

Spring Boot interview questions

1. Spring vs Spring Boot

Ans. Spring is java based framework used to built enterprise application.

Spring boot is one of module of spring which is easy to configure and allows developer to focus on business logic

2. Why spring boot or advantages of Spring Boot ?

Ans. -Less configuration

-embedded tomcat

-provides support for logging

3. few features of Spring Boot?

Ans. -Spring initializer : web application used to built internal project structure for spring boot

-Spring Actuator: used to manage monitor application ***in production*** eg. health status ,cpu usage, identification of bean, auditing.

it can be enabled by adding

spring-boot-starter-actuator

-Logging and Security

-Less configuration

-embedded tomcat

5. how to create a Spring Boot application using Maven.

Ans-Spring Boot CLI

-Spring initializer

-Spring Maven project

-Spring Starter Project Wizard**

6. possible sources of EXTERNAL CONFIGURATION? ***question jab koi puchhe tab smj ana chahiye

Ans. Application properties : by default spring uses Application.properties to load properties

Command Line properties : command line arguments are converted to properties

application-{profile} properties : These properties are loaded from the application-{profile}.properties file or its YAML file.

The{profile}

placeholder refers to an active profile.

7. Spring Boot starters and what are available the starters?

Ans: Dependencies used to support applications for fast development of application.

Available Starters are :

spring-boot-starter: it includes core features such as logging , auto configuration.***

spring-boot-starter-jdbc : it is used to access database using jdbc

spring-boot-starter-data-jpa : it is used to access database using JPA

spring-boot-starter-web : it is used for building web-application such as restful and mvc application

spring-boot-starter-security : it is used to secure application

spring-boot-starter-aop: It is used to provide aspect oriented programming

spring-boot-starter-test: It is used to test spring application

8. Thymeleaf and how to use thymeleaf?

Ans. It is server-side java engine used to "build templates for web-application" and can be integrated with HTML5, JavaScript ,XML, CSS.

It can be enabled using spring-boot-starter-thymeleaf.

We can place the thyme leaf templates which are just the HTML files in src/main/resources/templates/

9. Spring Boot DevTools?

Ans. As name suggest it is set of tools used in development environment which makes development easier .It is automatically

disabled in production environment.it can be enabled using spring-boot-devtools.

This is one of the amazing features provided by Spring Boot, where it restarts the spring boot application whenever any changes are being made in the code.

Here, you don't need to right-click on the project and run your application again and again. Spring Boot dev tools does this for you with every code change.

10. @RestController , @RequestMapping and @GetMapping?

Ans. @RestController is combination of @ResponseBody and @Controller. It is class level annotation.

-It converts the response to JSON or XML.

-It ensures that data returned by each

method will be written straight into the response body instead of returning a template.

@RequestMapping is used to map http get,put,post,delete request with method.it can be used at class and method level.

@GetMapping is only an extension of RequestMapping which specifically used for get request

Eg: @RestController

```
@RequestMapping("bank-details")
public class DemoRestController{
    @GetMapping("/{id}", produces ="application/json")
    public Bank getBankDetails(@PathVariable int id){
        return findBankDetailsById();
    }
}
```

11. What is POM ?

Ans Project Object Model

12. Auto-configuration in Spring Boot and how to disable the auto-configuration?

Ans. It automatically configures the Spring application based on the dependencies(jars added in Maven Dependencies) that we have added. To disable the auto-configuration property we have to use "exclude" property of @EnableAutoConfiguration

Syntax: @EnableAutoConfiguration(exclude={Class Name})

Eg: @EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})

13. @SpringBootApplication and @EnableAutoConfiguration annotation?

Ans:@EnableAutoConfiguration : Used to enable auto-configuration and component scanning in your project.

It is a combination of

@Configuration and @ComponentScan annotations

@SpringBootApplication :Used in the main class(Configuration class) or bootstrap class

It is a combination of
@Configuration, @ComponentScan and @EnableAutoConfiguration annotations.

14. Spring Boot key components?

Ans -Spring Boot auto-configuration.
-Spring Boot Actuators.
-Spring Boot starter POMs.
-Spring Boot CLI.

15. Why Spring Boot over Spring?

Ans -Spring Boot auto-configuration.
-Spring Boot Actuators.
-Spring Boot starter POMs.
-Embedded tomcat server.

16. How does Spring Boot works?

Ans. Spring Boot automatically configures your application based on the dependencies you have added. The entry point of the spring boot application is the class that contains @SpringBootApplication annotation and the "main method".

Spring Boot automatically scans all the components included in the project by using @ComponentScan annotation.

```
@SpringBootApplication  
public class MyApplication {
```

```
    public static void main(String[] args) {  
  
        SpringApplication.run(MyApplication.class);  
        // other statements  
    }  
}
```

SpringApplication#run method to bootstrap the application.

17. What is Spring Boot CLI and what are its benefits?

Spring Boot CLI is a command-line interface that allows you to create a spring-based java application using Groovy(Groovy is scripting language).

Example(Benefit): You don't need to create getter and setter method or access modifier, return statement. If you use the JDBC template, it automatically loads for you.

18. most common Spring Boot CLI commands?

Ans.-run, -test, -grap, -jar, -war, -install, -uninstall, --init, -shell, -help

To check the description, run spring --help from the terminal.

run :run a spring groovy script

test:run a spring groovy script test

grab:download spring groovy script to ./repository

jar :create a self contained executable jar file from a spring groovy script

install:install dependencies to lib directory

init :initialize a new project using spring initializer

19. Basic Annotations that Spring Boot Offers?

Ans. @EnableAutoConfiguration and @SpringBootApplication

20. Spring Boot dependency management?

Ans. configuration of dependency automatically without you specifying the version for any of that dependencies.

21. How to change port of tomcat ?

Ans. By using the "server.port" in the "application.properties".

23. Can we override or replace the Embedded tomcat server in Spring Boot?

Ans. Yes, by using the Starter dependency in the pom.xml file. we can use spring-boot-starter-jetty as a dependency for using a jetty server in your project.

24. Can we disable the default web server(Tomcat 9) in the Spring boot application?

Ans. Yes, we can use application.properties to configure the web application type i.e ***spring.main.web-application-type=none***.

25. difference between @RestController and @Controller in Spring Boot?

Ans. @Controller Map of the model object to view or template.

@RestController simply returns the object(response) . It converts the response to JSON or XML.

26. What are the actuator-provided endpoints used for monitoring the Spring boot application?

Ans./Health

/Info

/Beans : the list of all the spring beans in your application

/auditevents : Exposes audit events information for the current application

/env : returns the list of all the environment properties of running the spring boot application.

/Mappings

/Configprops

/Httptrace

/Threaddump

/Shutdown

27. IOC container?

Ans. It manages Bean(object) creation and its life-cycle and also injects dependencies into the class.

28. How to enable debugging log in the spring boot application?

Ans. We can set the logging.level.root=debug property in application.property file.

28. What is dependency Injection?

The process of injecting dependent bean objects into target bean objects is called dependency injection.

Setter Injection: The IOC container will inject the dependent bean object into the target bean object by calling the setter method.

Constructor Injection: The IOC container will inject the dependent bean object into the target bean object by calling the target bean constructor.

Field Injection: The IOC container will inject the dependent bean object into the target bean object by Reflection API.

29. Where do we define properties in the Spring Boot application?

Ans.we can define application and spring-boot properties in application.properties.we can create this file or if we use spring initializer it will automatically come.if file is there then spring boot will load this file automatically

30.What is response entity in spring boot?

Ans.Response Entity is basically an HTTP response which includes headers, status code and body of your response.

31.What is bootstrapping/loading program in computer) in spring boot?

Ans.One of the way to bootstrap your spring boot application is using Spring Initializer.

you can go to the official website of spring and select your version, and add you groupID, artifactId and all the required dependencies. And then you can create your restEndpoints and build and run your project.

There you go, you have bootstrapped your spring boot application.

32.Spring boot introduced in which year?

Ans.2002

33.How to handle exceptions in spring boot?

Ans.To handle exceptions in spring boot, you can use @RestControllerAdvice annotation to handle your exceptions "globally". In order to handle "specific exception" and send customized response, you need to use @ExceptionHandler annotation.

34.What is the difference between jar and war(Web Application Resource) files?(not to learn)

Ans.war file is a Web Application Archive which runs inside an application server while a . jar is Java Application Archive that runs a desktop application on a user's machine.

A war file is a special jar file that is used to package a web application to make it easy to deploy it on an application server.

35.Profiles in spring Boot?

Ans.While developing the application we deal with multiple environments such as dev,test, Prod, and each environment requires a different configuration.

For eg each environment works on different port so we can use application-{profile}.properties file .

In application-dev.properties we add server.port=3000

In application-test.properties we add server.port=4000

In application-Prod.properties we add server.port=5000

In application.properties we make profile as active by spring.profiles.active(spa)={profile}

36. How to create war file in spring boot?

Ans.To create a war file in spring boot you need to define your packaging file as war in your pom.xml(if it is maven project).

Then open terminal (Ctrl+Alt+T)just do maven clean and install so that your application will start building. Once the build is successful, just go into your Target folder and you can see .war file generated for your application.

37.Mention the minimum requirements for a Spring boot System.

Ans.Spring Boot ->2.1.7.RELEASE

Java 8 +
Spring Framework 5.1.9 +
Explicit build support
Maven 3.3+
Tomcat 9.0 - Servlet Version 4.0

38. Can you give an example for `ReadOnly` as true in Transaction management?

Ans.onsider a scenario, where you have to read data from the database. For example, let us say you have a customer database, and you want to read the customer details such as customerID, and customername. To do that, you will set read-only on the transaction

39. Can we create a non-web application in Spring Boot?

Ans. Yes, we can create a non-web application by removing the web dependencies.

1. Explain JDK, JRE and JVM?

Ans. JDK

JRE

Java Development Kit Environment	JVM	Java Runtime
JDK provides tools like compiler, libraries and classes for code development run the program.	Java Virtual Machine Runtime environment to execute Java byte code.	JRE provides required by JVM to

2. difference between ArrayList and vector in Java?

Ans. Array list
 not legacy class
class

vector legacy

```
        ArrayList is not synchronized.  
synchronized.  
        fast  
        new capacity=initial capacity*3  
capacity=initial capacity*2  
        Cursors -Iterator ,List iterator  
,List iterator,Enumeration
```

Vector is slow

now

cursors-Iterator

3.What is a classloader in Java?

Ans. Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

Bootstrap ClassLoader: This is the first classloader which is the superclass of Extension classloader. It loads the rt.jar file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.

Extension ClassLoader: This is the child classloader of Bootstrap and parent classloader of System classloader.

System/Application ClassLoader: This is the child classloader of Extension classloader.

It loads the class files from the classpath.

to the current directory.

using "-cp" or "-classpath" switch.
classloader.

By default, the classpath is set

You can change the classpath

It is also known as Application

4.What is a package in Java? List down various advantages of packages.
Ans.packages are used to store classes and interfaces.

-modularize the code

Advantages:

-provide hierarchical structure so it becomes easy to locate classes

-package can contain hidden classes which cannot be accessed

outside package

-helps to avoid name clash

5.Difference between String, StringBuilder, and StringBuffer.

Ans.	String	StringBuilder
StringBuffer		
Storage Area	Constant String Pool	Heap Area
Mutability	Immutable	Mutable
Mutable		
Thread Safety	Yes (sync)	No (Non -sync)
Yes (sync)		
Performance	Fast (2)	More efficient (1) Less
efficient (3)		
eg.appending a character is fast in StringBuilder than string then StringBuffer		

Eg.

```
String first = "InterviewBit";
String second = new String("InterviewBit");
// StringBuffer
StringBuffer third = new StringBuffer("InterviewBit");
// StringBuilder
StringBuilder fourth = new StringBuilder("InterviewBit");
```

6.JIT compiler.

Ans.JIT stands for Just-In-Time and it is used for improving the performance during run time.

it compiles byte-code having similar functionality at same time.

it converts byte-code to native machine code

java source code(.java)----javac compiler---->byte code(.class)----
---jit compiler----->native code

7.access modifiers in Java?

Ans.They are used to restrict the access of a class, constructor, data member and method in another class.

Java supports four types of access modifiers:

-Default

-Private

-Protected

-Public

Scope --> private < default < protected < public

8.Differentiate between the constructors and methods in Java?

Ans.methods

constructors	
Must have a return type	Do
not have any return type	
Needs to be invoked explicitly	Needs to
be invoked explicitly	
No default method is provided by the compiler	A default constructor
is provided by the compiler if the class has none	
can be of any name	name
same as class name	

9.Differentiate between static and non-static methods in Java.

Ans. Static methods

Non-Static methods	
It is class level property	it
is object level property	
It can be called using class name and . operator	it is called
using object and . operator	
it can access other static members	it can
access static and non static members	

10.main concepts of OOPs in Java?

class, Object, abstraction, Encapsulation

Ans.Inheritance: Inheritance is a process where one class acquires the properties of another.

Polymorphism :Polymorphism is the ability of a variable, function or object to take multiple forms.

11.What is the difference between a local variable and an instance variable?

Ans.a local variable is typically used inside a method, constructor, or a block and has only local scope.

instance variables are declared within a class, but outside a method.

12.Explain public static void main(String args[]) in Java.

Ans.public: Public is an access modifier

static: It is a keyword in java which identifies it is class-based.

void: It is the return type of the method.

main: It is the name of the method which is searched by JVM as a starting point for an application

String args[]: It is the parameter passed to the main method.

13.Why Java is platform independent?

Ans.compiler compiles the code and then converts it to platform-independent byte code

14.Why Java is not 100% Object-oriented?

Ans.bcoz it has primitive data types such as boolean, byte, char, int, float, double, long, short which are not objects.

15. wrapper classes in Java?

Ans.primitive data types are converted to classes .so these classes are wrapper classes

16.What is singleton class in Java and how can we make a class singleton?

Ans.Singleton class is a class whose only one instance can be created at any given time, in one JVM

A class can be made singleton by making its "constructor private."

17.Why pointers are not used in Java?

Ans.pointers are complex and unsafe.java wants to simplify code process and pointers can easily access and manipulate memory address. so to avoid direct access to memory we use "references"

18.What is a Map in Java?

Ans.-Map is an interface of Util package
-It has key value pairs.
-Map doesn't contain duplicate keys.
-Each key can map at max one value.

19.Polymorphism

Ans.Polymorphism means multiple forms

There are two types of polymorphism in java:

compile-time polymorphism: is achieved thru method-overloading.
call to a particular method

is known at compile time

runtime polymorphism/dynamic polymorphism/dynamic method dispatch:
is achieved thru method-overriding
call to a particular method

is known at compile time.

point to chilg class object
parent class reference will

20. What is inheritance in Java?

Ans.the properties of one class can be inherited by the other.

Parent class (Super or Base class)-class whose properties are inherited
Child class (Subclass or Derived class)-A class which inherits the
properties

21.types of inheritance

Ans.Single Inheritance: one class inherits the properties of another

Multilevel Inheritance: When a class is derived from a class which is
also derived from another class

Hierarchical Inheritance: When a class has more than one child classes

Hybrid Inheritance: Hybrid inheritance is a combination of two or more
types of inheritance.

22.Can you override a private or static method in Java?

Ans.No

23.Why is multiple inheritance not supported in java?

Ans. Consider a scenario where A, B, and C are three classes. The C
class inherits A and B classes. If A and B classes
have the same method and you call it from child class object, there will
be ambiguity to call the method of A or B class.

The problem is commonly referred to as Diamond Problem.
there will be a compile time error.

```
class A{
    void msg(){System.out.println("Hello");}
}
class B{
    void msg(){System.out.println("Welcome");}
}
class C extends A,B{//suppose if it were
```

```

Public Static void main(String arg      s[]){
    C obj=new C();
    obj.msg(); //Now which msg() method would be invoked?
}
}

```

25. What is a marker interface/tagging interfaces ?

Ans. an empty interface is called the Marker interface which do not have any methods or variables.

They are there for helping the compiler and jit to get run time-related information regarding the objects.

eg. Serializable, Cloneable etc.

```

public interface Serializable{
}

```

26.What is constructor chaining in Java?

As.constructor chaining is the process of calling one constructor from another

Constructor chaining can be achieved in two ways:

- Within the same class using this()
- From base class using super()

27.finalize keyword

Ans.Prior to the garbage collection of an object, the finalize method is called so that the clean-up activity is implemented.

```

public class yippee {

    public static void main(String[] args)
    {
        yippee obj = new yippee();
        System.out.println(obj.hashCode());
        obj = null;
        // calling garbage collector
        System.gc();
        System.out.println("end of garbage collection");

    }
    @Override
    protected void finalize()
    {
        System.out.println("finalize method called");
    }
}

```

28.When can you use super keyword?

Ans.-Accessing data members of parent class when the member names of the class and its child subclasses are same.

-To call the default and parameterized constructor of the parent class inside the child class.

-Accessing the parent class methods when the child classes have overridden them.

29.Can the static methods be overloaded?

Ans.Yess

30. Can the static methods be overridden?

Ans. No

But In the case of a child class, a static method with a method signature exactly like that of the parent class can exist without even throwing any compilation error. The phenomenon mentioned here is popularly known as method hiding, and overriding is certainly not possible.

31. main objective of garbage collection?

Ans. The main objective of this process is to free up the memory space by deleting the unnecessary and unreachable objects them this ensures that the memory resource is used efficiently, but it provides "no guarantee" that there would be sufficient memory for the program execution.

32. What part of memory - Stack or Heap - is cleaned in garbage collection process?

Ans. Heap.

33. Difference between Tree Set and Hash Set in Java?

Ans. Hash Set	Tree Set
underlying ds is hashtable	TreeMap
null values allowed	does not allow null
values allowed	
heterogenous data allowed	Not allowed
data is not sorted	data is sorted

34. What are the differences between HashMap and HashTable in Java?

Ans. HashMap allows one null key and multiple null values. Hashtable doesn't allow any null key or value.

HashMap is a not legacy class
Hashtable is a legacy class.
non-synchronized
synchronized

35. Java works as "pass by value" or "pass by reference" phenomenon?

Ans. Java always works as a "pass by value". There is nothing called a "pass by reference" in Java whenever an object is passed in method and it is received in another variable so a copy of reference is passed to that variable and both will point to same memory address

```
Eg1.class Driver {  
    public static void main(String[] args)  
    {  
        //create a reference  
        InterviewBitTest ibTestObj = new InterviewBitTest(20);  
        //Pass the reference to updateObject Method  
        updateObject(ibTestObj);  
        //After the updateObject is executed, check for the value of num  
        //in the object.  
        System.out.println(ibTestObj.num);  
    }  
    public static void updateObject(InterviewBitTest ibObj)  
    {  
        // Point the object to new reference
```

```

        ibObj = new InterviewBitTest();
        // Update the value
        ibObj.num = 50;
    }
}
Output:
20

eg2.class Driver{
    public static void main(String[] args)
    {
        //create a reference
        InterviewBitTest ibTestObj = new InterviewBitTest(20);
        //Pass the reference to updateObject Method
        updateObject(ibTestObj);
        //After the updateObject is executed, check for the value of num
        in the object.
        System.out.println(ibTestObj.num);
    }
    public static void updateObject(InterviewBitTest ibObj)
    {
        // no changes are made to point the ibObj to new location
        // Update the value of num
        ibObj.num = 50;
    }
}
Output:
50

```

36.What happens if the static modifier is not included in the main method signature in Java?

Ans.There wouldn't be any compilation error. But then the program is run, since the JVM cant map the main method signature.

37.How does an exception propagate in the code?

Ans.when an exception occurs it will find catch block in same class and execute catch block .if catch block not found. it will go to calling method and will find catch block. this will happen until catch block is not found.
. If the match is not found, then the program gets terminated in the main method.

38.when does object becomes eligible for garbage collection?

Ans.When an object does not have any reference, it becomes eligible for garbage collection.

39. Is exceeding the memory limit possible in a program despite having a garbage collector?

Ans.Yess bcoz we dont knw when garbage collection will happen it can happen at any point of time

40.In the given code below, what is the significance of ... ?

```

public void fooBarMethod(String... variables){
    // method code
}

```

Ans.These are varargs (variable arguments).It can receive multiple arguments

```
fooBarMethod("foo", "bar");
fooBarMethod("foo", "bar", "boo");
fooBarMethod(new String[]{"foo", "var", "boo"});
public void myMethod(String... variables){
    for(String variable : variables){
        // business logic
    }
}
```

41. Is it possible to import the same class or package twice in Java and what happens to it during runtime?

Ans. It is possible to import a class or package more than once, however, it is redundant because the JVM internally loads the package or class only once.

42. In case a package has sub packages, will it enough to import only the main package? e.g. Does importing of com.myMainPackage.* also import com.myMainPackage.mySubPackage.*?

Ans. This is a big NO. We need to understand that the importing of the sub-packages of a package needs to be done explicitly.

Importing the parent package only results in the import of the classes within it and not the contents of its child/sub-packages.

43. Will the finally block be executed if the code System.exit(0) is written at the end of try block?

Ans. NO. The program gets terminated which is why the finally block never gets executed.

44. What are the possible ways of making object eligible for garbage collection (GC) in Java?

Ans. -point the reference variable to null
eg. String s1 = "Some String";
// s1 referencing String object - not yet eligible for GC
s1 = null; // now s1 is eligible for GC

-point the reference variable to another object.

```
eg. String s1 = "To Garbage Collect";
s1 = "Another Object";
```

-When a reference is local to some method then it will be removed from the stack as soon as the method has finished executing

```
public class Tester {
public static void main(String[] args) {
    Customer cust1 = new Customer(31, 9663472291L);
}
}
```

45. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

Ans. refer for diagram-->(<https://www.interviewbit.com/java-interview-questions/#java-platform-independent>)

In the case of ArrayList, data storing in the form of primitive data types (like int, float, etc.) is not possible.

The data members/objects present in the ArrayList have references to the objects which are located at various sites in the memory. Thus, storing of actual objects or non-primitive data types (like Integer, Double, etc.) takes place in various memory locations.

46.What are the differences between Heap and Stack Memory in Java?

Ans.Stack

1.Stack contains local primitive variables and
All objects along with their instance variables are
Non-primitive(references) to an object.

Heap
contains
stored in the

heap

2.Stack memory is used only by one thread(Main)
by all the parts of the application.

3.Stack memory can't be accessed by other threads
heaps are globally accessible.

4.Exists until the end of execution of the thread. Heap memory lives
from the start till the end of application execution.

47.Java String Pool? or How is the creation of a String using new()
different from that of a literal?

Ans.Java maintains a string pool in Heap memory.whenever a new string. is
created.String pool first checks whether the
object is already present in the pool or not.If it is present, then the
same reference is returned to the variable else
new object will be created in the String pool .

String s1="Emily";

String s2="Emily";

here s1 and s2 will point to same value in heap memory.

String s3=new String("Emily");

here new object s3 with Emily value is stored

String s4=new String("Emily").intern();

here s3 will point to same value in heap memory as s1 and s2 is pointing

String s5="mango";

here new object is created

48.difference between equals() and == in Java?

Ans.Equals() method is defined in Object class in Java and used for
checking equality of two objects defined by business logic.

"==" or equality operator in Java is a binary operator and used to
compare primitives and objects(checks if both the objects are pointing to
the same memory location.)

49.Why Java Strings are immutable in nature?

Ans.String class maintains a literal pool, which is like a cache in the
heap."values" are shared between multiple variables .

changing value from one variable will affect values of other variables
also .so to avoid this,string is made immutable.

50. What is composition in Java?

Ans.Composition is again a specialized form of Aggregation. It is a
strong type of Aggregation.

In Aggregation ,there is weak bonding between parent and child class.But
in Composition,there is strong bonding between parent and child class.
child class does not have independent existence in composition.

eg. Car has a Music player (Aggregation bcoz car and music player can
exist independently)

Car has a Engine (composition bcoz car and Engine player are dependent on each other)

51.What is a copy constructor in Java?

Copy constructor is a member function that is used to initialize an object using another object of the "same class".

```
class Student{
    int id;
    String name;

    Student(int i, String n) {
        id = i;
        name = n;
    }

    Student(Student s){ -->Copy constructor
        id = s.id;
        name = s.name;
    }

    public static void main(String args[]){
        Student s1 = new Student(111, "Karan");
        Student s2 = new Student(s1);
    }
}
```

----- More

Questions :Set - 2 -----

-
Abstarct Class Constructor

Q1).Difference between HashMap and ConcurrentHashMap ?

Ans.

1.Thread Safe	HashMap
	ConcurrentHashMap
2.Synchronized using	Not Thread Safe
	Thread Safe
	Not Synchronized.We have to synchronize
	It synchronize only certain portion of map
	synchronizedmap.Whenever we
synchronize hashmap	it locks whole map.
3.Null Keys	One Null Key is allowed
	Null Keys Not allowed
4.Performance	faster
	slower
5.Concurrent Modification when one thread Exception	throws Concurrent Modification Exception
	tries to modify the thread when other
thread is iteraing	does not throw Concurrent Modification
	Exception

Q2) ConcurrentHashMap (Learnt from Durga Sir)

Ans)*** concurrency level = number of concurrently updating threads

1.Underlined Ds is hashtable
2.Null Key and Null Values not allowed + Never throw Concurrent Modification Exception
3.For Read ,Thread wont require any locks
For Update ,Thread requires lock of only particular part of Map(Bucket Level Lock)
4.Concurrent Update achieved by internally dividing map into 16(default initial capacity) small portions. So It is called to have 16 concurrency level
5.So as many read operations as u want +16 update operations

HashMap : Not Thread Safe Multiple Threads can perform operations ..data inconsistency can occur
HashTable : Thread Safe but Performance Not good... One thread is allowed to perform operation as complete hashmap is locked
ConcurrentHashMap: Thread safe With Best Performance.... Bucket Level lock

case 1 :What if Initial capacity=16 and concurrency level=8 so for every 2 buckets one lock will be maintained

case 2 :What if Initial capacity=16 and concurrency level=32 so for every buckets 2 lock will be maintained

Q3)Volatile Keyword in JAVA

Ans 1)Volatile keyword is used to modify the value of a variable by different threads
2)The volatile keyword does not cache the value of the variable and always read the variable from the main memory.
3)The volatile keyword cannot be used with classes or methods.
4)However, it is used with variables.

Code Example :

```
public class TestingDemo extends Thread {  
  
    TestingDemo(String name){  
        super(name);  
    }  
  
    static volatile boolean flag=false; *****STATIC AND VOLATILE  
    Dono likho  
  
    @Override  
    public void run() {  
        try {  
            flag=true;  
            while(flag) {  
                System.out.println("Roaming :" +flag);  
                Thread.sleep(50); #2 -> jab main thread ka sleep time  
                khatam hote hi yaha ayege toh y sleep hogा or fr #3 par jaega  
            }  
  
        }catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```

        }

    public static void main(String[] args) throws InterruptedException
{
    TestingDemo d1=new TestingDemo("Pawan");

    d1.start();

    Thread.sleep(500);--> #1 : yaha par main thread chahtha h
"Pawan" thread chale pehle
    flag=false; --> #3 yaha par value change ho jaegi or fr
"Pawan" thread ka execution band ho jaega

    System.out.println(d1.flag); -->false
    System.out.println(flag); -->false
}
}

```

Agr humne flag=false Sleep(5000) se pehle likh dete toh fr main pehle chalke flag ko false kr deta or flag hamesha true ho jata jab "Pawan" thread me jata or infinite loop chalta rehta

Q4) Difference between Static and volatile?

Ans)Declaring a static variable in Java, means that there will be only one copy, no matter how many objects of the class are created. The variable will be accessible even with no Objects created at all. However, threads may have locally cached values of it.

When a variable is volatile and not static, there will be one variable for each Object. So, on the surface it seems there is no difference from a normal variable

This means that if two threads update a variable of the same Object concurrently, and the variable is not declared volatile, there could be a case in which one of the thread has in cache an old value.

Even if you access a static value through multiple threads, each thread can have its local cached copy! To avoid this you can declare the variable as static volatile and this will force the thread to read each time the global value from Main Memory.

Q5) Statement vs PreparedStatement vs CallableStatement ?

Ans)

Statement :

1.How to Create Statement ?

Statement statement=conn.createStatement();

2.Advantage :Can be used to execute multiple types of sql queries
int rows= statement.executeUpdate(String INSERT/DELETE/UPDATE):

ResultSet=statement.executeQuery(String READ);

3.Disadvantages :If u want to excute same query multiple times..then every time statement will be compiled and executed.

if u want to execute a crud statement 1 crore times then 1 crore times it will compile. so performance dec

PreparedStatement :

1.How to create ?

public PreparedStatement prepareStatement(String query) throws
SQLException{}
PreparedStatement ps=conn.prepareStatement(Query)

2.Advantages : As we are giving at time of PreparedStatement connection creation so Query will be compiled only once and can be executed as many times ..so performance will be increased

3.Disadvantages : if we want to execute update / delete/ select or any other sql then we need a separate instance of prepareStatement again

Callable Statement :

1.How to Create ?

```
public CallableStatement prepareCall("{ call procedurename(?,?....?) }");
```

The example to get the instance of CallableStatement is given below:

```
CallableStatement stmt=con.prepareCall("{call myprocedure(?,?)}");
```

Q6)Can we execute a java program without a main method?

Ans)Yes, we can execute a java program without a main method by using a static block.

Static block in Java is a group of statements that gets executed only once when the class is loaded into the memory by Java ClassLoader, It is also known as a "static initialization block."

```
class StaticInitializationBlock{  
    static{  
        System.out.println("class without a main method");  
        System.exit(0);  
    }  
}
```

Output :class without a main method"

In the above example, we can execute a java program without a main method (works until Java 1.6 version). Java 7 and newer versions don't allow this because JVM checks the presence of the main method before initializing the class

Q7)is constructor inherited ?

Ans) NO

Q8)Can we make a constructor final?

Ans) No

The child class inherits all the members of the superclass except the constructors. In other words, constructors cannot be inherited in Java therefore you cannot override constructors. So, writing final before constructors makes no sense. Therefore, java does not allow final keyword before a constructor

Q9)Explain JDBC Drivers ?

Ans) JDBC Driver is a software component that enables "java application to connect with the database".

There are 4 types of JDBC drivers:

- 1.JDBC-ODBC bridge driver
- 2.Native-API driver (partially java driver)
- 3.Network Protocol driver (fully java driver)
- 4.Thin driver (fully java driver)

1.JDBC-ODBC bridge driver : The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls

2. Native-API driver (partially java driver) : The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.
3. Network Protocol driver (fully java driver) : uses middleware (application server) that converts JDBC calls directly or indirectly into the "vendor-specific database protocol". It is fully written in java.
4. Thin driver (fully java driver) : thin driver converts JDBC calls directly into the "vendor-specific database protocol".

Q10) ACID Properties in DBMS ?

Ans) Atomicity : All or Null ... i.e a transaction has to be fully executed or not executed at all .

eg . Suppose i made a paytm transaction and mobile switch off in between then Atomicity comes into role

Consistency : Measure of Correctness . Eg User A has Rs.100 and User B has Rs.400 then user A transfer 50 rupee to

User B so now User A has 50/- and User B has 450/-.

How to Check Consistency ?

Before Execution of Transaction => Total Amount of User A and B is 500

After Execution of Transaction => Total Amount of User A and B is 500

so Before and After transaction amounts are same so it is said to be consistent

Isolation : Suppose Multiple transactions are happening so no transaction should hinder or interfere any other transaction

Durability : Once transaction is committed .. updates to Db are stored and it should not get affected on system failure .

Eg. Bank server is not working that does not mean that Db data is lost/modified

Q11). When To use Interface and when to use Abstract Class ?

- Ans) 1) Define diff between variables , Methods and Multiple Inheritance
- 2) instance initialization blocks and static initialization blocks in an abstract class, whereas we can never have them in the interface
- 3) Abstract classes may also have constructors which will get executed during the child object's instantiation.
- 4) We can restrict Interface to have 1 functional interface by using @FunctionalInterface but it cannot be done with abstract class

When to Use an Interface (No Close Relationship):

1. If the problem needs to be solved using multiple inheritances
2. When unrelated classes implement our interface
3. When application functionalities have to be defined as a contract, but not concerned about who implements the behavior.

Use the interface when our problem makes the statement "A is capable of [doing this]". For example, "Clonable is capable of cloning an object"

```
public interface Sender {
    void send(File fileToBeSent);
}
public class ImageSender implements Sender {
    @Override
    public void send(File fileToBeSent) {
        // image sending implementation code.
    }
}
```

```
        }
    }
"Sender is capable of sending a file"
    We have used ImageSender ...we can further use VideoSender,
DocumentSender
```

When to Use Abstract Class (Close Relationship):

- When trying to use the inheritance concept in code (share code among many related classes), by providing common base class methods that the subclasses override
 - If we have specified requirements and only partial implementation details

Use abstract classes and inheritance when our problem makes the evidence "A is a B".

For example, "Dog is an Animal", "Lamborghini is a Car",

Note : <https://www.baeldung.com/java-interface-vs-abstract-class>

Q12).IMP:explain bean life cycle

container started ---> bean instanciated ---> dependency injected --->
post construct method --->utility methods of class ---> predestroy method

Q13) .Constructor Injection vs Field Injection vs Setter Injection ?(RIP -
-MTC (My Tiny Childhood))

Ans) Constructor Injection

Field Injection

Setter Injection

Readability | Better. Constructors visually stand separate.

| The Best - Less boilerplate code

Worst

from methods

1

1

Immutability | Supports immutability. (field declared should be

|No immutability (This happens after |No immutability.

(private and final)

(object construction)

1

Maintainability | Constructor injection helps in identifying the list of
| Since Autowired fields can easily mix | Same as Field Injection
| dependencies and can easily be used to make the
code | with regular fields, it gets difficult |
| more readable (All Dependency at one place)
| to track the dependencies of the class |

Testing |Mocking the dependencies become easier as they can be |Mocking dependencies Becomes difficult |Mocking is easy as mock object

|mocked and passed using constructor
|
|is directly set

Cyclic Dependency| Cyclic dependencies can easily be detected and fixed
| Does not detect | Does not detect
detection detection |using Constructor Injection.

Performance |having constructor injection means all the beans have |each object created independently and |each object created independently and

|to be created in correct order of dependencies
which in|dependency is injected later using |dependency is injected later using
|turn increases the "startup time" of the application |reflection, startup time is faster |setter, startup time is faster

Field Injection :

Advantages :

- The shortest way to inject a dependency. You only need to add the @Autowired over the field, and that's it.

Disadvantages :

- Can't be used with immutable fields. It forces you define the attributes mutable.
- More difficult to set a mock dependency in a unit test.

Constructor Injection:

Advantages

- You can define immutable instances by making the attributes final.
- You can define dependencies in unit tests directly when you create the instance of bean you want to test.

Disadvantages

- You need to write the constructor (which is more code in the class than in the field injection). However, with a dependency such as Lombok, this disadvantage disappears as you'll learn later in this article.

Q14) Difference between Delete and Truncate ?

Ans) Delete

Truncate

1) It is a DML(Data Manipulation Language) command.

It is a DDL(Data Definition Language) command.

2) WHERE Clause can be used

Can't be used

3) The DELETE statement removes rows one at a time and records

TRUNCATE TABLE removes the data by deallocating the data pages used to store

an entry in the transaction log for each deleted row.

the table data and records only the page deallocations in the transaction log.

```

        (Entry of Page deallocation in transaction
log)
4) Slower
    Faster
5) To use Delete you need DELETE permission on the table.           To
use Truncate on a table we need at least ALTER permission on the table.
6) This command can also active trigger.
    This command does not active trigger.
7) DELETE statement occupies more transaction spaces than Truncate.
    Truncate statement occupies less transaction spaces than DELETE.

```

Q15) Explain @Required ?

Ans) The @Required annotation applies to bean property "setter methods" mtlb bean ki property ke setter method pe. it indicates that the affected bean property must be populated in XML configuration file at configuration time. Otherwise, the container throws a "BeanInitializationException" exception.

Where Used --> On setter methods
Exception --> BeanInitializationException

Code Example :

```

public class Student {
    private Integer age;
    private String name;

    @Required *****
    public void setAge(Integer age) {
        this.age = age;
    }

    @Required*****
    public void setName(String name) {
        this.name = name;
    }
}

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("Beans.xml");***** Focus Here as
Exception will generate from this line

        Student student = (Student) context.getBean("student");
        System.out.println("Name : " + student.getName() );
        System.out.println("Age : " + student.getAge() );
    }
}

Beans.xml -
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
// Some Code

```

```
<!-- Definition for student bean -->
<bean id = "student" class = "com.tutorialspoint.Student">
    <property name = "name" value = "Zara" />
    <!-- property name = "age"  value = "11"--> **** THIS IS
COMMENTED
    </bean>

</beans>
```

Exception :
it will raise BeanInitializationException exception
Property 'age' is required for bean 'student'

Q16) Status Code
Ans.200 - OK
400- BAd request
401- unauthorized
403-Forbidden
404 - Not Found
500 - Internal Server Error
301 - Permanent Redirect

Q17) @PostConstruct
Ans) 1.Method Level annotation
2.it gets executed after the spring bean is initialized.
3.We can have only one method annotated with @PostConstruct annotation.
For instance, during development, we might want to create some default users:

Q17) @PreDestroy
Ans) 1.Method Level Annotation
2.It gets called when bean instance is getting removed from the "context".
Note :This is a very important point to understand - if your spring bean scope is "prototype" then it's not completely managed by the spring container and PreDestroy method won't get called.
Note :If there is a method named "shutdown" or "close" then spring container will try to automatically configure them as callback methods when bean is being destroyed.so shutdown() ,close(),@predestroy().First @predestroy() then shutdown() and close()
eg.release resources or perform other cleanup tasks, such as closing a database connection, before the bean gets destroyed.

Q18) @PostConstruct vs init method
Ans)
1.@PostConstruct is a JSR-250 annotation while init-method is Spring's way of having an initializing method.
2.If you have a @PostConstruct method, this will be called first before the initializing methods are called.
3.If your bean implements InitializingBean and overrides afterPropertiesSet , first @PostConstruct is called, then the afterPropertiesSet and then init-method.

This 3 step happens in the Bean Creation Life-Cycle Callback:
a) It is mentioned that @PostConstruct will be called.

- b) If InitializingBean is implemented, then afterPropertiesSet() will be called.
- c) If bean definition contains init-method or @Bean(initmethod="...") then it calls the init

Code example :

```
public class InitOrderofExecutionBean implements InitializingBean {

    private final static Logger LOG =
LoggerFactory.getLogger(InitOrderofExecutionBean.class);

    public InitOrderofExecutionBean(){
        LOG.info("Inside Constructor");
    }

    @PostConstruct
    public void postConstruct(){
        LOG.info("Inside @PostConstruct method");
    }

    public void init(){
        LOG.info("Inside bean init method");
    }

    @Override
    public void afterPropertiesSet() throws Exception {
        LOG.info("Inside afterPropertiesSet method");
    }
}

Output: Inside Constructor
Inside @PostConstruct method
Inside afterPropertiesSet method
Inside bean init method
```

Q19) Auto-Configuration in Spring Boot ?

Ans)

It attempts to auto configure your application based on jars added to CLASSPATH.

Suppose we are not having / using Spring Boot then
 #for MVC ,we need to configure (Configure means creation of beans)
 -Component scan
 -Dispatcher Servlet
 -view resolver

for Hibernate/Jpa ,we need to configure

-Datasource
 -entity manager factory
 -transaction manager

#Rest Api
 -Component scan
 -Datasource
 -Tomcat Server
 -RequestMappingHandler

So Solution that spring boot provides that Based on jars/dependency in classpath ..spring boot will auto-configure above things
 #How to see what classes are enabled for auto-configuration ?
 -MavenDependencies > spring-boot-autoconfigure.jar > META-INF
 >spring.factories > yaha classes k naam h jinko auto configure krna h
 # To See what is configured and what not configured for current project we can enable logs
 logging.level.org.springframework.web=DEBUG
 we find Positive (Beans Configured) and Negative Matches(Beans not configured) with 2 annotations in Debug ..
 @ConditionalOnClass -> if certain class found on class path or in jars then this Bean(on which this annotation is marked) is configured
 @ConditionalOnMissingBean -> if no bean of same name is present then it is configured

Q20) Circular /Cyclic Dependency ?

Ans) It can happen in Spring when using constructor injection. If we use other types of injections, we shouldn't have this problem since the dependencies will be injected when they are needed and not on the context loading.

```
@Component
public class CircularDependencyA {

    private CircularDependencyB circB;

    @Autowired
    public CircularDependencyA(CircularDependencyB circB) {
        this.circB = circB;
    }
}

Copy
@Component
public class CircularDependencyB {

    private CircularDependencyA circA;

    @Autowired
    public CircularDependencyB(CircularDependencyA circA) {
        this.circA = circA;
    }
}
```

Solution 1:A simple way to break the cycle is by telling Spring to initialize one of the beans lazily.

So, instead of fully initializing the bean, it will create a proxy to inject it into the other bean

The injected bean(CircularDependencyB) will only be fully created when it's first needed.

```
@Component
public class CircularDependencyA {

    private CircularDependencyB circB;

    @Autowired
    public CircularDependencyA(@Lazy CircularDependencyB circB) {
        this.circB = circB;
    }
}
```

```
}
```

Solution 2: Setter/Field Injection

Q21). Second Largest Integer in Array list using streams
Ans) //1,2,3,4,5,8,9

```
    List<Integer> list=Arrays.asList(2,4,1,8,3,9,5);
```

```
    System.out.println(list.stream().sorted(Comparator.reverseOrder()).  
skip(1).findFirst().get());
```

Q22). Summing List using Streams

```
Ans) List<Integer> integers = Arrays.asList(1, 2, 3, 4, 5);  
    Integer sum = integers.stream().reduce(0, (a, b) -> a + b);
```

Q23). Ways to Create Object in Class ?

Ans)There are five different ways to create an object in Java:

- a) Java new Operator
- b) Java Class.newInstance() method
- c) Java newInstance() method of constructor
- d) Java Object.clone() method
- e) Java Object Serialization and Deserialization

a) class_name object_name = new class_name()
b)-Java Class.newInstance() is the method of Class class.

```
    Product P=Product.class.newInstance()
```

c)-The newInstance() method belongs to java.lang.reflect.Constructor class.

-Both newInstance() method are known as reflective ways to create object.

```
Constructor<Product> obj =Product.class.getConstructor();  
Product obj1 = obj.newInstance();
```

Q24). Index / Indexing in SQL

Ans) Indexes are used to retrieve data from the database more quickly than otherwise.

The users cannot see the indexes, they are just used to speed up searches/queries.

Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

CREATE INDEX Syntax : Creates an index on a table. Duplicate values are allowed:

```
-CREATE INDEX index_name ON table_name (column1, column2, ...);
```

CREATE UNIQUE INDEX Syntax : Creates a unique index on a table. Duplicate values are not allowed:

```
-CREATE UNIQUE INDEX index_name ON table_name (column1, column2, ...);
```

Eg. (a) Indexing on 1 column

```
    CREATE INDEX idx_lastname ON Persons (LastName);
```

(b) Indexing on Multiple column\

```
    CREATE INDEX idx_pname ON Persons (LastName, FirstName);
```

Q25).Views in SQL

Ans) A view is a virtual table based on the result-set of an SQL statement.

Syntax :

```
CREATE VIEW view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

Eg.

```
CREATE VIEW [Brazil Customers] AS  
SELECT CustomerName, ContactName  
FROM Customers  
WHERE Country = 'Brazil';
```

```
SELECT * FROM [Brazil Customers];
```

Q26). Java Best Practices ?

Ans 1) Use Proper Naming Conventions
2) Class Members Should be Private
3) Avoid empty catch Blocks
4) Avoid Redundant Initializations
5) Use Advance For loop as compared to traditional for loop
6) Return Empty Collections instead of returning Null elements
7) Proper Handling of Null Pointer
8) Use Underscores in Lengthy Numeric Literals

```
int minUploadSize = 05437326;  
long debitBalance = 500000000000000L;
```

```
int minUploadSize = 05_437_326;  
long debitBalance = 5_000_000_000_000L;
```

Q27). @Primary in Spring Boot ?

Ans) Multiple Implementation of Interface is present and we want to Choose Preferred Implementation if nothing mentioned while autowiringso we use @Primary

It can be Used with @Component and @Bean

Q28). Why Constructor Injection is Preferred ?

Ans (a) All Required Dependencies Are Available at Initialization Time -The IoC container makes sure that all the arguments provided in the constructor are available before passing them into the constructor. This helps in preventing the infamous "NullPointerException."

(b) Constructor injection helps us to identify if our bean is dependent on too many other objects
-If our constructor has a large number of arguments this may be a sign that our class has too many responsibilities.
-Single Responsibility Principle Fails ... Restructure the Class

(c) Preventing Errors in Tests

-We can also pass mocks via setters, of course, but if we add a "new dependency" to a class, we may forget to call the setter in the test, potentially causing a NullPointerException in the test.

(d) Immutability

-Constructor injection helps in creating immutable objects because a constructor's signature is the only possible way to create objects
-Once we create a bean, we cannot alter its dependencies anymore

-With setter injection, it's possible to inject the dependency after creation, thus leading to mutable objects which, among other things, may not be thread-safe in a multi-threaded environment

Q29) Why Do we need Integer when we already have int ?

Ans (a) String to int casting not possible . So Integer class can be used to do this

```
int a = (int)"123"; -->This is Not possible  
Integer a=Integer.parseInt("123")
```

(b) Direct Conversion of value to other base

```
String bin = Integer.toBinaryString(123);  
String oct = Integer.toOctalString(123);
```

(c) Usage in Collections

Q30) Advantage of using SpringJDBC over Traditional JDBC API

Ans)-releasing of the database connection automatically

-JdbcTemplate provides a great exception handling mechanism...which converts SQLExceptions into RuntimeExceptions.....

which are generic and more informative, allowing developers to better identify the error

-The RowMapper or ResultSetExtractor is an interface typically used by the JdbcTemplate ... which allows to translates the SQL result direct into an Object or a list of Objects.

-JdbcTemplate provides methods to write the SQL queries directly....saving a lot of work

Q31) Advantages of Spring Boot JDBC over Spring JDBC

Ans) JDBC Using Spring Boot

JDBC Using Spring

1.one dependency "spring-boot-starter-jdbc" | Multiple dependencies like spring context and spring JDBC will make all modules available

2.Spring Boot will automatically register beans | we have to register explicitly

JdbcTemplate, NamedParameterJdbcTemplate

|

3.It Auto-Initiliaze DataSource Bean | it is required to create a database bean either using XML or javaconfig.

Q32) CSV Reader

```
Ans) FileReader filereader = new FileReader(file);  
CSVReader csvReader = new CSVReader(filereader);  
CSVReader csvReader = new  
CSVReaderBuilder(reader).withSkipLines(1).build();  
String[] nextRecord;  
while ((nextRecord = csvReader.readNext()) != null) {  
    for (String cell : nextRecord) {  
        System.out.print(cell + "\t");  
    }  
    System.out.println();  
}
```

Q33) . Immutable Class

Ans) 1. Make Class as Final
2. Make Variable as private
3. Make Variable as final
4. No setter methods
5. Make Deep copy of Objects

```
//THIS CLASS HAS COMPLETE GETTERS,SETTERS* AND CONSTRUCTORS

public final class ImmutableClass {

    private final String name;
    private final Engine engine;
    private final List<String> hobbies;
    private final List<Engine> engines;

    public ImmutableClass( String name,Engine engine) {

        this.name = name;
        Engine e=new Engine(engine.getSpeed());
        this.engine=e;
    }

    public String getName() {
        return name;
    }

    public Engine getEngine() {
        return new Engine(this.engine.getSpeed());
    }

    public List<String> getHobbies() {
        return new ArrayList<>(hobbies);
    }

    public List<Engine> getEngines() {
        return new ArrayList<>(engines);
    }
}

public class MainClass {

    public static void main(String[] args) {
        Engine engine=new Engine(10);
        ImmutableClass ic=new ImmutableClass(1, "pawan", engine);
        engine.setSpeed(20);
        System.out.println(ic);
    }
}
Output : ImmutableClass [id=1, name=pawan, engine=Engine [speed=10]]
```

Note : 1. Why do we need to make class as final ?

Bcoz child class can extend parent class ...variables are private so not inherited but methods inherited ...getId() and getName() inherited and it can be overridden to get some value .

Q34) Difference between @PutMapping and @PostMapping

Ans) Criteria

	POST
Use	PUT Create a new record Update a record
Idempotent	POST is non-idempotent PUT is Idempotent
Request Body	0 POST needs to send the entire resource as payload PUT sends the part to be updated in the request
Status Code	As new record gets created in each request, 201 is 200 when record gets updated, 201 if this record is new sent in response, 409 if duplicates are not
allowed	
cache	It cannot be cached. It can be cached.

Note : PUT is idempotent and POST is not

PUT requests are typically idempotent, meaning that multiple identical PUT requests will have the same effect as a single request. In contrast, POST requests are typically not idempotent, and multiple requests may have different effects. For example, if the first PUT request updates a record with a given updated data, subsequent duplicate request using the same data will not alter the system as it will update the same record with same data. However in POST request if duplicate data is sent, new records get created

a POST request must contain a body. Second, a PUT request does not require a body.

Q35) @Repository vs @Controller vs @service

Ans) @Repository : Its job is to catch platform specific exceptions and re-throw them as one of Spring's unified unchecked exception. For this, we're provided with PersistenceExceptionTranslationPostProcessor, that we are required to add in our Spring's application context .As we add @Repository ...PersistenceExceptionTranslationPostProcessor will be autoconfigured

@Controller : The dispatcher scans the classes annotated with @Controller and detects methods annotated with @RequestMapping annotations within them. We can use @RequestMapping on/in only those methods whose classes are annotated with @Controller and it will NOT work with @Component, @Service, @Repository etc...We cannot switch this annotation with any other like @Service or @Repository

@service : Apart from the fact that it's used to indicate, that it's holding the business logic, there's nothing else noticeable in this annotation; but who knows, Spring may add some additional exceptional in future.

Q36) Connecting Multiple DB ?

Ans)

1. Create 2 databases :

```
CREATE DATABASE `db1`  
CREATE DATABASE `db2`
```

2.create two tables called user1 in db1 and user2 in db2

3.Dependencies: mysql and jdbc

```

4.Application.properties
#first db
spring.datasource.jdbcUrl=jdbc:mysql://localhost:3306/db1
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driverClassName=com.mysql.jdbc.Driver

#second db
spring.second-db.jdbcUrl=jdbc:mysql://localhost:3306/db2
spring.second-db.username=root
spring.second-db.password=root
spring.second-db.driverClassName=com.mysql.jdbc.Driver

5. Create Configuration class
@Configuration
public class WebConfig {

    @Bean(name = "db1")
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource dataSource1() {
        return DataSourceBuilder.create().build();
    }

    @Bean(name = "jdbcTemplate1")
    public JdbcTemplate jdbcTemplate1(@Qualifier("db1") DataSource ds) {
        return new JdbcTemplate(ds);
    }

    @Bean(name = "db2")
    @ConfigurationProperties(prefix = "spring.second-db")
    public DataSource dataSource2() {
        return DataSourceBuilder.create().build();
    }

    @Bean(name = "jdbcTemplate2")
    public JdbcTemplate jdbcTemplate2(@Qualifier("db2") DataSource ds) {
        return new JdbcTemplate(ds);
    }
}

6. We can use wherever we want :
@.Autowired
@Qualifier("jdbcTemplate1")
private JdbcTemplate jdbcTemplate1;

@.Autowired
@Qualifier("jdbcTemplate2")
private JdbcTemplate jdbcTemplate2;

Q37) ResultSetExtractor vs RowMapper
Ans) RowMapper :
a) It defines only one method mapRow that accepts ResultSet instance and
int as the parameter list
    public T mapRow(ResultSet rs, int rowNum) throws SQLException
b) Row mapper processed per row basis

```

c) It iterates the ResultSet internally and adds it into the collection. So we don't need to write a lot of code to fetch the records as ResultSetExtractor.

Summary :
1) mapRow(ResultSet rs, int rowNumber)
2) It Iterates ResultSet internally and add to the collection
3) Code

ResultSetExtractor:

a) It defines only one method extractData that accepts ResultSet instance as a parameter
public T extractData(ResultSet rs) throws SQLException, DataAccessException
b) Resultset extractor we can navigate all rows and return type is object.

Code for RowMapper :

```
List<Employee> list=jdbcTemplate.query("select * from employee", new RowMapper<Employee>() {  
    @Override  
    public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {  
        Employee e=new Employee();  
        e.setId(rs.getInt(1));  
        e.setName(rs.getString(2));  
        e.setSalary(rs.getInt(3));  
        return e;  
    }  
});
```

Code for resultSetExtractor :

```
List<Employee> list=jdbcTemplate.query("select * from employee", new ResultSetExtractor<List<Employee>>() {  
    @Override  
    public List<Employee> extractData(ResultSet rs) throws SQLException, DataAccessException {  
  
        List<Employee> list=new ArrayList<Employee>();  
        while(rs.next()){  
            Employee e=new Employee();  
            e.setId(rs.getInt(1));  
            e.setName(rs.getString(2));  
            e.setSalary(rs.getInt(3));  
            list.add(e);  
        }  
        return list;  
    }  
});
```

What to choose : RowMapper is simpler choice.

Q38) Abstract Class vs Interface ?

Ans) -> By default variables are public, final and static
-> By default methods are abstract
-> if one abstract method present then lambda can be used (can't do with abstract class)
-> multiple inheritance

Q39) HashCode and equals

Contract : if 2 objects are equals then hashCode must return the same value

Ans_ Student{id, name}

```
HashMap<Student, String> map=new HashMap();
```

```
s1-> 1 pawan
```

```
s2-> 1 pawan
```

```
s3-> 2 ashish
```

```
map.put(s1,"hey")
```

```
map.put(s2,"hey")
```

```
map.put(s3,"hey")
```

HashCode	Equals	Output
100	Overridden	s1,s3
Not Overridden	Overridden	s1,s2,s3
Overridden	Not Overridden	s1,s2,s3
Overridden	true	s1,s3
Overridden	false	s1,s2,s3
1	true	s1 or s2 or s3 any
1	FALSE	s1,s2,s3

Q40) Put VS Patch

Ans) Patch is used when we want to update a "portion" of Object

Put is used when we want to update complete Object

```
### ESa samjo 2 tabs open h ab ek me address update kia or dusre me id  
kia
```

Initially :

```
Person {  
    id : 37  
    name :abc  
    address :XXY  
}
```

Request 1 : to update address -->PATCH use

```
Person {  
    id : 37  
    name :abc  
    address :MNB  
}
```

Request 2 : to update id -->PATCH use (Jab usne page load kia us time address yhi tha or usi time request1 kisi or ne add kri but yaha upi p XXY hi rha)

```
Person {  
    id : 38  
    name :abc  
    address :XXY  
}
```

Note : If we use Put in Above Request2 then Address will also be modified

Q41) Options Method in Rest

Ans) The OPTIONS method in a REST API is used to retrieve information about the communication options available for a resource.

It allows a client to discover the available methods and the communication options that the server supports, such as the supported HTTP methods, headers, and the format of the request and response.
Eg. CORS Preflight Request
Access-Control Allow -Origin

-Headers
-Methods

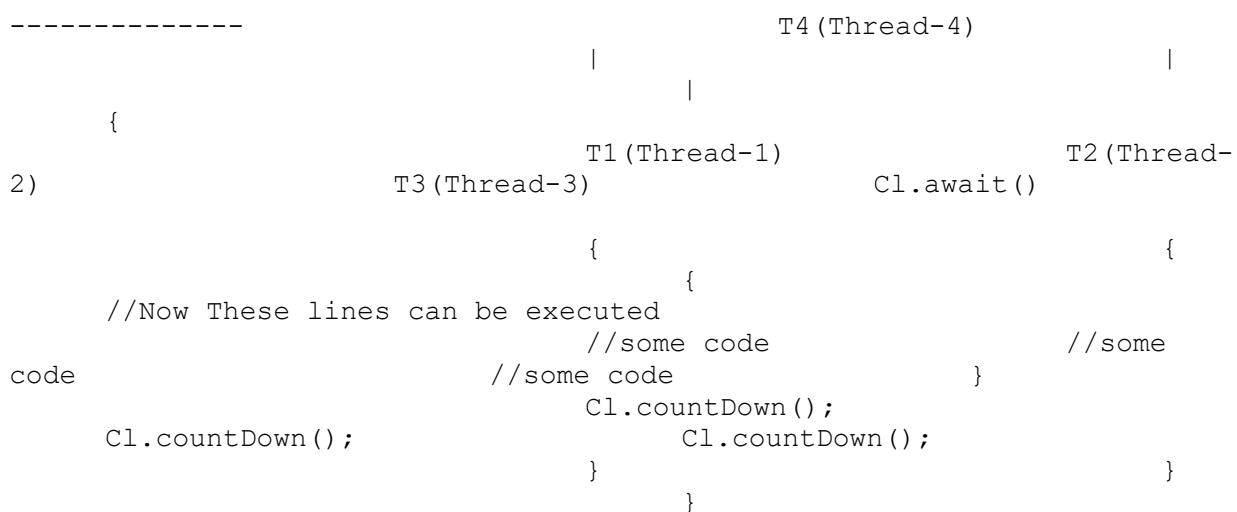
Content Type- Application/JSON

Q42) CountDown Latch

Ans)

CountDownLatch CL=New CountDownLatch (3)
| // CL

object divided amoung all threads



-It is a synchronization way that allows one or more than one thread to wait until a set of operations being performed in another thread is completed.

- It initializes with the count, which we pass to the constructor.
- The invocation of the countDown() method causes the await methods to block until the current count reaches zero, after which all waiting threads are released, and any subsequent invocations of await immediately return.
- CountDownLatch doesn't require that thread countdown() to wait for the count to reach zero before proceeding.
- It prevents any thread from proceeding past and await until all threads can pass.

```
public class DevTeam extends Thread {
    private CountDownLatch countDownLatch;

    public DevTeam(CountDownLatch countDownLatch, String name) {
        super(name);
        this.countDownLatch = countDownLatch;
    }

    @Override
    public void run() {
        System.out.println("Task assigned to development team "
+Thread.currentThread().getName());
```

```

        Thread.sleep(3000);

        System.out.println("Task finished by development team
"+Thread.currentThread().getName());

        //Each thread calls countDown() method on task completion.
*****
        countDownLatch.countDown();
    }
}

public class QATeam extends Thread {

    public QATeam(String name) {
        super(name);
    }

    @Override
    public void run() {
        System.out.println("Task assigned to
"+Thread.currentThread().getName();

        Thread.sleep(2000);

        System.out.println("Task finished by
"+Thread.currentThread().getName());
    }
}

public class AssignTaskManagerTest {
    public static void main(String[] args) throws InterruptedException
{
    //Created CountDownLatch for 2 threads
    CountDownLatch countDownLatch = new CountDownLatch(2);

    //Created and started two threads
    DevTeam teamDevA = new DevTeam(countDownLatch, "dev-A");
    DevTeam teamDevB = new DevTeam(countDownLatch, "dev-B");

    teamDevA.start();
    teamDevB.start();

    //When two threads(dev-A and dev-B) completed tasks are
    returned *****
    countDownLatch.await();

    //Now execution of thread(QA team) started..
    QATeam qaTeam = new QATeam("QA team");
    qaTeam.start();
}
}

43) Cyclic Barrier
Ans) Cyclicbarrier cb =new Cyclicbarrier(3,RunnableImplementation)
          T1                               T2
          T3                               {
          {

```

```

        //Some Code          //Some Code
//Some Code      cb.await();      cb.await();
cb.await();      }
}
}

```

Once all threads arrives cb.await then only RunnableImplementation can be executed

```
Main {
```

```

        System.out.println(Thread.currentThread().getName() + " has
started");

        CyclicBarrier barrier = new CyclicBarrier(4);

        PassengerThread p1 = new PassengerThread(1000, barrier,
"John");
        PassengerThread p2 = new PassengerThread(2000, barrier,
"Martin");
        PassengerThread p3 = new PassengerThread(3000, barrier,
"Joya");
        PassengerThread p4 = new PassengerThread(4000, barrier,
"Sam");

        PassengerThread p5 = new PassengerThread(1000, barrier,
"Pipa");
        PassengerThread p6 = new PassengerThread(2000, barrier,
"Dolly");
        PassengerThread p7 = new PassengerThread(3000, barrier,
"Harman");
        PassengerThread p8 = new PassengerThread(4000, barrier,
"Brad");

        p1.start();
        p2.start();
        p3.start();
        p4.start();

        p5.start();
        p6.start();
        p7.start();
        p8.start();

        System.out.println(Thread.currentThread().getName() + " has
finished");
    }
}
```

```

public class PassengerThread extends Thread {
    private int duration;
    private CyclicBarrier barrier;

    public PassengerThread(int duration, CyclicBarrier barrier, String
pname) {
        super(pname);
    }
}
```

```

        this.duration = duration;
        this.barrier = barrier;
    }

@Override
public void run() {
    try {
        Thread.sleep(duration);
        System.out.println(Thread.currentThread().getName() + " is arrived.");

        int await = barrier.await();
        if(await == 0) {
            System.out.println("Four passengers have arrived so cab is going to start..");
        }
    } catch (InterruptedException | BrokenBarrierException e) {
        e.printStackTrace();
    }
}
}

o/p==>main has started
main has finished
John is arrived.
Pipa is arrived.
Dolly is arrived.
Martin is arrived.
Four passengers have arrived so cab is going to start..
Joya is arrived.
Harman is arrived.
Sam is arrived.
Brad is arrived.
Four passengers have arrived so cab is going to start..

```

44) Explain OutOfMemoryError in Java ?

```

Ans) public static void main(String[] args) {
    String str = "Peter";
    LinkedList<String> linkedList = new LinkedList<String>();
    while(true)
    {
        str=str+" Welcome to India";
        linkedList.add(str);
    }
}

```

Console Error : Exception in thread "main" java.lang.OutOfMemoryError:
Java heap space

Resolution: Increase size of heap memory
: Horizontal or vertical scaling

-This OutOfMemoryError is thrown by JVM When memory cannot be allocated to an object due to lack of memory space where garbage collector has also not freed up the space

-This OutOfMemoryError extends VirtualMachineError Class which indicates that JVM Has run out of memory ...VirtualMachineError extends Error Class

which is used to indicate the serious problem that an application does not catch

45) Fail Fast vs Fail Safe Iterator ?

Ans) Iterator -> is an Interface that provides way to traverse thru collection

ConcurrentModificationException -> is thrown when a collection is modified while being iterated over. This happens when you try to add/remove elements from collection while using an iterator

Fail Fast Iterator :

-it immediately throws ConcurrentModificationException in case of structural modification of collection

-Structural modification means adding/removing/updating the value of element in collection while another thread is iterating over it

-Fail Fast Iterator uses original collection to iterate over elements

-Example are Iterator on ArrayList, HashMap collection classes

Fail Safe Iterator :

-it will not throw ConcurrentModificationException during iteration

-it uses copy of collection to traverse over elements

-require more memory bcoz they cloned the original collection

-Example of Fail safe Iterator is ConcurrentHashMap , CopyOnWriteArrayList

Code Fail Fast Iterator :

```
List<String> fruits=new ArrayList<String>();
fruits.add("apple");
fruits.add("banana");
fruits.add("orange");

Iterator<String> iterator=fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
    System.out.println(fruit);
    if(!fruits.contains("pineapple"))
        fruits.add("pineapple");

}
System.out.println(fruits);
```

O/p => apple

Exception in thread "main"

java.util.ConcurrentModificationException

Code Fail Safe Iterator :

```
List<String> fruits=new CopyOnWriteArrayList<String>();
fruits.add("apple");
fruits.add("banana");
fruits.add("orange");

Iterator<String> iterator=fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
    System.out.println(fruit);
    if(!fruits.contains("pineapple"))
        fruits.add("pineapple");
```

```

        }

        System.out.println(fruits);
o/p=>
apple
banana
orange
[apple, banana, orange, pineapple]

46) input =>List<Character> list=new ArrayList<Character>();
        list.add('a');
        list.add('a');
        list.add('b');
        list.add('b');
        list.add('c');
        list.add('c');
o/p =>{a=2, b=2, c=2}

Ans)
Map<Character, Long>
map=list.stream().collect(Collectors.groupingBy(Function.identity(),Collectors.counting()));
System.out.println(map);

47) Difference b/w CountDownLatch and Cyclic Barrier ?
Ans) Once Barriew is broken it is automatically re-created
          Cyclic Barrier
          |
          CountDownLatch
1.Basic           A synchronization aid that allows a      |A
synchronization aid that allows one
                     set of threads to all wait for each    |or more
threads to wait until a set of
                     other to reach a common barrier point|operations
being performed in other threads completes.

          |
2.Runnable         It has a constructor where Runnable   |It don't have such
constructor
                     can be provided.                      |

          |
3.Thread /Task   It maintains a count of threads      | It maintains
a count of tasks
count

4.Exception       If one thread is interrupted while   |Only "current
thread" will throw InterruptedException.
                     waiting then all other waiting threads|It will not
impact other threads
                     will throw BrokenBarrierException  |

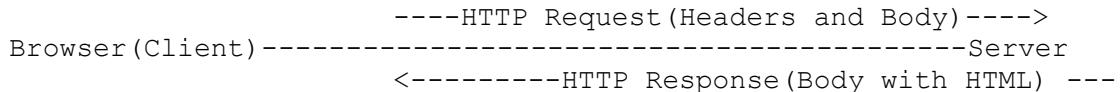
5.Reused          can be reused after holding threads |cannot be reused,
when count arrives at zero it can't be reset.
                     released

48) Explain Rest ?
Ans) REST(Representation state transfer): Transfer data between Client-
Server in format (Json/XML..etc)

1.Rest makes best use of HTTP (How ?? see 2)

```

2.Request and Response are in format defined by HTTP(Hyper Text Transfer Protocol) {Request and Response ..see 3 and 4}
3.Response -HTTP Response comes with HTML that loads on screen
4.Request -HTTP defines headers and body of request
Eg. if we search on google then HTTP request is sent to google server and it comes up with HTTP Response with HTML



-HTTP Methods like GET ,POST ,PUT ,DELETE are available to perform operations
-HTTP Status Code (200, 404...)

REST Guidelines :

- 1.Client-Server Architecture
- 2.Stateless
- 3.cacheable on client side

NOTE: When we want to develop web services with use of HTTP then comes picture of RESTful WebServices comes in

KEY ABSTRACTION -RESOURCE

Resource is something that we want to expose to outside world thru application

- * A resource has an URI (Uniform Resource Identifier)
 - /user/Ranga/todos/1
 - /user/Ranga/todos
 - /user/Ranga
- * A resource has different representations
 - XML
 - HTML
 - JSON
- * Data Exchange Format
 - No Restriction .JSON is popular
- * Transport
 - Only HTTP
- * Service Definition
 - No Standard (eg.Swagger)

Example :Create a user - POST/users
Delete a user -DELETE/users/1
Get All Users -GET/users
Get one user -GET/Users/1

Note : If we want to perform any operations on resource . we will make use of HTTP Methods like GET,PUT,POST,DELETE

48)What is Web Service ? Diff b/w SOAP and REST ?

Ans) WebService helps to communicate between 2 Applications may be Client and Server

Two types of Web Services : SOAP and REST



1.SOAP is protocol
|REST is architectural style

2.full form :Simple Object Access Protocol
|Representational State Transfer

3.SOAP cannot uses REST as it is protocol |REST can
use SOAP web service as it is concept and can use any protocol like http
,soap etc

4.SOAP uses service interface to expose business logic |uses URI to
expose business logic

5.SOAP defines Standard
| No standars

6.More Resources and bandwidth than REST | less

7.permits XML data format
text,HTML ,XML,JSON

8.Slow
|fast

9.SOAp Data cannot be cached at client side | can be
cached at client side
Shortcut : definition,resource and bandwidth ,data types ,speed
,cached...(rdsc)

49) Hash Collision

Ans)Two or more Objectproduce the same final hash value and hence point
to the same bucket location in a HashMap

50) JWT Token Parts ?

Ans)3 Parts : Header ,Payload,Signature

eg. xxxxx.yyyyy.zzzzz

a)Header

The header typically consists of two parts: the type of the token, which
is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.
{

```
"alg": "HS256",
"typ": "JWT"
}
```

b)The second part of the token is the payload, which contains the claims.
Claims is user data

c)Signature

SHA256

51) Which has More priority Setter or Constructor Injection ?

Ans) Setter Injection (Kuki vo constructor bane k baad call hoga tabi
shayad)

52)Sql Query optimization ?

Ans)1.Use Index for Search or create index on columns which are mostly
used in where clause **

2.Use numeric fields to store numeric values instead of character data
types

3.Replace UNION with UNION ALL where we don't have duplicate records

4.Minimum use of DISTINCT keyword

5.when u want to see some table structure having millions of records
avoid using select * from table_name ...instead use

select top(2)* from table_name

5.Dont use != or <> bcoz index will not be used in such case and full
table scan will be performed

6.Use Exists() instead of count() to check if table has specific record
or not

7. Use nvarchar/varchar instead of char .. bcoz trailing spaces are being used in char .. but trailing spaces are not present in nvarchar/varchar

53) char ,nchar ,varchar ,nvarchar data types ?

Ans)

UNICODE :UNICODE is a uniform character encoding standard.

A UNICODE character uses multiple bytes to store the data in the database.

It allows characters from lots of other languages like arabic ,chinese,japanese to be encoded in single character set.

Char, nchar, varchar, and nvarchar are all used to store text or string data in SQL.

char and nchar are fixed-length which will reserve storage space for a number of characters you specify even if you don't use up all that space.

varchar and nvarchar are variable-length which will only use up spaces for the characters you store. It will not reserve storage like char or nchar.

Varchar stores ASCII data and should be your data type of choice for normal use. Regarding memory usage whereas varchar uses 1 byte.

Nvarchar stores UNICODE data. If you have requirements to store UNICODE or multilingual data, nvarchar is the choice.

nvarchar uses 2 bytes per character

NVARCHAR is nothing more than a VARCHAR that supports two-byte characters
varchar(4000)=nvarchar(8000)

CHAR is faster than VARCHAR - some times up to 50% faster.

54) HashMap Improvements in Java 8

Ans) PROBLEM :

For Each Bucket There is Linked List present ..Now for certain scenario where hashCode is not proper or even in normal scenarios ..we can see that at particular bucket index .. there are lots of elements in Linked list ... So now we want to search an element using get() function of hashmap and at same bucket lots of elements are there in Linked list so it will traverse each element and find the result . Time Complexity = $O(n)$

This is bad design / scenario

SOLUTION:

1. Java 8 comes with resolution that HashMap replaces the linked list with another useful data structure i.e. binary tree on breaching a certain threshold, which is known as TREEIFY_THRESHOLD.

2. Linked List is Replaced with "Red Black Tree"

Each Linked List Node is Replaced with "TreeNodes "

Time Complexity = $O(\log(n))$.

3. TreeNodes are nothing but the structures supporting the binary trees which have two nodes, smaller node goes to the left and the larger to the right. "Whenever we want to search for any key the whole left or right subtree is discarded with a single check". This is how the time complexity is reduced to $O(\log(n))$. This change has lead to a significant improvement of HashMap.

4.Once the number of entries is decreased due to removal or resizing of HashMap, the structure is converted back to the older implementation which is LinkedList.

55) Difference between == and equals()

Ans)== checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

56) Concurrency vs Parallelism

Ans) Concurrency : One Cpu and Multiple task
when one task executing on CPU .. the other task will wait for same CPU

Parallelism : More than Multiple CPU and Multiple Task

one task executing on One Cpu and Other task executing on other CPU

57) Convert List to Map (Function Used is - toMap())

Ans) List<Employee> emps = new ArrayList<>();
emps.add(new Employee(1, "Kavi"));
emps.add(new Employee(5, "murugan"));
emps.add(new Employee(2, "Kaaaavi"));
emps.add(new Employee(4, "sss"));

```
Map<Integer,Employee> map= emps.stream()  
.collect(Collectors.toMap(Employee::getId, Function.identity()));
```

```
System.out.println(map);
```

58) find duplicates in List using Java 8

```
List<Integer> list=Arrays.asList(1,2,3,4,4,5,6,6,7,9,9);  
  
List<Integer>  
finalList=list.stream().collect(Collectors.groupingBy(Function.identity()  
,Collectors.counting()))  
.entrySet().stream().filter(e->e.getValue()>1).map(e->  
e.getKey()).collect(Collectors.toList());  
  
System.out.println(finalList);[4, 6, 9]
```

59) Find The Non duplicate Element / Element Occuring Only Once / remove duplicate elements

```
List<Integer> list=Arrays.asList(1,2,3,4,4,5,6,6,7,9,9);  
  
List<Integer>  
finalList=list.stream().collect(Collectors.groupingBy(Function.identity()  
,Collectors.counting()))  
.entrySet().stream().filter(e->e.getValue()==1).map(e->  
e.getKey()).collect(Collectors.toList());  
  
System.out.println(finalList);[1, 2, 3, 5, 7]
```

60) Majority of an Element / Element with highest occurrence in array (One Duplicate with Maximum times)

```
List<Integer> list=Arrays.asList(2,3,4,5,2,3,3,3,3);
```

```

        int
key=list.stream().collect(Collectors.groupingBy(Function.identity(),Colle
ctors.counting()));
    entrySet().stream().max((e1,e2)-
>e1.getValue().compareTo(e2.getValue())).get().getKey();
    System.out.println(key);3

```

61.Q.Why do we need Web API versioning?

Ans.We want to create the new version of application with new requirements + we dont want to disturb the existing running application

Types Of Versioning :

- 1.URI versioning
2. Request Parameter versioning
- 3.(custom)Headers Versioning
- 4.Media Type Versioning(a.k.a "content-negotiation" or "accept header")

Factors Affecting Versioning /DrawBacks /Disadvantages

- URI Pollution (URI versioning and Request Parameter versioning changes uri)
- misuse of HttpHeaders (Headers Versioning and Media Type Versioning .Actually HttpHeaders were never intended for versioning)
- caching (we cannot cache directly Headers Versioning and Media Type Versioning as Uri is same and need to look at headers that increases complexity)
- can we execute request on browser (NO-Headers Versioning and Media Type Versioning ,YES-URI versioning and Request Parameter versioning)
- API documentation(EASY-Headers Versioning and Media Type Versioning ,LIL COMPLEX-URI versioning and Request Parameter versioning)

-No Perfect Solution

For Example -->See how structure is done in versioning package

```

//URI Versioning
@GetMapping("v1/person")
public PersonV1 personV1() {
    return new PersonV1("Bob Charlie");
}

@GetMapping("v2/person")
public PersonV2 personV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}
-----
//Parameter versioning
//http://localhost:8005/person/param?version=1
@GetMapping(value="person/param",params = "version=1")
public PersonV1 paramV1() {
    return new PersonV1("Bob Charlie");
}

//http://localhost:8005/person/param?version=2
@GetMapping(value="person/param",params = "version=2")
public PersonV2 paramV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}

```

```

        }
-----
    //--Customized Header versioning
    @GetMapping(value="person/header",headers =
"X_API_VERSION=1")
    public PersonV1 headerV1() {
        return new PersonV1("Bob Charlie");
    }

    @GetMapping(value="person/header",headers =
"X_API_VERSION=2")
    public PersonV2 headerV2() {
        return new PersonV2(new Name("Bob", "Charlie"));
    }
-----
//--Accept Header versioning OR mime type versioning-->syntax
:application/something+json
//ACCEPT=application/v1+json
@GetMapping(value="person/produces",produces = "app/v1+json")
public PersonV1 produces() {
    return new PersonV1("Bob Charlie");
}

//ACCEPT=application/v2+json
@GetMapping(value="person/produces",produces =
"application/v2+json")
public PersonV2 producesV2() {
    return new PersonV2(new Name("Bob", "Charlie"));
}

```

62. Future Vs Completable Future ?

Ans. (Part Of Concurrency in java means belongs to `java.util.concurrent`)
 Future is introduced in 1.5 version
 Completable Future is introduced in 1.8 version

Future :

1. Future is an interface
2. Future used to create asynchronous tasks.

Drawbacks with Future :

1. Future can't be finished explicitly.
2. Without blocking, we can't do anything else with a Future's result.
3. There is no way to execute multiple Futures in simultaneously and then combine the results.
4. There are no exception handling constructs in the Future.

What is CompletableFuture?

1. CompletableFuture is an class in Java that implements Future interface
2. CompletableFuture resolved issues mentioned with Future(mentioned above)
3. CompletableFuture implements CompletionStage interface through which we can combine the results with other steps

Now, why there is a need to create asynchronous tasks?

- Let us assume that we are creating a website where we have to do some computation in order which means we have to complete Task-1 before Task-

2. With synchronous tasks, our web application will wait for Task-1 to be completed before proceeding to Task-2 where the application could be something else during this wait time.

-In asynchronous design, our application will continue working on tasks that are not dependent on Task-1 to be completed and that's why there is a need for asynchronous architecture.

Methods : Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation.

boolean cancel(boolean mayInterruptIfRunning) -> Attempts to cancel execution of this task.
V get() -> Waits if necessary for the computation to complete, and then retrieves its result.
V get(long timeout, TimeUnit unit)-> Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.
boolean isCancelled() -> Returns true if this task was cancelled before it completed normally.
boolean isDone() -> Returns true if this task completed.

Future Example :

```
public class FutureDemo {  
  
    public static void main(String[] args) throws ExecutionException,  
InterruptedException {  
    Future<String> futureObj =  
    Executors.newSingleThreadExecutor().submit(() -> {  
        TimeUnit.SECONDS.sleep(2);  
        return "Hello World!";  
    });  
  
    while (!futureObj.isDone()) {  
        System.out.println("Result hasn't yet returned");  
    }  
  
    System.out.println("Result -> " + futureObj.get());  
}  
}
```

CompletableFuture Example 1:

```
public class CompletableFutureDemo1 {  
  
    public static void main(String[] args) throws ExecutionException,  
InterruptedException {  
    CompletableFuture<String> futureObj = new CompletableFuture<>();  
  
    Executors.newSingleThreadExecutor().submit(() -> {  
        try {  
            TimeUnit.SECONDS.sleep(2);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
        futureObj.complete("Hello World");  
    });  
}
```

```

        });

        while (!futureObj.isDone()) {
            System.out.println("Result hasn't yet returned");
        }

        System.out.println("Result -> " + futureObj.get());
    }

}

CompletableFuture Example 2:
public class CompletableFutureDemo2 {

    public static void main(String[] args) throws ExecutionException,
InterruptedException {
        CompletableFuture<String> completableFutureObj1 =
CompletableFuture.supplyAsync(() -> "Hello").thenApply(hello -> hello +
"World");
        CompletableFuture<String> completableFutureObj2 =
getHello().thenCompose(hello -> getHelloWorld(hello));

        System.out.println(completableFutureObj1.get());
        System.out.println(completableFutureObj2.get());
    }

    private static CompletableFuture<String> getHello() {
        return CompletableFuture.supplyAsync(() -> "Hello");
    }

    private static CompletableFuture<String> getHelloWorld(String hello)
{
        return CompletableFuture.supplyAsync(() -> hello + " World");
    }

}

```

63. Java Memory management Model before and After Java 8 ?(Before and After Java 8 and Significance Of this Change +Note)

BEFORE JAVA 8:

-->PermGen (Or Permanent Generation)

1. It is memory sapce in Heap that is used by JVM to store class and method objects and intern string

-if ur application has lots of classes ..permgen utilization will be high and we could encounter "java.lang.OutOfMemoryError :PermGen space"

2. Size of PermGen is configured using java command line option

 "-XX" : MaxPermSize

3. (extra) Typically 264 MB memory of permGen should be more than enough for most of application

4.-PermGen is part of Java Heap (Maximum size of Heap(not PermGen) configured by "-Xms" & "-Xmx" option)

AFTER JAVA 8 :

-->Metaspace

-With Java 8 ,there is NO PERMGEN ..So NO OutOfMemoryError due to PermGen -Metaspace is Part of NATIVE MEMORY(PROCESS MEMORY) which is only limited by Host Operating System,METASPACE IS NOT THE PART OF HEAP .

So What is Significance of this Change ?

-While there is no permGen..u may consume excessive native memory making total process size large .If ur application has lots of classes (and /or intern strings), YOU MAY BRING DOWN THE ENTIRE SERVER(not just ur application but other application on same server).Why ?Bcoz Native memory is limited to Operating System and This means u can literally take all memory of Server which is not good. (THIS POINT IS DRAWACK OF METASPACE)
-To handle above mentioned point we should use "-XX:MaxMetaspaceSize" which sets Max metaspace Size of ur application

Note : Before Java 8 u have to keep an eye on Heap Size But After Java 8 we have to keep eye on Heap Size and Process size (Under Task Manager)

64. How Constructor Injection Supports Immutability ?Ans)

It makes the fields immutable by marking them as final, ensuring you cannot accidentally modify the injected dependencies in your code.

65.Why String is made Immutable in Java ? (SST)

Ans)

a) Storage Saving :String variables refer to the same object in the String pool.

-one copy of each literal String in the pool. This process is called interning

b) Security :It increases security because any hacker can't change its value and it's used for storing sensitive information such as a database username or password.

c) Synchronization : it's safe to use in multi-threading and you don't need any synchronization.

immutable objects, in general, can be shared across multiple threads running simultaneously. They're also thread-safe because if a thread changes the value, then instead of modifying the same, a new String would be created in the String pool. Hence, Strings are safe for multi-threading.

66.Concurrent Package in Java

Ans) java.util classes -> fail fast
java.util.concurrent -> failsafe

Concurrent classes introduced in java.util.concurrent pkg

- a) CountDownLatch
- b) CyclicBarrier
- c) CopyOnWriteArrayList
- d) ConcurrentHashMap
- e) BlockingQueue

67) HashMap vs HashTable vs Synchronized HashMap vs ConcurrentHashMap

Ans) | Synchronized | Locking | No Of Threads | Null
| It's a lock | | |
| When-to-use |

HashMap | It is Not Synchronized | No Locking | Multiple Thread | one
Null Key + | Fail Fast | Single Threaded Environment |
| Synchronized |
| Multiple Null Values |

HashTable	All Operations keys and Values	Object Level Fail Safe r Synchronized	One Thread it is Legacy class and lock not recommended to use	No Null
Sync HM Null Key +	All Operations Fail Safe Multiple Null Values	Object Level Multi Threaded r Synchronized	One Thread Multi Threaded lock Environment	one
Concurrent keys and HM Values	Only Write Fail Safe operation are Synchronized	Bucket Level Multi Threaded lock Environment	16 update + unlimited read	No Null

68) Disadvantages or Challenges in REST:

ans)

a) Lack of state: most web applications require stateful mechanisms. Suppose you purchase a website which has a mechanism to have a shopping cart. It is required to know the number of items in the shopping cart before the actual purchase is made. This burden of maintaining the state lies on the client, which makes the client application heavy and difficult to maintain.

b) Lack of security: REST does not impose security such as SOAP. That is the reason REST is appropriate for public URLs, but it is not good for confidential data passage between client and server.

69) Strategies In JPA

Ans)

\-----

SITA (Sequence ,Identity ,Table ,Auto)

1.Sequence -> @SequenceGenerator(sequenceName="tableName")
Table created initial value starts from 1.

2.Identity -> works best with auto incremental column
eg. MySQL

3.Table -> creates table with property value of @TableGenerator
(name="custom name")
2 columns created with property value of ->
"pkColumnName" , "valueColumnName"

4.Auto -> Based on Database it automatically picks b/w Sequence and Identity Generation Strategy

/-----

1.Auto (Default generation Type) : {Pick any from other 3}
AUTO picks any of the other three (IDENTITY, SEQUENCE and TABLE) strategy based on the underlying database

-For MySQL .. Hibernate_Sequence table created in order to maintain auto_increment count

-For Oracle ,PostSql DB ..They have their own sequence mechanism
-Hiberante most pf times use this startegy

2.Identity :.{Depends on Auto_incremenet Column} This generation type is most suited for databases which has an auto-incremented column (E.g. AUTO_INCREMENT feature in MySQL and IDENTITY in SQL server).

- It relies on auto_increment database column and let database generate new value on every insertion

```
@Id  
@GeneratedValue(  
    strategy = GenerationType.IDENTITY,  
    generator = "id_generator"  
)  
private Integer id;
```

3.SEQUENCE : A table will be created with value set for property "sequenceName" in @SequenceGenerator and initial value will start from 1 A database sequence object generates the identifier values.

```
@Id  
@GeneratedValue(  
    strategy = GenerationType.SEQUENCE,  
    generator = "id_generator"  
)  
@SequenceGenerator(  
    name = "id_generator",  
    sequenceName = "id_generator_name",  
    allocationSize = 1  
)  
private Integer id;
```

Once you run your application, you will see a table named id_generator_name will get created and initial value will be 1.
Hibernate: create table id_generator_name (next_val bigint) engine=MyISAM
Hibernate: insert into id_generator_name values (1)

4.TABLE :The @TableGenerator annotation is used to create a table and set the initial value for the id value.

```
@TableGenerator(  
    name = "id_generator",  
    table = "id_gen",  
    pkColumnName = "gen_name",  
    valueColumnName = "gen_val",  
    pkColumnValue = "iden_Gen",  
    initialValue = 10000,  
    allocationSize = 100)  
@Id  
@GeneratedValue(  
    strategy = GenerationType.TABLE,  
    generator = "id_generator"  
)  
private Integer id;
```

70.Why main() is static in Java ?

Ans)The main() method is marked static so that the JVM may call it without having to create an instance of the class that contains the main() function

71.Functional Interface override methods of Object class ?

Ans) Below has overriden method from Object class but it has to be abstract method

```
@FunctionalInterface
public interface DemoInterface {
    void hello();

    @Override          --> Works Fine
    boolean equals(Object obj);

    @Override          -> compilation error
    boolean equals(Object obj) {

    }

}

72.Convert int[] to stream
int[] arr= {1,2,3};
Arrays.asList(arr).stream().forEach(e-
>System.out.println(e.toString())); // [I@1218025c
Arrays.stream(arr).forEach(e->System.out.println(e));// 1 2 3
```

73.Find Output

Case 1: here we are getting Compiled Time Exception toh run toh kya hi krenge program

```
public static void main(String[] args) {
    System.out.print("a");
    try {
        System.out.print("b");
        throw new Exception(); //--> Compiled Exception
    }
    catch (RuntimeException e) {
        System.out.print("c");
    }
    finally {
        System.out.print("d");
    }
    System.out.print("e");
}
```

Case 2: IllegalArgumentException is Runtime Exception and its not catched here

```
System.out.print("a");
try {
    System.out.print("b");
    throw new IllegalArgumentException();
}
catch (NullPointerException e) {
    System.out.print("c");
}
finally {
    System.out.print("d");
}
```

```
System.out.print("e");
```

Output -> abd and exception trace

Case 3:IllegalArgumentException is Runtime Exception and its catched here

```
System.out.print("a");
try {
    System.out.print("b");
    throw new IllegalArgumentException();
}
catch (RuntimeException e) {
    System.out.print("c");
}
finally {
    System.out.print("d");
}
System.out.print("e");
```

74) Remove all values with contains any special character and give count of each value

```
Map<Integer, String> map=new HashMap<Integer, String>();
map.put(1, "Ravi");
map.put(2, "gh65@");
map.put(3, "Ravi");
map.put(4, "Alex");

Map<String, Long> fmap=map.entrySet().stream().filter(e->Pattern.matches("[a-zA-Z]{1,}", e.getValue()))
    .map(e->e.getValue())
    .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));
System.out.println(fmap);

output ->{Alex=1, Ravi=2}
```

75) What will be the output and Why ?

Scenario 1:

```
public class OverloadingExample {

    public static void main(String[] args) {
        someMethod(null);
    }

    public static void someMethod(String message) {
        System.out.println("Hello String");
    }

    public static void someMethod(Object obj) {
        System.out.println("Hello Object");
    }
}
```

Output -> Hello String

```

Scenario 2:
void method(int ... a) // compilation exception at this line
{
    System.out.println(1);
}
void method(int[] a)
{
    System.out.println(2); // compilation exception at this line
}

```

Explanation : In Java, when you have method overloading and you call the method with a null argument, the compiler will choose the most specific method based on the parameter types.

Interviewer:

why String is the most specific in this example

Answer: In Java, when the compiler is deciding which method to call during method overloading, it follows a process called "Compile-time Polymorphism" or "Static Binding."

String is a more specific type

String being a more specific type than Object leads to the compiler choosing the method with the String parameter when you pass null

76) Sort Map by Values in Increasing order using Java 8

```

Ans) Map<String, Integer> wordCounts = new HashMap<>();
      wordCounts.put("USA", 100);
      wordCounts.put("jobs", 150);
      wordCounts.put("software", 50);
      wordCounts.put("technology", 70);
      wordCounts.put("opportunity", 200);

      Map<String, Integer>
sortedMap=wordCounts.entrySet().stream().sorted((e1,e2)->e2.getValue()-e1.getValue())
      .collect(Collectors.toMap(Map.Entry::getKey,
      Map.Entry::getValue,(e1,e2)->e1,LinkedHashMap::new));
      System.out.println(sortedMap);

```

Output -> {software=50, technology=70, USA=100, jobs=200, opportunity=200}

77) ,Comparing() , ComparingDouble() , ComparingInt() ,reversed()....

```

Ans)

class User {
    public String name;
    public double salary;

    //getter & setters
}
```

```

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

```

        User u1 = new User("Aaman", 3500.56);
        User u2 = new User("Joyita", 4222.99);
        User u3 = new User("Suvam", 2832.34);
        User u4 = new User("mahafuj", 2522.76);
        User u5 = new User("Pawan", 2522.76);
        List<User> list = Arrays.asList(u2, u1, u4, u3, u5);

Collections.sort(list, Comparator.comparingDouble(User::getSalary) .
    // thenComparing((e1,e2)-
>e1.getName().compareTo(e2.getName())));
    thenComparing(User::getName));

System.out.println(list);
System.out.println("=====");

Collections.sort(list, Comparator.comparingDouble(User::getSalary).reversed());
    System.out.println(list);
    System.out.println("=====");

Collections.sort(list, Comparator.comparing(User::getName).reversed());
    System.out.println(list);
}

Output =>
[User [name=Pawan, salary=2522.76], User [name=mahafuj, salary=2522.76],
User [name=Suvam, salary=2832.34], User [name=Aaman, salary=3500.56],
User [name=Joyita, salary=4222.99]]
=====
[User [name=Joyita, salary=4222.99], User [name=Aaman, salary=3500.56],
User [name=Suvam, salary=2832.34], User [name=Pawan, salary=2522.76],
User [name=mahafuj, salary=2522.76]]
=====
[User [name=mahafuj, salary=2522.76], User [name=Suvam, salary=2832.34],
User [name=Pawan, salary=2522.76], User [name=Joyita, salary=4222.99],
User [name=Aaman, salary=3500.56]]
```

78) Indexing (reference : <https://www.geeksforgeeks.org/difference-between-clustered-and-non-clustered-index/>)

Ans) Indexing makes data retrieval fast

Index Key : Column on which we create index

Types :

1. Clustered Index :

You can create only one clustered index in a table.

Data is arranged in sorted order based on index created column.

In a clustered index, the index contains a pointer to block but not direct data.

2. Non - Clustered Index :

A Single table can have many non-clustered indexes as an index in non-clustered index is stored in different places stores the data at one location and indices at other location .

The index contains pointers to location of that data .

The non-Clustered Index is similar to the index of a book. The index of a book consists of a chapter name and page number, if you want to read any topic or chapter then you can directly go to that page by using the index of that book. No need to go through each and every page of a book.

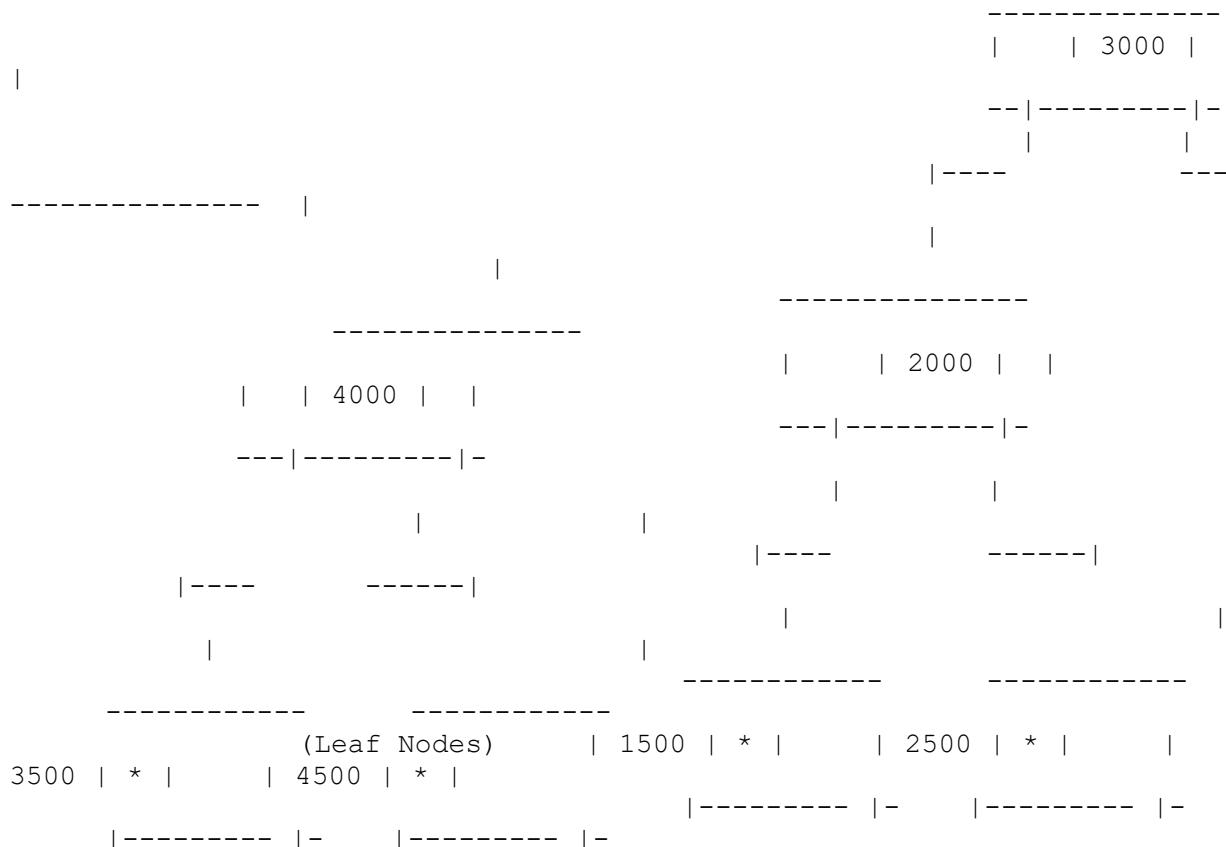
The data is stored in one place, and the index is stored in another place. Since the data and non-clustered index is stored separately, then you can have multiple non-clustered indexes in a table.

When Search is happening in Db it could be of

- a) Table Scan (When index is not there) - does linear search where searches for each and every record
- b) Index unique Scan -index unique scan or scan on primary key
- c) index range scan - where condition has equality or range predicate

Default Index : BTREE

Create Index index_name on Table(Column_name)



Select * from employee where salary=2000

If index not created then takes more time
if index created then by default Btree index created and takes less time
bcoz we can discard one side of tree

When to use Btree?

it is used where we have wide range of data .. almost all columns are unique eg . Employee Number

```
Create index EmpNoIndex on Employee (Emp_no)
```

Index Types :

1. Unique Index

```
Create Unique Index indexName on Table(Column_name)
```

```
eg. Create index EmpNoIndex on Employee (Emp_no)
```

2. Composite Index :

```
Create Index indexName on Table(Column1 , Column2)
```

3. Function Based Index :

```
Create Index FT1 on EMP( UPPER(NAME))
```

4. BitMap Index

It can be used where we dont have wide data eg Gender Column can have only 'Male' , 'Female'

Note :

- 1) Index for primary key and unique constraints are automatically created and dropped during table creation and deletion
- 2) index contains redundant data already exists in table hence consumes more space
- 3) Each table can have only one clustered index usually created on primary key
- 4) No limit on non-clustered index

```
Select * from user_index where table_name ='EMP'  
select * from user_ind_columns where table_name ='EMP'
```

79) How to Create Immutable List

Ans)

```
List<String> mutableList = new ArrayList<>();  
mutableList.add("Apple");  
mutableList.add("Banana");  
mutableList.add("Orange");
```

```
List<String> immutableList = Collections.unmodifiableList(mutableList);
```

```
System.out.println("Mutable List: " + mutableList); --> [Apple, Banana,  
Orange]
```

```
System.out.println("Immutable List: " + immutableList);--> [Apple,  
Banana, Orange]
```

```
immutableList.add("pawan");
```

```
System.out.println("Immutable List: " + immutableList); This will throw  
UnsupportedOperationException
```

80) Collections.singletonMap() ?

Ans) immutable map containing only one mapping

```
Map<String, Integer> singletonMap = Collections.singletonMap("key", 42);  
// Attempting to modify the singleton map will result in  
UnsupportedOperationException  
// singletonMap.put("newKey", 99); // This will throw  
UnsupportedOperationException
```

```

81) Collections.unmodifiableMap()
Ans) singletonMap -> contains single key value
    unmodifiableMap -> contains multiple key value
Map<String, Integer> originalMap = new HashMap<>();
originalMap.put("key1", 42);
originalMap.put("key2", 24);

// Create an immutable view of the original map
Map<String, Integer> immutableMap =
Collections.unmodifiableMap(originalMap);

// Attempting to modify the immutable map will result in
UnsupportedOperationException
// immutableMap.put("key3", 36); // This will throw
UnsupportedOperationException

```

81) Function FunctionalInterface ?

```

Ans) Interface Function<T,R>{
    R apply(T t);
}
T -> represents Input
R-> Result

Function<Integer, Double> half = a -> a / 2.0;
System.out.println(half.apply(10));

```

82) find nth maximum salary

Approach 1:

```

SELECT salary FROM employees ORDER BY salary DESC LIMIT 1 OFFSET n - 1;
OFFSET : specifies the number of rows to skip before starting to return
rows.

```

Approach 2: without using limit

```

SELECT DISTINCT salary FROM employees emp1 WHERE (n - 1) = ( SELECT
COUNT(DISTINCT salary)   FROM employees emp2   WHERE emp2.salary >=
emp1.salary);

```

83) Singleton containing Prototype

Ans) Method -1 :

```

@Component
public class SingletonLookupBean {

    @Lookup
    public PrototypeBean getPrototypeBean() {
        return null;
    }
}

```

Method -2 :

```

public class SingletonObjectFactoryBean {

    @Autowired
    private ObjectFactory<PrototypeBean> prototypeBeanObjectFactory;

    public PrototypeBean getPrototypeInstance() {
        return prototypeBeanObjectFactory.getObject();
    }
}

```

```
}
```

84) DRY Principle ("Don't Repeat Yourself.") means reducing duplication of code

Ways to achieve this :

- Extract Reusable Code
- Use Inheritance and Polymorphism
- Create Utility Classes
- Use Generics
- Automate Repetitive Tasks

85) Guess Output (Interview Zensar)

```
int a = 10;
int b = 20;

int c = 30;

int res = (a > b) ? (a > c) ? a : c : (b > c) ? b : c;

System.out.println("java----- " +res);
```

Ans) java----- 30

86) Explain Distributed Transaction | Transaction Management in Microservices ?

Ans) -Transaction spread across multiple microservices .

-"coordination of transaction" ensures that changes will be "committed together" to all microservices or not committed at all

In Monolithic Application we have :

```
"Single Transaction"
{
Create order();
ValidatePayment();
DeliverOrder();
OrderMarkedSuccessful();
}
```

73.Linked List ?

Ans) -Variable Size

-Non Contiguous Memory

-insert in O(1)

-Search in O(n)

=====Odd Even Thread Printing=====

```
public class MainClass {

    int counter=1;
    static int n;

    public static void main(String[] args) {
        n=10;
        MainClass poe=new MainClass();

        Thread t1=new Thread(new Runnable() {
            public void run() {
```

```

                poe.printOddNumber();
            }
        }, "Thread-1");

        Thread t2=new Thread(new Runnable() {
            public void run() {
                poe.printEvenNumber();
            }
        }, "Thread-2");

        t1.start();
        t2.start();

    }

    public void printOddNumber() {
        synchronized (this) {
            while(counter <n) {
                if(counter%2==0) {
                    wait();
                }
                System.out.println(Thread.currentThread().getName()+""
"+counter);
                counter++;
                notify();
            }
        }
    }

    public void printEvenNumber() {
        synchronized (this) {
            while(counter <n) {
                if(counter%2!=0) {
                    wait();
                }
                System.out.println(Thread.currentThread().getName()+""
"+counter);
                counter++;
                notify();
            }
        }
    }
}
*****SAGA DESIGN PATTERN *****
Distributed Transaction : A transaction which spans across multiple
microservices
```

Swiggy => Order -> Payment -> Delivery
-->Monolithic Application

```

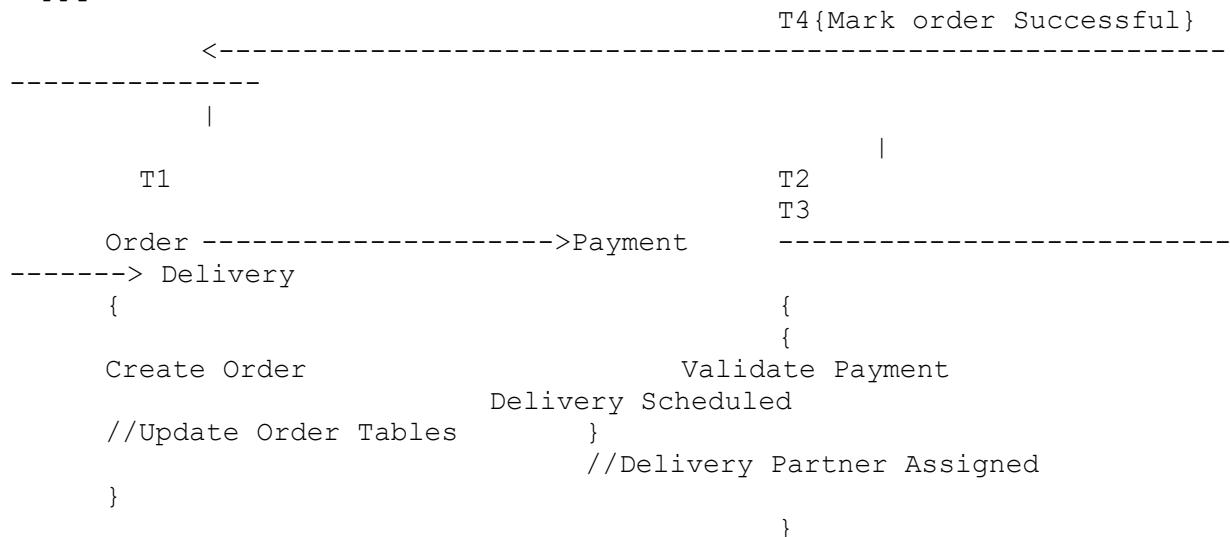
Single Transaction (@Transactional )
{
Create Order
Validate Payment
Deliver Order
Order Marked Successfull
}
If Anything Fails then complete Order is Rolled Back
```

--> Microservice Application

- 1.Order Service
- 2.Payment Service
- 3.Delivery Service

T1,T2,T3,T4 are transactions happening in different microservices

Happy Scenario :



Failure Scenario :

Suppose Transaction T1 and T2 completed but not T3 Failed as No Delivery Partner is available so we need to reverse T1 AND T2 also Here Saga Design pattern helps to resolve the issue

-T3 can be rolled back as exception occurred in Delivery Service .But We need to roll back transaction of Other Microservices (Order and Payment Microservice)

Q) What is Saga ?

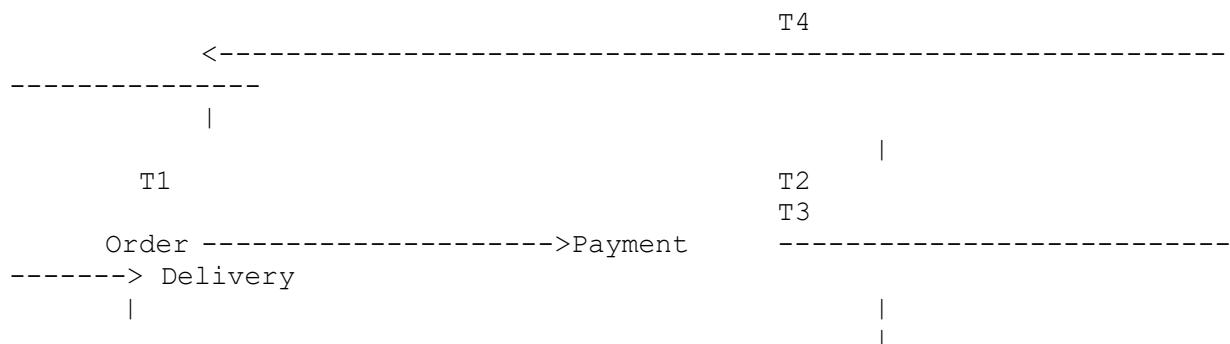
Saga : It is sequence of local transactions

Each Saga has 2 Jobs : Update the Current Microservices and make required changes

Publish Events To trigger next transaction of next microservices

In above Example T1,T2,T3,T4 are saga transaction

So Suppose if T3(Delivert Transaction) Fails it will roll back the changes on previous microservices



```
sequenceDiagram
    participant A
    participant B
    A->>B: Cancel Order Event
    activate B
    B-->>A: Revert Payment
    deactivate B
    B-->>A: Delivery Failed
    deactivate B
```

The diagram illustrates a sequence of events between two participants, A and B. It begins with participant A sending a 'Cancel Order Event' to participant B. Participant B then sends a 'Revert Payment' message back to participant A. Finally, participant B sends a 'Delivery Failed' message to participant A.

1.Choreography /Event Based : The first service executes a transaction and publish the event

This event is listened by one or more services which executes local transaction and may /may not publish an event

Happy Scenario :

Order Service ---Publish-> \$\$ Order Created Event \$\$ ---Listen-> Payment Service--Publish--> \$\$ Validate Payment Event \$\$

Listen

Order Marked Successful <---- \$\$ Delivery Partner
Assigned Event\$\$ <--Publish---Delivery Service

Failure Scenario :

```

sequenceDiagram
    participant DS as Delivery Partner Not Available Event is
    participant OS as Order Service
    participant PS as Payment Service
    OS->>DS: ---Publish->
    DS-->>PS: $ Order Created
    PS-->>OS: ---Listen->
    activate DS
    activate OS
    deactivate DS
    deactivate OS

```

The sequence diagram illustrates the communication between the Order Service and the Payment Service. The Order Service initiates a publish action, which triggers the Payment Service to listen for the event. The event is then published, and the Payment Service responds with a payment confirmation.

publish

Validate Payment Event \$\$

5

Listen

|

|

V

Delivery Service

|

Listen

Publish

|

V

\$\$

Delivery Partner Not Available Event\$\$ -----
