Interview Questions

1. Convert Array to ArrayList — Arrays.toList();
2. Sort the elements in Array list using comparator or comparable interfaces —> both are interfaces, comparable need to be implemented by pojo class while comparator is implemented by eg. Age comparator class(sorting on multiple elements is possible)
3. Sort the elements in Hash Set (Not possible because hash set does not guarantee order)
4. Working of hash set internally -internally uses hash map to store data
5. Difference b/w hashMap and treeMap? —> ordering is missing in HashMap =, which is included in TreeMap, In TreeMap keys can be ordered using comparator or using comparable interface. linkedHashMap - insertion order is maintained.
6. Can you override object class methods in interface? - No it will give compile time error.
7. Aggregation vs composition (weak Association & strong association)
8. ClassNotFoundException vs classDefNotFoundError (ClassNotFoundException is an exception that occurs when you try to load a **class** at run time using **Class**. forName() or loadClass() methods and mentioned **classes** are **not found** in the classpath. **NoClassDefFoundError** is an error that occurs when a particular **class** is **present** at compile time, but was **missing** at run time)
9. Why String class is defined as Final (To make it immutable)
10. Difference b/w Immutable collections & unmodifiable collections (modified collections change impact unmodified collection (unmodifiable), modifiable collections change doesn’t impact unmodified collection (immutable))
11. How internally Collections.sort() works
12. How internally Concurrent Hash map works?
13. Why Java 8 came & agenda behind it ? (Significant reason is to introduce conciseness to the code, enables functional programming by lambda expressions, to compete with languages like python & Scala java upgraded itself from oops to FP to create concise code base)
14. Features of Java (Lambda Expressions, Stream API, Default and Static methods in interface, Functional Interface, Optional, Method references, Date API, Nashron JS Engine)
15. Advantages of Java 8 — Compact Code (less boiler plate code), readable & concise code, testable code & Parallel operations
16. What is Lambda Expression -> Lambda Expression is a anonymous

function with out name, return type, & access modifiers and having only one symbol () -> {}

17. What are Functional Interfaces -> Interfaces with one abstract method, it can have any number of static & default methods, (eg: comparable, Runnable, comparator)
18. How Lambda expression related to Functional interfaces (Function interface is used to provide reference to lambda expressions)
    1. Comparator<String> c = (s1, s2) -> s1.compareTo(s2)
    2. (Function interface). = (lambda expression)
19. Can we create own Fountain interface -> yes we can create by annotating class with FuntionalInterface annotation and declaring only one abstract method & no restriction on default , static methods
20. Is Function Interface annotation mandatory —> not necessarily, but we add it, compiler will throw errors related to it incase we do any mistakes
21. Method referencing : replacement of lambda exp -> Method referencing is a replacement for lambda expression, code reusability & used to refer method of functional interface to an existing method of any class.
22. What are default methods ? —> Java 8 default methods are way adding new methods to interface without effecting implementing classes. Backward compatibility is achieved.
23. Is it necessary to override default methods -> no not required, if the implementing class is not ok it can override
24. How to override default method -> by removing default key key word, as default keyword is not used in class but only in switch case
25. Can we override object class methods in interface using default method —> no we can’t - it throws compile time error
26. How default methods in interface cope up with diamond problem? -> If two implemented interfaces are having same method diamond problem will occur & code will not compile in such case. Solution is use InterfaceName.super.methodName();
27. Why static methods in interface ? —> No need to create a class and object to access a method. We can just call methods using interface name.
28. Are static methods available to implementing classes by default ? —> Static method in interface are not available for implementing interfaces, we need to explicitly call those static methods from implementing class methods.
29. What are predicates -> predefined functional interface (only one abstract method in predicate is Test(T t) return type is boolean)
30. What is predicate joining -> three ways to join OR, AND, Negate -> used for checking multiple condition
31. What are Functions -> predefined function interface -> R apply (T t) -> accepts one input returns one output

32. Difference b/w predicate & function
33. Functional chaining
    1. f1.andThen(f2).apply(input) - first f1 then f2
    2. f1.compose(f2).apply(input) - first f2 then f1
    3. F1.andThen(f2).andThen(f3).apply(input) - function chaining
34. Consumer Function interface —> just consumes input and don’t return anything & no compose method here in consumer
35. Supplier Function interface —> its just return but never consume input (No chaining is required in supplier functional interface because we don’t need input)
36. Use of BiConsumer, BiFunction, BiPredicate & why no BiSupplier -> used in case of two inputs
    1. Predicate<T> test() -> return boolean
    2. Function<T, R> apply() -> returns anything
    3. Consumer<T> accept() -> returns nothing
    4. Supplier<R> get() -> returns anything
37. If we want to works on three inputs —> Java Functional interfaces supports only 1 or 2 inputs not more than that.
38.
39. How to create immutable objects in Java ? — make fields private final, no setter methods, class to make final
40. Is Java pass by value or pass by reference? - according to java doc java is always pass by value, though reference is passed it is passed as value by copying the address in bit by bit
41. What is the use of finally block ? — finally block is used to do clean up codes
42. why there are two date classes ? Util data give both time and date, sql date gives only date
43. Explain Marker interfaces ? —> Interface with no methods ex- serializable - adds metadata at runtime
44. Why main method is static public void ? — can be access from any class, no object required to call method
45. Difference between new String() and String literal ? —String str1 = “Uday” -> creates object in string pool, new String(“Uday”) -> create object both in string pool and heap memory
46. How does substring method internally works? — Sub string does not actually reduce size, instead offset value and count will be changed
47. Explain the working of HashMap? — decide the bucket based on hashcode and equals method to check whether the object already exists
    1. Both should be overridden, if two objects are equal by equals method —> hashcode should be also equal. Bucket capacity, index = hashcode(key) & (n-1) n =16(default bucket size)
48. Difference b/w interface and abstract class? - Abstract classes can have

some common implementation

49. When do u override equals and hashcode methods? When we use that object to store as hash map keys
50. Why finallize() method should be avoided? — finalise method is invoked before an object is garbage collected. If implemented manually leads to o severe performance issue.
51. Why hash map should be avoided in multi threading env ? Can it cause invite looping? - because its not synchronous
52. Explain encapsulation and abstraction ? How are they related — encapsulation means binding state and behaviour together as a capsule and abstract is hiding the implementation and showing only functionality.
53. How StringBuffer save the memory? Stringbuffer will not create new instance of string each and every time instead it increases it size for each change
54. Why wait and notify declared in Object class instead of thread?— In java there are not special Classe to be used as a shared resource, any object can be shared so these methods are declared on Object class
55. Write a java program to create dead lock and fix it? — both objects having lock on two different resources and waiting for each other thread to release lock on their resources.
56. Serializable class contains field when doesn’t need to be serial, how to fix it? - transient key word and @transient JPA annotation- to avoid it to be persisted in database.
57. Explain transient and volatile annotations? Volatile - used in case of multi threaded env if a field is volatile -> each thread works directly on the memory of the field instead copy of the object.
58. Difference b/w iterator and list iterator? — iterator can iterate all collections but list iterator only list, iterator can only remove element but list iterator can add modify remove.
59. Deep copy and shallow copy - shallow copy in general, deep copy is objects complex object is also copied.
60. Synchronisation, class level and object level locking - allowing only one thread to execute method. Object level - each time a thread can work on a instance where as class level is irrespective of multiple instance each time only a single instance is executed at once
61. Difference b/w sleep() and wait() — sleep is of thread class wait is of object class, sleep method pauses the execution for a while and does not release lock on resource while sleep and put back thread to runnable state — wait method in turn immediately releases lock and used for inter thread communication
62. Difference b/w & and && & bit wise operation and && logical operation right is executed only if left is true
63. Explain all access modifiers — private public default protected

64. What is garbage collections and how to enforce it? system.gc()
65. What is native keyword - used to have some native code
66. What is serialisation and explain its catches? Converting java object in to byte stream
67. Checked Exceptions - IO Exception - FileNotFoundException, MalFormedUrlException,
68. UnChecked Exceptions - Null pointer exception, arrayindexboundException, StringIndexBoundofException, NumberFormatException, Arthictmatic exception, clascastexception, illegalArgumentException, IllegalStateException

1. Is String keyword in Java? — no its not keyword
2. Why are strings immutable? — because they are more commonly used so they save lot of memory
3. What is String constant pool? — where all string are stored
4. Keyword 'intern' usage — used to return interned string returns value from string pool
5. Matching Regular expressions?
6. String comparison with equals() and '=='? — equals compare only data where as == compares both data and reference
7. Memory leak issue in String class — substring issue
8. How does String work in Java? —
9. What are different ways to create String Object?
10. How to check if String is Palindrome.
11. How to remove or replace characters from String.
12. How to make String upper case or lower case?
13. How to compare two Strings in java program?
14. Can we use String in the switch case?
15. Write a program to print all permutations of String?
16. Write a java program to reverse each word of a given string?
17. How to Split String in java?
18. Why is Char array preferred over String for storing password? -- Because once the password is used it is still present in string pool before gc and can't be overdone with junk data
19. Is String thread-safe in Java — yes as it is immutable
20. Why String is popular HashMap key in Java - it is immutable and no need to generating hashcode again and again.
21. Difference between String, StringBuffer(synchronised & thread safe) and StringBuilder (not thread safe)?
22. How to concatenate multiple strings. str1.concat(str2)
23. How many objects will be created with string initialization code? - 2
24. How do you count the number of occurrences of each character in a string?

25. Write a java program to reverse a string? Using getBytes of a string and create a new byte array and store data from last, using string builder reverse method

1) What is the Java Collections API? List down its advantages? - List, Set(sortedSet), Queue and Map interfaces(sortedMap), ArrayList, LinkedList, Stack, Vector, HashSet, TreeSet, HashMap, LinkedHasMap, TreeMap.
2) Explain Collections hierarchy?
3) Why Collection interface does not extend Cloneable and Serializable interface? - clone able and serialisable are specific functionality and need to be available for all the classes.
4) Why Map interface does not extend Collection interface? - they way map stores data is different and it adding and retrieval process.
5) Why we use List interface? What are main classes implementing List interface? - array list, linked list, vector, stack
6) How to convert an array of String to ArrayList? - Arrays.asList();
7) How to reverse the list? - collections.revers(list)
8) Why we use Set interface? What are main classes implementing Set interface? , HashSet, TreeSet
9) How HashSet store elements?
10) Can a null element added to a TreeSet or HashSet? In hash set yes but in tree set can't store because tree set internally using navigable hash map.
11) Why we use Map interface? What are main classes implementing Map interface?
12) What are IdentityHashMap and WeakHashMap? - weak hash map - GC dominate over weak hash map and in hash map - hash map dominates over gc
13) Explain ConcurrentHashMap? How it works? Used in multi threaded env
14) How hashmap works? -
15) How to design a good key for hashmap? String - because no need to caluculating dashcode and its immutable
16) What are different Collection views provided by Map interface? Entry Set(), key Set,
17) When to use HashMap or TreeMap? - if we need the elements to be in sorted manner treetop is required
18) Difference between Set and List? Set- stores unique values and not insertion order guaranteed.
19) Difference between List and Map?
20) Difference between HashMap and HashTable? Hash map non-sync hash table - sync
21) Difference between Vector and ArrayList? Vector uses iterator or enumerator.
22) Difference between Iterator and Enumeration? Enumerator (legacy vector and hashtable)- read only access where as iterator used for read as well as

remove lements.
23) Difference between HashMap and HashSet?
24) Difference between Iterator and ListIterator?
25) Difference between TreeSet and SortedSet?
26) Difference between ArrayList and LinkedList?
27) How to make a collection read only?
Collections.unmodifiableList(fruitList);
28) How to make a collection thread safe?
Collections.synchornizedList(),map(),set()
29) Why there is not method like Iterator.add() to add elements to the collection? It might corrupt the underlying data structure of collection as each collection uses different algorithms to store elements
30) What are different ways to iterate over a list?
31) What do you understand by iterator fail-fast property? - fail fast — leads to concurrent modification exception when a collection structure is modified while traversing and fail safe does not throw exception because it works in a copy of collection
32) What is difference between fail-fast and fail-safe?
33) How to avoid ConcurrentModificationException while iterating a collection? Collection.synchornizedList(list) conn=current collections classes
34) What is UnsupportedOperationException? List list = Arrays.asList(); list.add() will throw an exception (Arrays.asList , List.of({1, 2,3})) return unmodifiable collections
35) Which collection classes provide random access of it's elements? Array list, HashMap, TreeMap, HashTable
36) What is BlockingQueue? can not dequeue an empty queue until an element is added
37) What is Queue and Stack, list their differences? Fifo lifo
38) What is Comparable and Comparator interface?
39) What are Collections and Arrays class?
40) Recommended resources

1. What is Spring Framework? What are it's main modules? AOP, IOC, Dependency injection, beans, core, context, spring web , spring data access layer.
2. What are the benefits of using Spring Framework? Light weight , inversion of control, aop, container, mvc framework, transaction management, exceptional handling
3. What is Inversion of Control (IoC) and Dependency Injection? - Instead we creating the beans ioc take care of creating the beans in the way we define in xml files
4. Explain IoC in Spring Framework?

5. Difference between BeanFactory and ApplicationContext?
6. In how many ways, you can configure Spring into our application? - xml file, annotations, java based configuration
7. What is Spring XML-Based Configuration? - configurations is defined in xml files
8. What is Spring Java-Based Configuration? - configuration is done by adding @bean annotation on methods
9. What is Spring Annotation-based Configuration? Configuration is done by @configuration, @service, @controller, @component
10. Explain Spring Bean lifecycle? -
11. What are different Spring Bean Scopes? - singleton, prototype, session, request, global session
12. What are inner beans in Spring?
13. Are Singleton beans thread safe in Spring Framework? - No not thread safe
14. How can you inject a Java Collection in Spring? Give example?
15. How to inject a java.util.Properties into a Spring Bean? Using props element in bean config file or using and manually loading using properties class
16. Explain Spring Bean Autowiring? Auto wiring wires up all the required beans in component and configure them
17. Explain different modes of bean autowiring? No, byType, byName, constructor and autodetect
18. How do you turn on annotation based autowiring? <context:annotation-config/>
19. Explain @Required annotation with example?20. Explain @Autowired annotation with example? - required annotation is added on top of setter method to tell that this bean is mandatory. The **@Required** annotation applies to bean property setter methods and it indicates that the affected bean property must be populated in XML configuration file at configuration time. Otherwise, the container throws a BeanInitializationException exception.

21. Explain @Qualifier annotation with example?
There may be a situation when you create more than one bean of the same type and want to wire only one of them with a property. In such cases, you can use the **@Qualifier** annotation along with **@Autowired** to remove the confusion by specifying which exact bean will be wired. Following is an example to show the use of @Qualifier annotation.
22. Difference between constructor injection and setter injection? Constructor injection is used to inject all the required beans in a component where as setter injection is used to inject optional beans.

23. What are the different types of events in spring framework?

24. Difference between FileSystemResource and ClassPathResource?

25. Name some of the design patterns used in Spring Framework?

Describe Spring AOP? Spring Aspect oriented programming is used to implement the cross cutting concerns of an application. Cross cutting concerns are the one which are found across the modules.
What is the difference between concern and cross-cutting concern in Spring AOP? Concern - specific to module and ccc - present across all modules
What are the available AOP implementations? Xml schema based and @AspectJ annotation based
What are the different advice types in spring?
What is Spring AOP Proxy?
What is Introduction?
What is joint point and Point cut?
What is Weaving in AOP?
1. What is Spring boot? How it is different from Spring framework? Springboot is for rapid application development and comes up with an opinionated view by having auto configuration feature enabled and auto configurng the default present in class path along with having embedded server and hot deployment.
2. Advantages and disadvantages of spring boot? Disadvantage - heavy weight
3. What is auto-configuraion? How to enable or disable certain configuration? @enableAutoConfiguration , using exclude attribute we can specify any bean to be auto wired
4. What are starter dependencies? Starter dependencies comes up with all the related dependencies as collection and example we have spring web starter includes tomcat son servlet etc etc
5. Spring boot common annotations? @springbootApplication, @Configuration, @component, @Service, @Repository, @query
6. Embedded server support in Spring boot? Yes default tomcat 8080
7. Why we use spring boot maven plugin? Works with springboot in packaging the application as jar or war.
8. How to create and bootstrap a simple boot application?
10. How to enable debug logging? logging.level = debug.
11. What is Spring Actuator? What are its advantages? Acutators are to monitor and manage the application in runtime. We can control the actuators by enabling the desired one like /env, /beans, /status, /health etc and can use all together different port for it.
12. What is relaxed binding in Spring boot? Srping Boot starter dependency will be resolving the compatible version of a required dependency.
13. How to unit test and intration test a spring boot application?
14. How to enable hot deployment and live reload on browser? Dev tools

15. How to enable HTTPS/SSL support in Spring boot? No sure
What is Spring MVC framework? Model view controller
What is DispatcherServlet and ContextLoaderListener? Dispatcher servlet is the one which handles all the http request
What is the front controller class of Spring MVC?
How to use Java based configuration?
How can we use Spring to create Restful Web Service returning JSON response? @Response Body @Controller or @RestController
Can we have multiple Spring configuration files?
Difference between <context:annotation-config> vs <context:component-scan>? Enables annotations and enables component scan
Difference between @Component, @Controller, @Repository & @Service annotations?
What does the ViewResolver class?
What is a MultipartResolver and when its used?
How to upload file in Spring MVC Application?
How does Spring MVC provide validation support?
How to validate form data in Spring Web MVC Framework?
What is Spring MVC Interceptor and how to use it?
How to handle exceptions in Spring MVC Framework?
How to achieve localization in Spring MVC applications?
How to get ServletContext and ServletConfig object in a Spring Bean?
How to use Tomcat JNDI DataSource in Spring Web Application?
How would you relate Spring MVC Framework to 3-tier architecture?

1. How you will design a good key for HashMap?
2. Difference between HashMap and ConcurrentHashMap?
3. Difference between HashMap and Collections.synchronizedMap(HashMap)?
4. Difference between ConcurrentHashMap and Collections.synchronizedMap(HashMap)?
5. Difference between HashMap and HashTable?
6. Difference between HashTable and Collections.synchronized(HashMap)?
7. Impact of random/fixed hashCode() value for key?
8. Using HashMap in non-synchronized code in multi-threaded application?

---

1. What is hibernate ? How does it work - Hibernate is a ORM framework which maps java entities with tables in. Database. Queries work on entity classes instead of db
2. Hibernate architecture — configuration, session factory, session, query, first-level cache, transaction, persistent objects, second level cache
3. Difference b/w JPA and hibernate ? How can we swap out of hibernate

and use other orm framework

4. Difference b.w session load and session get? Load method load data from cache and it element not present in db throws exception
5. Difference b.w merge and refresh ?
6. Difference b.w persistent entity and transient entity ? Newly created object is in transient state and entity associated to session is in persistent state
7. Difference b.w save and saveOrUpdate method?
8. Persistence life cycles Transient state, Persistent state, Detached State
9. Casading types ? Cascade.ALL, Cascade.
10. Lazy Loading?
11. Criteria query —The Criteria API allows you to build up a criteria query object programmatically;
12. Named queries ?
13. Cascade types ?
14. HQL vs SQL
15. Cache - first level cache second level cache
16. Cardinality mappings : @oneToOne, @oneToMany, @manyToMany
17. A detached entity can not be deleted in jpa but its possible in hibernate
18. Hibernate is default spa provider of jpa data spring
19. Hibernate - persist, save, saveOrUpdate, merge, refresh, update
20. Persist vs save (save is actual hibernate method), save has return type and persist does not, persist method can not save a detached entity but save method can save detached entity, but creates new entry
21. We can call update method on only persistence object not transistor first session.save(person), session.evict(person), modify object session.update(person)
22. Merge method - The main intention of the *merge* method is to update a *persistent* entity instance with new field values from a *detached* entity instance. Gives third(merged) object.
23. SaveOrUpdate can be used against transient object or persistence object
24. Update method can transition detached object to persistent state, when used on transient object throws error
25. Refresh() - get data from database and update the persistence entity.

1. What are micro services and what the advantages and disadvantages?
    1. Developments becomes easy — easy for new team members to quickly start working
    2. Testing and deployment becomes easy - no need to test complete flow and no complete deployment
    3. Cost effective - since we only need to increase load on intended

    micro service
    4. Help full for new team members.
2. Spring cloud components
3. Non functional requirements of micro services —
    1. Service discovery - Eureka naming server
    2. Load balancing - ribbon
    3. Fault tolerence - hystrix
    4. Easy integration - feign client
    5. Cross cutting concerns
    6. Distributed tracing - zipkin and slueth

Design patterns : Design patterns are solutions for common problem and which fully tried and tested solutions and language natural

1. Creational
2. Structural
3. Behavioural

1. Creational
    1. Factory - single interface, multiple classes implementing it and a factory class which accept string and return with any of the implantation objects
    2. Abstract Factory - Abstract factory producer, ProfessionAbstractFactory, TranieeAbstractFactory, —> implements AbstractFactory and
    3. Singleton - at any time only one instance of object is returned
    4. Builder - Director class(manages the object as well as return the constructed object), Home class, builder interface, EarthQuakeResHomeBuilder, FloodResistanceHomeBuilder.
    5. ProtoType -Pojo class should implement cloneble interface and have method which will return cloned object (super.clone())

Exceptional Handling:

1. Throwable class
2. Exception and Error
3. Un checked Exception vs checked exception
4. Checked Exception - IO Exception, SQL Exception
5. Un checked Exception - null pointer, array out of bound , number format exception, athematic exception

Lambdas -

    1. Allows us to enable functional programming
    2. Supports parallel programming
    3. Readable and concise code
    4. Easier to use and libraries
1. Why default methods in interfaces in java 8 - > to provide backward compatibility so that existing interfaces can use lambda expressions with out implementing in implementation class
2. In general methods in interface are abstract methods (these method won't have definition only declaration) and we can add default methods with java 8
3. Java 8 introduced a new class Optional in JDK 8. It is a public final class and used to deal with NullPointerException in Java application.
4. Functional Interfaces - > java.util.function package
    1. Biconsumer -> takes two arguments -> accept(T t1, T t2)
    2. Predicate -> take one arg and return boolean -> test(T t)
    3. Function -> takes one arg and return argument -> apply(T t) -> function chaining
    4. Consumer -> takes arg that's it -> accept(T t) -> consumer chaining
    5. Supplier -> will not take any input but return R
    6. Streams -
5. Difference between streams and collections
6. What does map() function does? - map function converts one object in to other object
7. What does filter() does —> it filters out elements based on condition by calling predicate method
8. Difference b'w map and flat map?
9. Difference b/w intermediate operations & terminal operations?
10. What is peek in streams -> to print debug message while using lambda expression
11. Why stream is lazy (till terminal operation is specified then only it will work)
12. What is functional interface-> Interface with only one abstract method and it can contain many default & static methods (Runnable, callable, comparator, comparable, supplier, consumer, function, predicate, Bifunction, Biconsumer, BiPredicate)
13.

Failed questions —>

1. Java 8 features
2. Implicit type casting (refer all core java concepts)
3. Lambdas and Streams

cloud interview

1. Benefits of clouds
2. Clouds services types iaas, paas, saas
3. AWS Benefits - Operation Excellence, Security, Reliability, better performance and cost optimization.
4. AWS S3 upload File - PutObject, GetObject, DeleteObject
5. AWS S3 upload large file - MultipartUpload - used when file size is more than 5 GB
6. AWS s3 - cap of the largest file - 1 TB
7. AWS S3 pre-signed urls - makes the object available public for upload or download for a specific amount of time
8.

Questions which require more then one line answer

1. What is Spring Framework and its components

Ans: Spring is an open source development framework for enterprise Java. Spring comes with features like light weight, inversion of control, AOP, Container, transaction management.

1. What is Spring IOC

Ans: The Spring IoC creates the objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction. The Spring container uses dependency injection (DI) to manage the components that make up an application.

1. What is Spring Dependency Injection
2. What is Spring Aspects
3. Which DI would you suggest Constructor-based or setter-based DI?

   Ans: Since you can mix both, Constructor- and Setter-based DI, it is a good rule of thumb to use constructor arguments for mandatory dependencies and setters for optional dependencies. Note that the use of a *@Required* annotation on a setter can be used to make setters required dependencies.

What is Dependency Injection?
This concept says that you do not create your objects but describe how they should be created. You don't directly connect your components and services together in code but describe which services are needed by which components in a configuration file. A container (the IOC container) is then responsible for hooking it all up.
What are the benefits of IOC?
Ans: t minimises the amount of code , Loose coupled, IOC containers support eager instantiation and lazy loading of services.

Difference b/w @bean and @configuration
Ans: @Configuration tells spring IOC to create a bean out of this class and

@bean tell the this method gives us a bean which need to be registered in spring application context.

Data Structures and Algorithms

1. Data structures are nothing but organising the data in to memory
2. Primitive and non primitive data structures
3. Non primitive - linear(list, stack, array, queue) and non-linear (tree, graph)
4. Tree - binary search binary tree search
5. Linear search(normal search comparing one by one) vs binary search (sort the elements and do search operations)
6. Sorting - bubble sort, insertion sort, quick sort, merge sort

Threads:

1. What is thread and process
2. How to create thread

Concurrent collections:

1. Difference b/w concurrent collections and synchronized ? Ans : Synchronized classes are not good in terms of performance because of wide-locking mechanism
2. Collections in single threaded app and concurrent collections in multiple threaded app
3.

Angular preparation :

1. Angular is Java script open source framework by google for developing single page web application
2. It is entirely different from Angular js by following component based, set friendly, mobile app friendly, using typescript and provided with cli to fasten development.
3. Angular App Modules —declarations, imports, providers, bootstrap
4. Mani components of Angular - Component, Service, Module, Directives,
5. Built in Directives - attribute directives - NgClass, NgStyle, NgModel — structural directives — *NgFor, *NgIf, *NgSwitch-ngSwitchCase-ngDefaultCase
6. @component decorator for component, @injector for service, @NgModel for modules (appModule, RouteModule)

7. Forms - Reactive Forms vs Template driven forms
8. Template driven forms are for simple forms, ideal for very basic requirement and logic that can be maintained in template.
9. Reactive form on other hand gives direct access to forms object model. More robust than template, scalable, reusable, and testable.
10. Routing - AppRoutingModule - created by @NgModule decorator by importing RouterModule, Routes and have const routes = [] where we need to specify path and component
11. HttpClientModule register in AppModule, HttpClient inject in service and make this.http.get(url).subscribe —Because the service method returns an Observable of configuration data, the component *subscribes* to the method's return value.
12. Requesting a typed object type from http call - his.http.get(this.configUrl); —> this.http.get<Config>(this.configUrl);
13. Form element to create form <form #f="ngForm"></form>
14. https://dzone.com/articles/what-is-a-single-page-application
15. Angular - Angular is java script data binding framework which is used to develop single page applications using concepts of routing. It also has lot of features like HTTP, DI etc.
16. Difference B/w Angular JS and Angular - Language (js & ts), Architecture(controller , component), Mobile Compliant (No(), Yes), CLI(No, Yes), Lazy Loading(No, Yes), SEO(No, Yes), Server Side (No, Yes)
17. What is a directive in Angular? Directive is the one which changes the behaviour HTML DOM
18. Types of Directives? Structural, Attribute and Component
19. Examples of Structural -> NgFor, NgSwitch
20. Examples of Attribute —> NgModel, hidden, ngIf
21. Component directive - Directives with template and its a user control
22. Importance of NPM and Node Modules —> NPM is node package manger which manages the dependencies and node modules is the folder which contain all the modules
23. Pacakage.json File - is a file which contains all the references of java script frameworks and rather installing one package we can install multiple packages
24. Type Script - Type script is superset of javascript and it is strongly typed and provide object oriented programming environment which transpires into javascript
25. What is Angular CLI - 1pm install @Angular/cli - ng new my app -> creates a scaffolf/strcuture of angular project with app component, app module, main.ts, etc etc
26. Difference b/w Component(@Component) and Module(@NGModule) -> Component is the one which bind data b/w view and model where

module logically groups the components.

27. Template is the html view of angular where we write directives and there are two types of template like inline and separate file
28. What is data binding - Nothing but how view and component communicate with each other
    1. expression or interpolation {{}} - data flows from component to view
    2. Property binding - [] - same as above - its get attached to user input.
    3. Event binding -() - click, —data flows from view to component
    4. Two way data binding — [()] - ngModel — data flows in bi directional
29. Explain architecture of Angular —
    1. Template
    2. Component
    3. Modules
    4. Bindings
    5. Directives (changes the behaviour of HTML DOM)
    6. Services (contains common logic which used across the porject)
    7. DI (Inject the dependency instead and instancing the object in code which leads to tightly coupling)
30. SPA - Single Page Application - Application where main UI get loaded once and needed UI get loaded on demand
31. How to Implement SPA - using Angular Routing and defining the routing collections. const routes = [{ path':', component: HomeComponent}, {path:'login', component:LoginComponent}]
32. Router Outlet and Router Link — > Router outlet is the place where the dynamic component is loaded on clicking the router link and if from component - using this.route.navigate(['/home'])
33. Lazy Loading - on demand loading - loading only necessary html, css and java script files to have better performance.
34. How to implement lazy loading - Divide project in to modules and use load children in routes to load on demand
35. Define Services
36. What is dependency injection and how to implement - its a application design patterns where rather than creating object instances with in the component Angular injects it via constructor - its implemented by using providers property in NgModule decorator providers:[{provide: BaseLogger, useClass: HttpLogger}] - decoupling
37. Ng serve - ng built —> ng server builds in memory and ng build builds in to hard disk - list folder
38. Ng build —prod — > do all possible minification, remove all comments and other necessary things and gives out a most compressed in to a single js file
39. Explain ViewChild and ViewChildren?
40. Why do we need Template reference variables?

41. What is ContentProjection?
42. Explain Content projection Slot?
43. What is ContentChild and ContentChildren?
44. ViewChild vs ViewChildren vs ContentChild vs ContentrChildren?
45. Explain the importance of Component life cycle ?
46. Explain events and sequence of component life cycle ?
47. Constructor vs ngOnInit() ?
48. How to make HTTP calls using Angular ?
49. What is the need of Subscribe function ?
50. How to handle errors when HTTP fails ?
51. How to pass data between components ?
52. Explain importance of input, output & event emitters ?
53. How to pass during routing ?
54. Is it a good practice to pass data using services ?
55. What is the need of Angular Pipes?
56. Can you name some built-in Angular Pipes?
57. How to create Custom pipes in Angular?
58. Whats the full form of RxJs?
59. What is the purpose of RxJs?
60. What are observables and observers?
61. Explain the use of Subscribe with sample code.
62. How to unsbscribe in RxJs?
63. Explain concept of operators with sample code.
64. How to install RxJs?
65. Differentiate between promise and RxJs?
66. In Angular where have you used RxJs?
67. Which operators have you used from RxJs?
68. What is Push/reactive vs Pull/Imperative?

SQL vs NO SQL

1. No SQL - Highly flexible and scalable document structure
2. No SQL - faster due to efficient indexing and storing techniques
3. SQL - what is indexing in database

MIcro service spring cloud

1. Zulu Gateway
2. Eureka server - service discovery
3. Hystrix - faut tolerance
4. Client side load balancing - feign client with ribbon
5. Declarative Rest client - feign
6. Slueth & zipkin - distributed tracing and dash board

7. Spring could config bus

1. Why String is best suited for HashMap - Because string comes with Hash-code and no need to calculating hash-code every time

Micro service vs monolith :

Advantages of monolith :

1. If team is small, we don't afford time to break up this in to pieces.
2. Deployments are easy
3. Code for Setting up connections, setting up other various things need not be duplicated (less duplication)
4. Calls are faster as we don't need to make RPC calls

Disadvantages:

1. If a new member joins, if he to understand more context to get started
2. Deployments are complex as we need to do deploy whole application even for small fix (frequent deployments are required) each deployment whole functionality need to be tested
3. Too much dependency on a single server, due to any mistake a single server crashes, all other services get crashes (single point of failure)

Micro services:

1. Micro services are easier to scale
2. New developer steps in - He only requires to know the context of single micro service in which he works
3. Parallel development becomes easy ()
4. If there are more load on any micro service , we can scale it alone
5. Deployment flexibility, technology flexibility, can be scaled separately

Disadvantages:

1. Difficult to design micro services
2. If a micro service is only talking to a single micro service —(there is no need of having two micro services)

Micro Services Design Patterns:

1. Decomposition Patterns
2. Integration Patterns
3. Database Patterns
4. Observability Patterns
5. Cross Cutting Concerns Patterns

Architectural level questions

1. How micro services need to splitter up and when to use micro services architecture
2. Importance of logging and how to debug a production issue
3. What is virtualisation and containers.
4. What is the role of docker and why we need it
5. What does Kubernete’s does?
6. What is load balancing and its algorithms
7. Deployment Architecture for a micro service architecture
8. When to use no sql and sql and what is the difference?
9. Object oriented programming vs procedural programming
10. Query execution process steps
11. How exactly JVM works and memory mapping for objects
12. Sass, pass, pass difference with example
13. 5 strong reasons why we need to use cloud services
14. What is a CI CD pipeline
15. ER diagram and how to design db architecture
16.

Java Script:

1. Why we need strict keyword? --> JavaScript in strict mode does not allow variables to be used if they are not declared.
2. Difference between let and var and const—> Before ES2015 JavaScript did not have **Block Scope**.Variables declared with the let keyword can have Block Scope.
3. What is java script hoisting —> JavaScript only hoists declarations, not initialisation
4. What is Ajax and what before Ajax and how it is asynchronous.
5. Difference BOM and DOM?
6. What is a single page application mean and why angular best suits for it why not java script?
7. Variable var can and let be redeclared where as const can not
8. Block scope variables — let and scope
9. Const -It does NOT define a constant value. It defines a constant reference to a value.
10. Window can not use globally declared let
11. Redeclaring a let variable with let, in the same scope, or in the same

block, is not allowed

12. Variables defined with let are hoisted to the top of the block, but not initialized
13. Variables defined with var are **hoisted** to the top and can be initialized at any time
14. Variables defined with const behave like let variables, except they cannot be reassigned

Working of java script:
Java script is a scripting language, loosely coupled, object based , synchronous and single threaded language.
Javascript program contains all functions and variables
Javascript runs in a execution context, which contains memory allocation and code execution
Memory allocation assign memory for variables as well as for functions
Code execution executes code line by line and for each function invocation a new execution context is created and deleted post completion of the method
All these execution contexts are maintained by Execution Context Stack or program stack, control stack, runtime stack &. Machine stack.

Docker :

1. What is virtualisation ? - virtualisation comes with an optional host os of hardware and all vm comes with a guest OS (virtualisation happens on hardware level)
2. What is containerisation? (In case of containers , on top of host OS we have some thing called container engine, which isolate each container and each container have its own dependencies and library and app) - simply OS level virtualisation
3. Bare metal virtualisation - no host OS on hardware
4. Best of both = Vm's + containers - hardware —> hypervisor —> operating systems(vm's) - each vm having containers - virtualisation happens at both ends and efficient utilisation of hardware and packing of application and seamlessly move from one system to other
    1. Difference b/w vm and containers
        1. Startup time - laughing a virtual machine take lot of time (minutes), but containers in milliseconds (boot up time less)
        2. Memory - more memory - less memory , container runs on top of kernel which is light weight
        3. Disk space - more space —less space
        4. Portability - vms comes with diff os, lib, dependencies but container are seamless, consistent exp and portable
        5. Efficiency - not efficient, very efficient (easily bootup) to run at

scale
6. OS /kernel - VMS have dedicated and containers share the kernel
5. Virtualisation - hypervisor is installed on top of OS - type 2 hypervisor — type 1 hypervisor directly installed on hardware
6. Difference b/w containers and traditional deployments - installation, software dependencies, packaging , shipping , isolation, scalability

Azuga :

1. Difference b/w POST and Put Mapping API?
2. What happens if we don't override equals method but override hashcode method - if only hashcode overridden -> selecting buckets works well but "abc" & "ABC" will be present in same bucket - because default equals checks the reference equality -> . If only equals method overridden —> hashcode method select different buckets for objects which are equal by equals method. (Equals object appear in different buckets)
3. Do we something like integer constant pool
4. Find out the least repeated character in string
5. Find out the second least number in a array of integers
6. Print the ascii values of a string.
7. How to find the middle number in a linked l
8. Ways to create object in java -
    1. new key word
    2. Implementing clonal interface and override clone method-> calling clone method of object will create a clone of object
    3. newInstance() method of Class class- > Class.forName(fullyqulalifiednameofpackage).newInstance();
    4. newInstance() method of constructor class -> Constructor<Employee> constructor = Employee.**class**.getConstructor(); , Employee emp3 = constructor.newInstance();
    5. Using Deserialization —> convert json string to object

Spring Security:

1. Spring security is kind of a security guard who always ask who are you and what do you need?
    1. Handles common vulnerabilities like session fixation, clickjacking & clickSiteRequestForgery
    2. User name password Authentication, SSO(outa, ldap), App level Authorization, Intra app authorisation like OAuth, micro services security & method level security

3. Authentication, Authorization, Principaal, Principaal, Granted Authority & Roles.
4. Knowledge passed authentication - easy to implement -
5. Possession based authentication (text/phone, key cards, access token device)
6. K based authentication possession based auth (multi factor)
7. Principal - is currently logged user.
8. Granted Authority - roles, permission, users, groups.

1. What is database indexing? - Indexing is used to optimise the performance of database by minimising the no of disk access required when a query processed.
2. Types of Indexing
    1. Primary - index table contain index as well as pointer value of actual data and main table (contains actual data)
        1. Dense - (used when the primary key is not in sequential)
        2. Sparse - (used when primary key is sequential)
    2. Clustered - indexing done on non unique key (departmentId in employee table) (diff deep employee sits in different blocks)
    3. Secondary - two level indexs—> 1, 101,201 —> 1,11,21,32. ->>>>1,2,3,4,5,6,7,8,9.10,11,12,13...
3. What are spring actuators
4. What are methods present in session class

SQL Queries :

1. SQL Query to find second highest salary of Employee
2. SQL Query to find Max Salary from each department.
3. Write SQL Query to display the current date
4. Write an SQL Query to check whether date passed to Query is the date of given format or not
5. Write an SQL Query to print the name of the distinct employee whose DOB is between 01/01/1960 to 31/12/1975.
6. **Write an SQL Query to find an employee whose Salary is equal or greater than 10000**.
7. Write an SQL Query to find name of employee whose name Start with ‘M’
8. SELECT YEAR(GETDATE()) as "Year"

1. Abstract class and interface
2. @table annotation mandatory

3.

    1. Instance block, static block, constructor
    2. Bean validation - validation-api dependency
    3. More deeper in to controller class generics and return type
    4. Global level exceptional handlers and controller advice
    5. @bean annotation for custom beans — @service annotation - general class.
    6. Complete picture of class object in java —> class, constructors, instance block- for each instance, static block - executes one.

JVM Architecture:

1. What are the components involved in JVM
2. Difference between JVM JDK & JRE
3. Memory Management of JVM —> Stack , Heap, Native Method stack, PC Register
    1. Heap space divided in Young generation & old generation
    2. Permgen - comes with default size 32 mb 84 mb, can be increased and GC is not very efficient & stores static content and metadata required by JVM
    3. Java 8 introduced Meta Space
    4. The biggest disadvantage of PermGen is that it contains a limited size which leads to an OutOfMemoryError
    5. Pergen completely removed in Java 8 and meta space was introduced instead of it with efficient GC.
4.

EY Interview Questions :

1. Find the first non repeating character in a string
2. Find the most repeating string in a text file
3. Find longest substring with out repeating characters
4. Explain the deployment methodology of micro services
5. Explain internal working of concurrent hash map and hashMap in java & what was new feature added in java for hash map
6. Explain about micro services architecture
7. Features of micro services
8. How micro services work
9. What is docker, how it is used along with micro services
10. Disadvantages of micro services —> network latency, logging,
11. Role of offset in Apache Kafka

12. Role of zookeeper in Apache Kafka
13. Requirements of class to be called as JPA Entity class
14. Criteria API and how it is used
15. How does Caching works
16. What is the role of Architect or engineer when it comes to micro service Architecture
17. What is the weakness you have noticed some Developers have and that you don't have.
18. What does programs to interface but not to implementation means?
19. What is perm gen or perm generator.
20. How do you ensure that the code you are writing adapts to the company's growing application needs.
21. JVM Architecture along with JVM Memory Architecture
22. 12 factor app. -> twelve factor app is a methodology for building software as service.

12 factor App:

1. Code Base : Single code Repository—> many deployments.
2. Dependencies -> Explicitly declare and isolate dependencies
3. Config —> configurations should not be part of code —vary between environments
4. Backing Service —> database. Messaging queues, SMTP server —> (**The code for a twelve-factor app makes no distinction between local and third party services**)
5. Build-> Release -> Run — no codes changes (build — contains only code—> release contain both cod and config —> runtime actually runs the release)
6. Processes ->
7. Port Binding —> Export services via port binding
8. Concurrency —>
9. Disposability —> minimal startup time and graceful shutdown by refusing any request on port and completing the current request and exit
10. Dev/Prod Parity—> Need to keep dev, stag & prod as similar as possible
11. Logs —>
12. Admin processes

1. How to avoid singleton class being cloned —> Implement cloneable interface and throw clone not supported exception in clone method
2. Method overloading vs method overriding —> difference —> method overloading -> arguments can change, arguments data type can change, return type can't change for the same method arguments

3. How to do serialisation in java — using ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutPutStream("file.txt")) , outputStream.writeObject()
4. How to do deserialisation in java.
5. Ways to break singleton design pattern —> using clone method, de-seralisation and serialisation and using reflection Class.class.getDeclaredConstructors.setAccessible(true), constructor.newInstance(); , (serialisation —> by overriding readResolve method to return singletonInstace will make singleton more strong)
6. How to avoid class from inheritance — by declaring it as final —> other solution are there —> need to explore
7. What is AOT ? AOT vs JIT
8. Difference b/w rest service and micro service
9. Put mapping vs post mapping
10. How concurrent hash map works

————————oops————

1. Abstract class over super class : If we are not using super class in code, then its better to use abstract class so by we can reduce the instance creation of base class.
2. Composition vs inheritance : https://www.journaldev.com/1775/multiple-inheritance-in-java
3.

DuCont Interview

1. **Javascript typescript Angular**
2. **Java Spring Spring boot**
3. **Sql and No Sql**
4. **AWS (s3, Ec2, lamda, Api Gateway, IAM, DynamoDB, Elastic bean stack, SNS, SQS, SES, RDS, cloud watch)**

Cloud watch - performance monitor
Cloud trail - monitor API calls in the was platform and used for auditing
AWS Config - records the state of your aws env and notify the changes

Java - core java features, oops, exceptional handling, collections, servlets, jdbc ,hibernate, spring springboot. — Completed
No Sql vs mysql - differences, queries, joins, cardinality

Javascript, Angular - have a look at the Angular and js interview questions and create a sample Angular application and see all the features
AWS - concentrate on Lambda, api gateway, Dynamo DB, S3, SNS, SQS, SES, RDS, IAM

Oops Concepts:

1. **Abstraction**
2. **Encapsulation**
3. **Inheritance**
4. **PolyMorphism**
5. **Association**
6. **Aggregation - weak association Bank and Employee association**
7. **Composition - strong association Library and Books association**
8. **Class and Object**

Why Composition is favoured over inheritance ?
Abstract class vs Interface ?
Implicit Type casting vs explicit type casting — Animal animal = new Dog();
Dog dog = animal //throws error saying can't convert parent to child.

---

1. What logging frame work is used -> how logs are stored in your application

---

1. Practice simple programs
2. Lambda Expressions, functional interfaces
3. Refresh all the interview questions
4. Micro services Design pattrens & java design patterns
5. S3(Encryption)
6. AWS Cloud interview questions on lambda, RDS, DynamoDB, code commit, code deploy, code pipeline