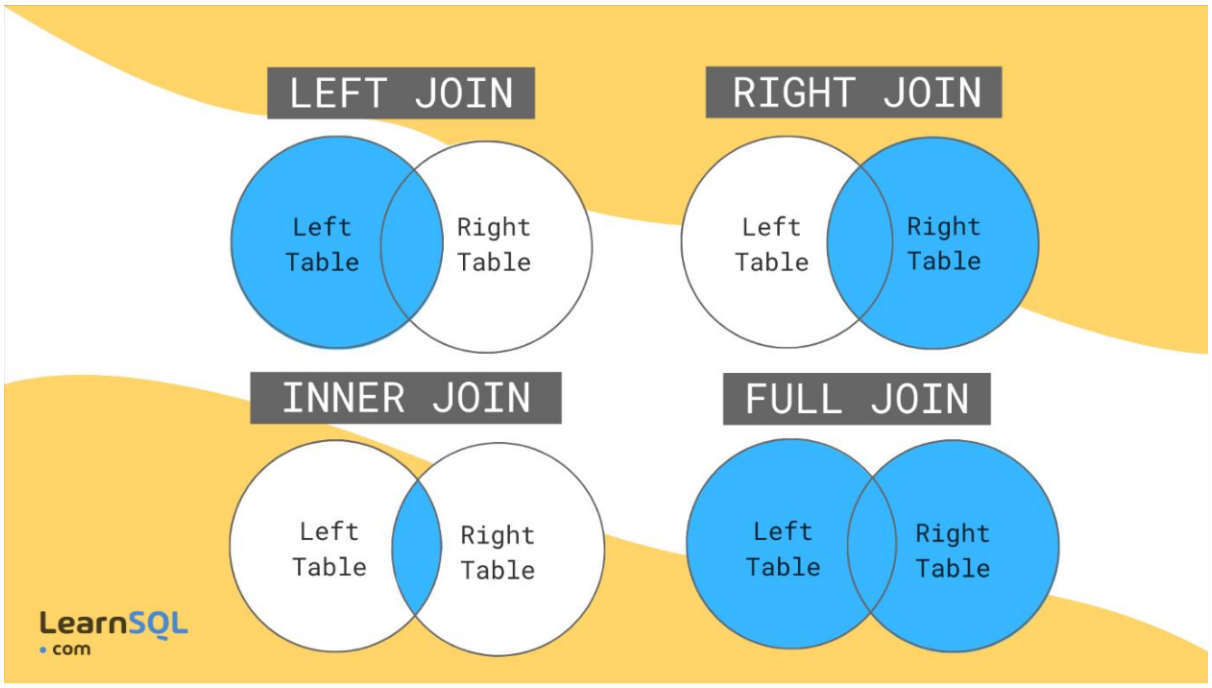
51. Explain Joins in SQL

➔ The SQL Join clause is used to combine records (rows) from two or more tables

In SQL 4 important Joins

1. Inner Join- Common data from both table
2. Left Outer Join- All Data from Left and common data from Right table
3. Right Outer Join- All data from Right table and common data from Left table

4. Full Outer Join- All Data from both table, But MySQL does not support to Full Outer Join. In MySQL we are using Left Outer & Right Outer Union to achieve Full Outer Join.



52. How many Aggregate function in SQL?

➔ In SQL 5 aggregate functions
MIN, MAX, AVG, SUM & COUNT

53. Write a SQL query to find second largest salary

➔

SELECT MAX(empsalary) FROM employee WHERE
empsalary NOT IN (SELECT MAX(empsalary) FROM employee)

54. Differentiate Delete and Truncate

➔ Delete & Truncate
Delete:

1. The DELETE command deletes one or more existing records from the table in the database.

2. We can restore any deleted row or multiple rows from the database using the ROLLBACK command.
3. The DELETE command performs slower
4. The DELETE command is Data Manipulation Language Command.

Truncate:

1. The TRUNCATE Command deletes all the rows from the existing table, leaving the row with the column names.
2. We cannot restore all the deleted rows from the database using the ROLLBACK command.
3. The TRUNCATE command works faster
4. The TRUNCATE command is a Data Definition Language Command.

55. What is Normalization?

➔ Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

56. What is a Stored Procedure?

➔ A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

57.What are ACID properties?

➔ ACID stands for Atomicity, Consistency, Isolation, Durability. They are database transaction properties which are used for guaranteeing data validity in case of errors and failures.

Atomicity: This property ensures that the transaction is completed in all-or-nothing way.
Consistency: This ensures that updates made to the database is valid and follows rules and restrictions.
Isolation: This property ensures integrity of transaction that are visible to all other transactions.
Durability: This property ensures that the committed transactions are stored permanently in the database

58. How do you add a column to an existing table?
➔ ALTER TABLE employee ADD (empaddress, varchar(255));

59. What is Cursor? How to use a Cursor?
➔ A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

60. What is a View?
➔ A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

61. What is Hibernate?
➔ Hibernate is an ORM tool[Object Relational Mapping].
Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight ORM tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.
Why use Hibernate?
Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

62. What are features in Hibernate?

➔ Features:

1. Database Migration Easy
2. Hibernate will take care of creating table
3. HQL[Hibernate Query Language]- Database Independent
4. Fast Performance
5. Open Source and Lightweight

63. What is a Session in Hibernate?

➔ A session is an Interface that maintains the connection between Java object application and database. Session also has methods for storing, retrieving, modifying or deleting data from database using methods like persist(), load(), get(), update(), delete(), etc. Additionally, It has factory methods to return Query, Criteria, and Transaction objects.

64. What is a SessionFactory?

➔ SessionFactory provides an instance of Session. It is a factory class that gives the Session objects based on the configuration parameters in order to establish the connection to the database.
With in application there will be single instance of SessionFactory.

65. Differentiate first level cache and second level cache in Hibernate

➔ First Level Cache:

1. This cache is enabled by default and there is no way to disable it.
2. For each transaction we creating new Session
3. The first level cache is available only until the session is open, once the session is closed, the first level cache is destroyed.

Second Level Cache:

1. This is disabled by default, but we can enable it through configuration.
2. For whole application we have one SessionFactory
3. The second-level cache is available through the application's life cycle, it is only destroyed and recreated when an application is restarted.

66. Differentiate session.get() and session.load() in Hibernate

➔ Get():

1. get returns null if the object is not found.

2. get hit every time the method is called.
3. This method gets the data from the database as soon as it is called.
4. This method should be used if we are unsure about the existence of data in the database

Load():

1. Load throws ObjectNotFoundException if the object is not found.
2. Load hit only when it is really needed and this is called Lazy Loading which makes the method better.
3. Load returns a proxy object and loads the data only when it is required.
4. This method is to be used when we know for sure that the data is present in the database.

67. What is the actual use of @Entity, @Id, @GeneratedValue, @Temporal, @Transient, @Casecade annotation

➔ @Entity- Its JPA Entity. It will consider POJO class name as table name
@Id- Primary key
@GeneratedValue- Id will be Auto increment
@Temporal- we are using to declare Date with DateTime
@Transient- It will avoid persist data into db
@Casecade- It will help us to perform same operations with entity relation table

68. Explain save(), persist(), update(), saveOrUpdate(), merge()

➔

1. save()- It is used to insert the newly created object in the datastore

2. saveOrUpdate()- This method will work for both new and old objects.
If object is new, it will work like simple save or if object is already persistent, it will work like update

3. update()- Update the data

4. persist()- It is used to insert a new entity into the context and there must not
be another entity with the same ID, otherwise an exception will be thrown (EntityExistsException)

5. merge()- It is used to put entity back to persistence context if the entity was detached
and was changed.

69. What is Spring Framework & Why its popular?

➔ Spring is a powerful open-source, loosely coupled, lightweight, java framework meant for reducing the complexity of developing enterprise-level applications. This framework is also called the "framework of frameworks" as spring provides support to various other important frameworks like JSF, Hibernate, Structs, EJB, etc.

Features of Spring Framework:

1. Lightweight
2. Simple
3. Loosely Coupled
4. Testability
5. Development Faster

70. What is IOC?

➔ IOC: Inversion of Control

1. Entire ApplicationContext(container) beans that called IOC
   General Life Example- Coffee Machine

Spring container forms the core of the Spring Framework. The Spring container uses Dependency Injection (DI) for managing the application components by creating objects, wiring them together along with configuring and managing their overall life cycles. The instructions for the spring container to do the tasks can be provided either by XML configuration, Java annotations, or Java code.

71. What is DI?

➔ DI: Dependency Injestion

In ApplicationContext there is available bean id that called DI. With in ApplicationContext there will be unique bean id.

The main idea in Dependency Injection is that you don't have to create your objects but you just have to describe how they should be created.

In Java, the 2 major ways of achieving dependency injection are:

Constructor injection: Here, the IoC container invokes the class constructor with a number of arguments where each argument represents a dependency on the other class.

Setter injection: Here, the spring container calls the setter methods on the beans after invoking a no-argument static factory method or default constructor to instantiate the bean

72. differentiate constructor and setter injection DI

➔ Constructor:

1. In constructor *injection*, partial injection is not allowed
2. The constructor injection doesn't override the setter property
3. Constructor injection creates a new instance if any modification is done
4. In case the bean has many properties, then constructor injection is preferred

Setter:

1. Partial injection allowed in Setter
2. Setter override the Constructor DI
3. The creation of a new instance is not possible in setter injection
4. If few properties, then setter injection is preferred

73. How many Beans Scope available in Spring Framework?

➔ The Spring Framework has five Beans scope

1. Singleton: The scope of bean definition while using this would be a single instance per IoC container.
   By default, singleton beans scope available
2. Prototype: Each time it will create new Instance
3. Request: The scope of the bean definition is an HTTP request.
4. Session: Here, the scope of the bean definition is HTTP-session.

5. Global-session: The scope of the bean definition here is a Global HTTP session

74. How many Autowiring mode available in Spring?

➔ The IoC container autowires relationships between the application beans. Spring lets collaborators resolve which bean has to be wired automatically by inspecting the contents of the BeanFactory.

5 Autowiring mode available in Spring:

1. no: This means no autowiring and is the default setting. An explicit bean reference should be used for wiring.
2. byName: The bean dependency is injected according to the name of the bean. This matches and wires its properties with the beans defined by the same names as per the configuration.
3. byType: This injects the bean dependency based on type.
4. constructor: Here, it injects the bean dependency by calling the constructor of the class. It has a large number of parameters.
5. autodetect: First the container tries to wire using autowire by the constructor, if it isn't possible then it tries to autowire by byType

75. What is Spring Boot and List out the Features?

➔ Spring Boot is an open-source, java-based framework that provides support for Rapid Application Development and gives a platform for developing stand-alone and production-ready spring applications with a need for very few configurations

Features of Spring Boot

➔ 1. With in Minute Production Ready Application we are able to create
2. Inbuild Tomcat, Jetty, Undertow web server available
3. No XML Configuration required
4. DevTool Depedency- Not need to restart application after changes
5. Increased productivity
6. Reduced development time
7. Spring Boot also offers pinpointed ‘started’ POMs to Maven configuration
8. Spring Initializer – This is a web application that helps a developer in creating an internal project structure. The developer does not

have to manually set up the structure of the project while making use of this feature.

9. Auto-Configuration – This helps in loading the default configurations according to the project you are working on. In this way, unnecessary WAR files can be avoided.
10. Spring Actuator – Spring boot uses actuator to provide "Management EndPoints" which helps the developer in going through the Application Internals, Metrics etc.
11. Logging and Security – This ensures that all the applications made using Spring Boot are properly secured without any hassle
12. Development Faster

76. What does @SpringBootApplication annotation do internally?

➔ @SpringBootApplication annotation is one point replacement for using @Configuration, @EnableAutoConfiguration and @ComponentScan annotations alongside their default attributes.
This enables the developer to use a single annotation instead of using multiple annotations thus lessening the lines of code. However, Spring provides loosely coupled features which is why we can use these annotations as per our project needs.

77. Can we change the default port of the embedded Tomcat server in Spring boot?

➔ Yes, by using server.port=2025 in application.properties file.
Here you can set any 4 digit port number.

78. What are the uses of below annotations?

➔ **RestController -** This is applied to a class to mark it as a request handler thereby creating RESTful web services using Spring MVC. This annotation adds the @ResponseBody and @Controller annotation to the class.

@RequestMapping- This provides the routing information and informs Spring that any HTTP request matching the URL must be mapped to the respective method.

@Autowired- We can use the @Autowired to mark a dependency which Spring is going to resolve and inject. We can use this annotation with a constructor, setter, or field injection.
We are not creating object by using new keyword, will prefers here @Autowired annotation.

@Bean- @Bean is a method-level annotation and a direct analog of the XML element.
The annotation supports some of the attributes offered by, such as init-method, destroy-method, autowiring and name.

@Qualifier- This annotation helps fine-tune annotation-based autowiring. There may be scenarios when we create more than one bean of the same type and want to wire only one of them with a property. This can be controlled using @Qualifier annotation along with the @Autowired annotation

Required- The @Required annotation is method-level annotation and applied to the setter method of a bean.
This annotation simply indicates that the setter method must be configured to be dependency-injected with a value at configuration time.

@Scope- We use @Scope to define the scope of a @Component class or a @Bean definition. It can be either singleton, prototype, request, session, globalSession or some custom scope.

@Controller- This annotation is simply a specialization of the @Component class and allows
implementation classes to be autodetected through the classpath scanning.
We can define a Spring MVC controller with @Controller.

@CrossOrigin- @CrossOrigin enables cross-domain communication for the annotated request handler methods.
It help us to communicate with Front End Application

@RequestBody- @RequestBody annotation indicating a method parameter should be bound to
the body of the web request. The body of the request is passed through an HttpMessageConverter to resolve the method argument depending on the content type of the request. Optionally, automatic validation can be applied by annotating the argument with @Valid

@GetMapping- @GetMapping annotation for mapping HTTP GET requests onto specific handler methods.
Specifically, @GetMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET)

@PostMapping- @PostMapping annotation for mapping HTTP POST requests onto specific handler methods.
Specifically, @PostMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST)

@PutMapping- @PutMapping annotation for mapping HTTP PUT requests onto specific handler methods.
Specifically, @PutMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.PUT).

@PatchMapping- @PatchMapping annotation for mapping HTTP PATCH requests onto specific handler methods.
Specifically, @PatchMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.PATCH).

@DeleteMapping- @DeleteMapping annotation for mapping HTTP DELETE requests onto specific handler methods.
Specifically, @DeleteMapping is a composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.DELETE).

@ControllerAdvice- @ControllerAdvice annotation is a specialization of @Component. The classes
annotated with @ControllerAdvice are auto-detected by classpath scanning.
The use of @ControllerAdvice is advising all or selected controllers for

@ExceptionHandler, @InitBinder, and @ModelAttribute. What we have to do is create a class annotated with @ControllerAdvice and create a required method which will be annotated with @ExceptionHandler for global exception handling, @InitBinder for global init binding and @ModelAttribute for global model attributes addition.

@ResponseBody- When you use the @ResponseBody annotation on a method, Spring converts the return value and writes it to the HTTP response automatically. Each method in the Controller class must be annotated with @ResponseBody.
The @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object.

@ExceptionHandler- @ExceptionHandler annotation for handling exceptions in specific handler classes and/or handler methods.
Handler methods which are annotated with this annotation are allowed to have very flexible signatures.
Spring calls this method when a request handler method throws any of the specified exceptions. The caught exception can be passed to the method as an argument.
@ResponseStatus- We can specify the desired HTTP status of the response if we annotate a request handler method with this annotation. We can declare the status code with the code argument, or its alias, the value argument.
Also, we can provide a reason using the reason argument.

@PathVariable- This annotation indicates that a method argument is bound to a URI template variable. We can specify the URI template with the @RequestMapping annotation and bind a method argument to one of the template parts with @PathVariable.

@RequestParam- @RequestParam annotation which indicates that a method parameter should be bound to a web request parameter. We use @RequestParam for accessing HTTP request parameters.

@SpringBootApplication- @SpringBootApplication annotation indicates a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.
The @SpringBootApplication annotation it includes @Configuration, @EnableAutoConfiguration, and @ComponentScan with their default attributes.

@ComponentScan- It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components.

@Repository – JPA Repo- @Repository Annotation is a specialization of @Component annotation which is used to indicate that the class provides the mechanism for storage, retrieval, update, delete and search operation on objects. Though it is a specialization of @Component annotation, so Spring Repository classes are autodetected by spring framework through classpath scanning. This annotation is a generalpurpose stereotype annotation which very close to the DAO pattern where DAO classes are responsible for providing CRUD operations on database tables. It is a class-level annotation. The repository is a DAOs (Data Access Object) that access the database directly.

@Component- @Component is a class-level annotation. It is used to denote a class as a Component. We can use @Component across the application to mark the beans as Spring's managed components. A component is responsible for some operations. Spring framework provides three other specific annotations to be used when marking a class as a Component

@Service- It is also used at class level. It tells the Spring that class contains the business logic. Service class usually talk to the dao or data access layer to get data and combine them to required form that controller needs.

79. Differentiate @Controller and @RestController

➔ Controller-

1. Controller does not contains @ResponseBody
2. Mostly will use Controller in Spring MVC

RestController-

1. Itself contains @ResponseBody
2. In Spring Boot always will prefers to use @RestController

80. Explain Spring MVC Workflow

➔ Spring MVC-

End User requests for a page by specifying the Web URL for the page. E.g. https://amazon.in

Client request is intercepted by the Dispatcher Servlet also known as Front Controller. Dispatcher Servlet is a servlet specified in Web.XML file (for XML Based configurations) or in the Web Configuration class (for java based configurations).

Dispatcher Servlet uses URL Mapping Handler to find out the relevant controller class to which request should be passed for subsequent processing. For example, If you have a Controller defined for all requests by specifying "/" in the URL, all requests will be entertained by that controller.

Once Dispatcher Servlet has identified the Controller to be considered, it passes the client request to the controller.

The controller class is the main class controlling the business logic flow once request has been dispatched it it by dispatcher servlet. This class will implement the methods for different type of http requests (e.g. GET, POST, PUT, DELETE) and all logic to call Service layer methods will reside in this controller class.

The controller class will also be responsible for returning the ModelAndView object back to the dispatcher servlet after getting all business logic executed and any data returned from DAO layer. ModelAndView object returned by the controller back to the controller specified both view and model objects.

After receiving ModelAndView object from the controller, Dispatcher Servlet now sends model object to view resolver to get the name of the view which needs to be rendered.
Once the view to be rendered has been identified, Dispatcher Servlet passes model object to the view. Model object contains the data which needs to be displayed in the view. View will be rendered with the model data. Views can be designed in any front-end technology.
This view is returned to the client and client can see the view and associated data on his browser.

81. Differentiate @RequestParam and @PathVariable

➔ @RequestParam-

1. It is used to extract query parameters that is anything after "?" in the URL
2. Mostly RequestParam used in Spring MVC

@ PathVariable-

1. It is used to extract the data present as part of the URI itself.
2. Mostly PathVariable used in Spring Boot

82. How to create custom exception in Spring Boot

➔ Create package as exception then inside package we are going to create Custom Exception class- Ex. RecordNotFoundException then extends Exception/RunTimeException
Use Parameterized Constructor and pass String type of parameter.
Use super() keyword pass this String type parameter.
In class level use @ResponseStatus(value=HTTPSTATUS_NOTFOUND)

83.What are Profiles in Spring Boot?

➔ Profiles in the Spring framework enables users to map components and beans to specific profiles, such as the Development (DEV) profile, Production (PROD) profile, the Test(QA) profile.

In Spring Boot, the annotation @Profile is used to map components and beans to a certain profile.

Once you are done with the profile-specific configuration, you have to set the active profile in an environment. To do that, either you can

Use -Dspring.profiles.active=prod in arguments
Use spring.profiles.active=prod in application.properties file

84. Differentiate SOAP vs REST

➔ SOAP- Simple Object Access Protocol

1. It only supports XML format.
2. SOAP is more secure
3. SOAP cannot use REST because SOAP is a protocol and REST has an architectural style.
4. SOAP uses WSDL to expose supported methods and technical details
5. SOAP defines standards to be strictly followed.
6. SOAP is less preferred than REST.
7. SOAP is by default stateless

REST-

1. It supports various formats like HTML, XML and JSON.
2. REST is less secure than SOAP
3. REST can use SOAP as a protocol for web services
4. REST exposes methods through URIs, there are no technical details.
5. REST does not define too much standards like SOAP
6. REST more preferred than SOAP
7. It is stateful, i.e. no server-side sessions occur.

85. Have you written any web service from scratch?

➔ Many services written. Example- Java Mail API, Signup, Login, Upload CSV File, Update, Delete, etc.

86. Have you modify any existing web service?

➔ Yes, day to day as client business requirements doing modifications with in services. Ex. Java Mail API recently modified for Multiple Participants as TO & CC.

87. What is actuator in Spring Boot?

➔ It help us monitor systems and provides endpoint security
By default, spring security is enabled for all actuator endpoints if it available in the classpath.
Spring Boot Actuator Features-
There are 3 main features of Spring Boot Actuator:

Endpoints- The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. We can also create our own endpoint. We can enable and disable each endpoint individually.
Metrics- Spring Boot Actuator provides dimensional metrics by integrating with the micrometer. The micrometer is integrated into Spring Boot. It is the instrumentation library powering the delivery of application metrics from Spring. It provides vendor-neutral interfaces for timers, gauges, counters, distribution summaries, and long task timers with a dimensional data model.
Audit- Spring Boot provides a flexible audit framework that publishes events to an AuditEventRepository. It automatically publishes the authentication events if spring-security is in execution.

88. Give some brief about application.properties file

➔ In application.propeties file we are adding mostly db configuration.
In a Spring Boot Application, 'application.properties' is an input file to set the application up. Unlike the standard Spring framework configuration, this file is auto detected in a Spring Boot Application. It is placed inside "src/main/resources" directory. Therefore, we don't need to specially register a PropertySource, or even provide a path to a property file.

An application.properties file stores various properties in key=value format. These properties are used to provide input to Spring container object, which behaves like one time input or static data.

89. What is Data JPA & Why its more powerful?

➔ Its lightweight framework, And Data JPA uses Hibernate as internal implementation.

Spring Data JPA is part of Spring Data family. Spring Data makes it easier to create Spring driven applications that use new ways to access data, such as non-relational databases, map-reduction frameworks, cloud services, as well as well-advanced relational database support

Features-

1. Reduce code size for CRUD operations by using JPARepository
2. Support QueryDSL and JPA queries
3. Audit of domain classes
4. Support for batch loading, sorting, dynamical queries
5. We are able to create custom methods as per requirements

90. What is Maven and Explain its actual use?

➔ Maven is a Local Build Management Tool

It will help us to create new Fresh JAR/WAR File.

Maven helps the developer to create a java-based project more easily. Accessibility of new feature created or added in Maven can be easily added to a project in Maven configuration. It increases the performance of project and building process.

The main feature of Maven is that it can download the project dependency libraries automatically.

91. Which tool you have used in your last project for Code Control Management

➔ GitHub

92. What is Microservices?

➔ 2 or more Spring Boot module communicate with each other that's called Microservices.

Microservices or more appropriately Microservices Architecture is an SDLC approach based on which large applications are built as a collection of small functional modules. These functional modules are independently deployable, scalable, target specific business goals, and communicate with each other over standard protocols. Such modules can also be implemented using different programming languages, have their databases, and deployed on different software environments. Each module here is minimal and complete.

Benefits:

Self-contained, and independent deployment module.
Independently managed services.
In order to improve performance, the demand service can be deployed on multiple servers.
It is easier to test and has fewer dependencies.
A greater degree of scalability and agility.
Simplicity in debugging & maintenance.
Better communication between developers and business users.
Development teams of a smaller size.

Drawbacks:

Due to the complexity of the architecture, testing and monitoring are more difficult.
Lacks the proper corporate culture for it to work.
Pre-planning is essential.
Complex development.
Requires a cultural shift.
Expensive compared to monoliths.
Security implications.
Maintaining the network is more difficult.

93. Explain the architecture of Microservices?

➔ The main idea behind microservices is that some types of applications are easier to build and maintain when they are broken down into many small pieces that work together.

API Gateway- End user request always comes on API Gateway and API Gateway will take care of traversing this request to particular services. Basically, the API Gateway is a reverse proxy to microservices

and acts as a single-entry point into the system. It is similar to a Facade pattern from object-oriented design and similar to the notion of an “Anti-Corruption Layer” in Domain Driven Design. It makes the processes of API design, implementation, and management considerably simpler and more consistent.

The API gateway also helps by recording data for analysis and auditing purposes, load balancing, caching, and static response handling. The diagram below shows how the gateway typically fits into the overall microservice architecture.

Eureka Server- Register all services as instances in Eureka Server.

It, also referred to as Netflix Service Discovery Server, is an application that keeps track of all client-service applications. As every Microservice registers to Eureka Server, Eureka Server knows all the client applications running on the different ports and IP addresses. It generally uses Spring Cloud and is not heavy on the application development process.

Hystrix Dashboard- Monitor System Health

Netflix has provided a bunch of libraries to form an ecosystem in microservices. Like Eureka, Hystrix is also an open source library provided by Netflix in the Microservices space. Hystrix implements the Circuit Breaker pattern. You don’t have to write the network or thread programming to handle fault tolerance in the Microservices. You need to use Hystrix library just by giving parameters and that’s it.

Hystrix is going to do all the work for you internally. The best part of it is that it works amazingly with Spring Boot. If you pick any Microservice based project, there are pretty good chances that they are using Hystrix.

Zipkin- It will help us for distributed log tracing.

Zipkin is very efficient tool for distributed tracing in microservices ecosystem. Distributed tracing, in general, is latency measurement of each component in a distributed transaction where multiple microservices are invoked to serve a single business usecase.

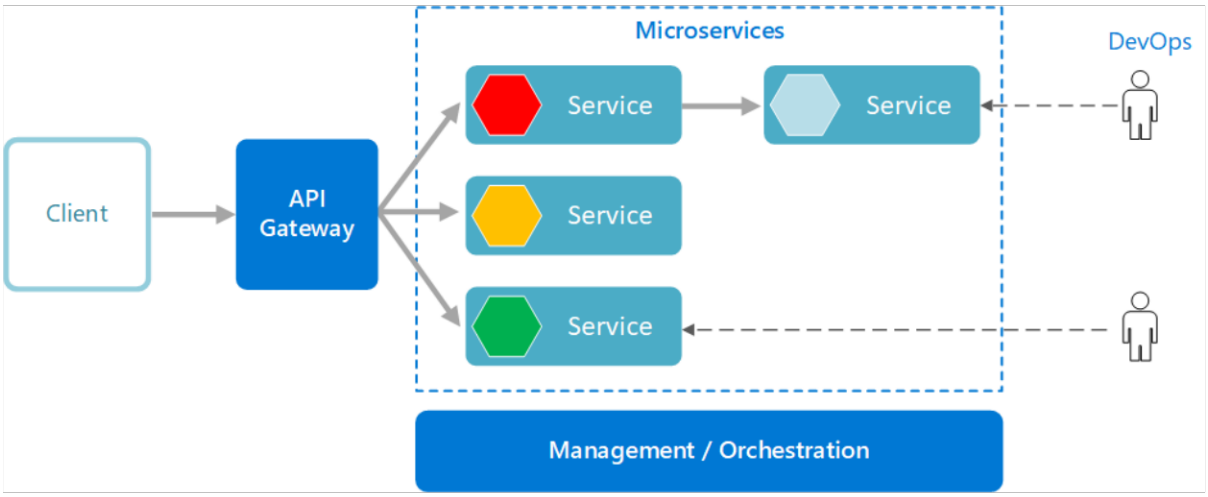
Zipkin was originally developed at Twitter, based on a concept of a Google paper that described Google’s internally-built distributed app debugger – dapper. It manages both the collection and lookup of this

data. To use Zipkin, applications are instrumented to report timing data to it.

Sleuth- It identify which service is depends with another services

Sleuth introduces unique IDs to your logging which are consistent between microservice calls which makes it possible to find how a single request travels from one microservice to the next.

Spring Cloud Sleuth adds two types of IDs to your logging, one called a trace ID and the other called a span ID. The span ID represents a basic unit of work, for example sending an HTTP request. The trace ID contains a set of span IDs, forming a tree-like structure. The trace ID will remain the same as one microservice calls the next.



Microservice Architecture

94. What is Semantic Monitoring?

➔ The semantic monitoring method, also called synthetic monitoring, uses automated tests and monitoring of the application to identify errors in business processes. This technology provides a deeper look into the transaction performance, service availability, and overall application performance to identify performance issues of microservices, catch bugs in transactions and provide an overall higher level of performance.

95. What is OAuth?

➔ Generally speaking, OAuth (Open Authorization Protocol) enables users to authenticate themselves
with third-party service providers. With this protocol, you can access client applications on HTTP for third-party providers such as Gamil, Linkedin, GitHub, Facebook, etc. Using it, you can also share resources on one site with another site without requiring their credentials.

96. What is Circuit Breaker in Microservices?

➔ In a microservice based application, Circuit Breaker is a technique, where we stop executing an erroneous method and redirect every request to a custom method (Fallback method). Generally, we stop execution of a particular method if it is continuously throwing an exception. When we break the circuit, we also avoid any cascading failures to the downstream services. Its basic function is to interrupt current flow after a fault is detected. A circuit breaker can be reset to resume normal operation either manually or automatically.

Depending on the state, Circuit Breaker changes its behavior.

A]Closed

If Client request is sent to the actual service method only, then it is called as CLOSED CIRCUIT. Hence, this state represents that the service is running properly and providing the expected functionality.

B]Open

If Client request is redirected to Fallback method, then such case is an OPEN CIRCUIT. Hence, this state represents that service is unavailable or faulty and error rate is beyond the threshold.

C]Half-Open

Once the state becomes OPEN, we wait for some time in the OPEN state. After a certain period of time, the state becomes HALF_OPEN.
During this period, we do send some requests to Service to check if we still get the proper response.
If the failure rate is below the threshold, the state would become CLOSED. If the failure rate is above the threshold, then the state becomes OPEN once again. This cycle continues till the service
becomes stable

97. Explain TDD & DDD

➔ **TDD- Test Driven Development**

First come tests and then the code. The minimal piece of code is written in order to pass the designed test. In other words, it is the process of testing the code before its accrual writing. If the code passes the test, then we can proceed to its refactoring.

DDD- Domain Driven Development

The way of creating complex systems by developing the separate parts of it. At first, the domain (a set of functionality) is defined and described as before creating something it is necessary to understand what exactly it will be. In other words, it is the process of being informed about the domain before code writing.

98. Difference between Monolithic, SOA and Microservices Architecture

➔ **Monolithic Architecture:** It is "like a big container" where all the software components of an application are bundled together tightly. It is usually built as one large system and is one code-base.

SOA (Service-Oriented Architecture): It is a group of services interacting or communicating with each other. Depending on the nature of the communication, it can be simple data exchange or it could involve several services coordinating some activity.

Microservice Architecture: It involves structuring an application in the form of a cluster of small, autonomous services modeled around a business domain. The functional modules can be deployed independently, are scalable, are aimed at achieving specific business goals, and communicate with each other over standard protocols.

99. What is Jenkins & Its usage in Industry?

➔ Jenkins is Centralize Build Management Tool

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in

the Java programming language. It is used to implement CI/CD workflows, called pipelines.
Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven 2 project, Amazon EC2.

100. What is Agile and explain various Agile Ceremony?

➔ Agile help us to give Productivity from each team mates.
The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

Agile Process have 4 Ceremony:

1. Sprint Planning- Here will discuss about sprint task estimation and development- story point efforts
   Sprint Cycle should be 2 week- 10 Days
   Purpose: Sprint planning sets up the entire team for success throughout the sprint. Coming into the meeting, the product owner will have a prioritized product backlog. They discuss each item with the development team, and the group collectively estimates the effort involved. The development team will then make a sprint

forecast outlining how much work the team can complete from the product backlog.

2. Daily Scrum- Here each team mates need to give below updates-
    a. What you did yesterday
    b. What are you doing now
    c. Which Problems you are facing

    Daily Scrum call not be more than 30 Minutes

    Purpose: Stand-up is designed to quickly inform everyone of what's going on across the team. It's not a detailed status meeting. The tone should be light and fun, but informative.

3. Backlog Refinements- Here will discuss about Last Sprint Backlog story
    Purpose: Priority Backlog will get completed using skill full resources.

4. Sprint Retrospective- It will help us to create positive energy.
    This is the right time to say thanks whoever help us to complete story in current sprint.
    Purpose: Agile is about getting rapid feedback to make the product and development culture better. Retrospectives help the team understand what worked well–and what didn't.
    Retrospectives aren't just a time for complaints without action. Use retrospectives to find out what's working so the team can continue to focus on those areas. Also, find out what's not working and use the time to find creative solutions and develop an action plan. Continuous improvement is what sustains and drives development within an agile team, and retrospectives are a key part of that.

Note- Please reach out me at contact@fullstackjavadeveloper.in

Your suggestions always welcomes

Thanks,

Full Stack Java Developer, Pune Team