1. S3
2. dynamodb
3. smssqs
4. singleton -> single object at a time(e.g Configuration manager, logging service, connection pool manager)
5. factory method -> using interface and also define factory class as well
6. static method vs instance -> class level method , Object level method
7. interface impl
8. SOLID principle -> Single respo, Open for extention Close for modifi(interface use),Liskov(inheritance),Interface segregation(divide interfaces in small interfaces),Dependency inversion(Depend on abstractions, not concrete implementations.)
9. static method ko dusre side call kar shakte hai kya ->using Class name because that static method is known as class method
10. hashmap deeply
11. thread lifecycle ->new,runnable,running,wait,timed wait,terminated
12. MYSQL (joins,group by) select orders.name, shipping.id from orders join shipping on order.id=shipping.orderId;
13. callable and runnable -> Concurrent-java.lang,call()-run(),return value future object-void method
14. checked and unchecked -> Compile time(IOException,SQLException,FileNotFoundException),Runtime(Nullpointer,ArrayIndexOutofBound)
15. try catch(Exception) catch(IOException) error(Order block error)
16. try catch(IOException) catch(Exception) valid
17. try finally valid
18. try only not valid
19. What is String? -> sequence of character
20. immutable? -> string pool which is in heap.
21. linkedlist vs arraylist
22. array vs arraylist
23. what is concurrent collection?-> concurrent thread access(ConcurrentHashMap,CopyOnWriteArrayList)
24. how hash map work?
25. static method can not override -> Static method associate with class not subclass or subtype so if static method overriding is actually called method hiding.
26. final class(is accessible from other class with any way) -> yes but not extends
27. @Entity @Repository @Table :-> HQL- create query need entity and for create table need @table.
28. Spring performance(Connection pool, cache, lazy initialization, optimise db queries, reduce garbage collection(use premitive))
29. Inheritance(parent can call child class ref, but child can not refer parent class)Class B extends A{},B b=new A() not allowed
30. Stream API ->filter,map,collect,foreach,sort
31. why we use strem API over traditional for loop? -> readability and more features like filter and map.
32. java 8 feature? Optional, stream, method reference, for each method
33. hash code -> int value of object in java, hashmap

34. java 8 to 11 feature diff
35. JDK(kit) -> JRE(code compile convert to byte code) -> JVM(execution of code)
36. Comparator vs comparable -> use case vs natural order
37. Generate nullpointerexception program. String s= null, s.toString();
38. Thread safe singleton -> create while class loading(use static and private constructor)
39. Stringbuffer vs stringbuilder (thread safe vs not thread safe,less effi vs more effi)
40. design pattern-> singleton, factory,
41. bitwise operator -> & | (1001 & 1011 = 1001)
42. how cache works?? -> store in memory(hardware,software,network any),database cache, in-memory cache,web cache
43. redis cache vs normal cache -> provide vast features like sorting, queue, searching etc. also store in key value pair.
44. AWS(s3,sqs,sns)
45. what is flatmap -> 3D or 2D array to convert 1D array for better performance
46. @Transactional how it is working -> at class level or method level we can define to control transaction flow. also we can undo and commit transaction.
47. method reference basic rule and def -> static->Classname::method ,instance::instanceMethod,Constructor->Classname::new,
48. what is actuator -> for checking health, heapdump, thread dump and manything of service
49. Static variable is not allowed to use as a local variable
50. what is use of clonable interface in java? -> to clone() use to copy of object without disturb original object
51. public class Tast10 {
52.
53.   public static void main(String[] args) {
54.
55.   float f=increment();
56.   f++;
57.   System.out.println(f);
58.
59.   }
60.
61.   static float increment()
62.   {
63.   static float f=1.5f;
64.   return f;
65.   }
66. }
67.
68. public class Tast10 {
69.   public static void main(String[] args) {
70.
71.   int k=1;
72.
73.   if(k)
74.   {

```
75.             System.out.println("this is true");
76.         }else {
77.             System.out.println("this is false");
78.
79.         }
80.
81.     }
82.
83. }
84. public class Tast10 {
85.
86.
87.     public static void main(String[] args) {
88.
89.          A a= new B();
90.
91.         if(a instanceof A)
92.         {
93.             System.out.println("this is A");
94.         }
95.         if(a instanceof B){
96.             System.out.println("this is B");
97.
98.         }
99.         if(a instanceof Object){
100.                System.out.println("this is Object");
101.
102.          }else{
103.                System.out.println("Exception");
104.            }
105.         }
106.
```

107. write a sql query to find the employee name and total salary from employee table and salary table where in salary table two column one is salary and another is variable and quert should return sum of both in final result
108. How concurrent hashMap works in java? -> multiple thread lock map and work parallel. 16 at a start
109. What are the annotation you have used in spring Boot?
110. How to implement pagination at controller level in spring boot application -> JPA feature or use logic
111.
112. About your past experience- discussion based on that
113. Microservices - Can eureka be used as a load-balancer -> no we will use ribbon client for this
114. Core basic
115. Write abstract class -> have abstract and non abstract method.
116. override - He will give samples, and decide, Is that valid
117. overload - He will give samples, and decide, Is that valid

118. Design patterns
119. Which kind of design pattern have you used in real time?
120. Swapping program
121. @FiegnClient ->eg:@FeignClient(name = "user-service", url = "http://user-service-api-url") // Specify the name and URL of the target service
    1. @GetMapping("/users/{userId}")
122. User getUserById(@PathVariable("userId") Long userId);
123. About experience and explain your last project
124. Why do you use spring boot and spring data JPA (mentioned in the resume) -> to better performance, standalone app, easy to use, DB access, more feature
    a. boiler plate code provide by jpa, less complexity then jdbc.
125. How microservice communicate with each other? Why Rest Template? (In my project we were using Rest Template not Eureka)-> @feign client
126. Employee Management database design -> how you will create tables with columns? What will be primary key and foreign key?
127. create table table_name(colum1 datatype,etc, primary key(ID),Foriegn key (dept ID) reference table_name(dept ID);, emp id primary ket, dept id foreign key
128. Total 3 barrels are there with capacity of
    a. 6 ltr,
    b. 3 ltr
    c. 1.5 ltr
    d. You need to fill a 6 ltr barrel with condition that we can use mini barrels (3 and 1.5 ltrs) only to fill one 6 ltr barrel. What are the minimum steps we can fill the barrel?
129. How will you explain database to 8 years old child?
130. Core java – what is abstract class? Write one abstract class with one abstract and one normal method
131. Queue -> What is RabbitMQ? How it works? What is difference between topic and queue? (mentioned RabbitMQ in resume)
132. What is collection in java? How sorting works in collection? When to use comparator and when to use comparable?
133. Why hashcode and equals method in Object class? What is the contract between hashcode and equals method?
134. If a requirement document (BRD) given to you, how will you start with development?
135.
136. Swap two variable’s value without using third variable.
137. Given an array, we need to find elements whose occurrence is greater than or equal to half length of an array.
138. Which design patterns you used in your previous organization, explain them.
139. What do you know about binary tree? Swap left and right node of complete binary tree.
140. Design question on Employee class with its designation’s details, how would you design Employee class and corresponding tables.
141. How would you improve searching operation on above table?
142. How would you create indexing? -> create index index_name on table_name (column1,2,...)
143. Query to print all the employees belonging to same department in above design.

144. Linked list program. Node which accept simple and decimal numbers.
145. Find length of linked list with recursion.
146. ELK stack use in microservices.
147. What is microservices.
148. Difference in spring mvc and spring boot.
149. challenges I faced in spring mvc configuration like dependency compatibility issues.
150. Write a program which take integer as input and print following output.
    a. If you pass 5, the output should be 15. (5+4+3+2+1)
    b. Program should contain all validation. Minimum limit and maximum limit of number.
    c. What if I pass Integer.MAX_VALUE in this program? What will be the result and why?
151. Car parking problem. Car should be parked in valid empty space.
152.
153. What is your Life's Goal?
154. One test suite is there which has 50 test cases. I am scheduling that test suite. It will take 5 hours to run. Now service is unavailable in middle of execution. How to resume pending test cases?
155. How can I get my custom alert when my service is down? actuator
156. How can I get my custom alert when my database is down?
157. How can I get my custom alert when my service is up but database is down?
158. Current project explanation and counter questions. (In current project, I told him that we are using 3rd Part APIs)
159. If you are using any 3rd party APIs in your project and that API is not responding, then how to handle it? ->circuit breaker pattern and contact to third party
160. What if 3rd party API is not providing proper response? OR What if 3rd party API is providing 500 error code?
161. How you have implemented authentication in microservice?
162. What is AWS?
163. In my CV -> Achievements section -> Best Team (Q2, 2018-2019 T-Mobile Digits) award is there
164. Why do you think your team is the best team and why only your team got this award?
165.
166. Brief about previous organization like employee strength, domain
167. Why did you change previous organization?
168. How many years of experience in spring boot?
169. How's your experience with spring boot? (explain advantages)
170. Have you face performance issues in project? How you deal with?
171. How string works? How its immutable?
172. About OutOfMemory error. How to get it programmatically?
173. Why we get OutOfMemory error even we have garbage collector.
174. Do you know sorting algorithms? How merge sort work? Which one is better for N elements? (Complexity)
175. How to analyse larger log file in production?
176.

177. Web Service for Employee records. How you get records based on location and how do you handle if records are more than 1000?
178. If front end team said, we are facing issue in random bases then what is your approach to resolve same?
179. What is docker? How you used in your project? (As its mentioned in my resume)
180. What is EJB, how its work and how its different from spring boot? (As its mentioned in my resume)
181. Oracle functions and how you used in project? (As its mentioned in my resume)
182. What you know about microservices?
183. Implement link list with insertion and display. Create loop of link list with skipping head, write function to identify loop.
184.
185. Palindrom
186. a. Solution:
187. public static boolean isPalindrome(String str){
    i.
    ii. char[] word = str.toCharArray();
    iii.
    iv. int start = 0;
    v. int end = word.length - 1;
    vi.
    vii. while (end > start) {
        1.
        2. if (word[start] != word[end]) {
            a. return false;
        3. }
        4. start++;
        5. end--;
    viii. }
    ix. return true;
b. }
188.
    a. Above method will work fine for general Strings like (“12321”, “malayalam”), then they asked to modify the above method so that it should work by ignoring letter cases and special characters also, e.g. above method should return true for string “Malayalam” and “Madam, I’m Adam!!!”. They told me to check for alphabets only, so “12321” will not be considered as palindrome. So some modifications can be done in above method and final method will be as below:
189.
    a. public static boolean isPalindrome(String str){
        i.
        ii. char[] word = str.toLowerCase().replaceAll("[^A-Za-z]+", "").toCharArray();
        iii.
        iv. int start = 0;
        v. int end = word.length - 1;
        vi.

```
        vii.    while (end > start) {
                    1.
                    2. if (word[start] != word[end]) {
                            a. return false;
                    3. }
                    4. start++;
                    5. end--;
        viii.   }
        ix.     return true;
    b. }
190.
    a. After this, they expected to create a test method for this functionality, e.g. assertTrue method implementation for palindrome, and solution is below method:
191.
    a. public static void assertTrue(String input, boolean expected, boolean actual) {
        i.
        ii.     if(actual != expected) {
                    1.
                    2. System.out.println("Test case failed for string : "+input);
        iii.    }
    b. }
    c. And above method can be called for testing as below:
192.
        i.      assertTrue("Malayalam",true,isPalindrome("Malayalam"));
193.
    a.
194. Problem 2: Create Employee class with just 2 fields i.e. id and firstname?
195. Solution:
    a. class Employee{
    b.
        i.      private long id;
        ii.     private String firstName;
196. }
197. Now, they asked that what if I don't want to create getter and setter methods for these fields, so straight forward way will be making all fields as public as below:
198. class Employee{
    a.
        i.      public long id;
        ii.     public String firstName;
199. }
200. Now, create a constructor that can set these fields for particular Employee object, and solution will be as below:
201. class Employee{
    a.
        i.      public long id;
        ii.     public String firstName;
202.
```

i. public Employee(long id, String firstName) {
ii.
1. this.id = id;
2. this.firstName = firstName;
iii. }

203. }
204. What can we do if I want to implement hierarchy system for employees in organization within this class, so solution with final class will be as below:
    a. class Employee{
    b.
        1. private long id;
        2. private String firstName;
        3. public Set<Employee> reportees = new HashSet<Employee>();

205.
        1. public Employee(long id, String firstName) {
    ii.
            a. this.id = id;
            b. this.firstName = firstName;
        2. }

206. }
207. Next question was expectation of tree data structure knowledge with recursion. i.e. what should I do if I want to print the organization hierarchy starting from CEO to the lowest level sub ordinate?
208. To this, you can explain that you can create a recursive function that will take Employee object as argument and you will check for Set<Employee>, and if it is not null then you will call that function recursively and print the firstName variable of particular Employee object, and if Set<Employee> is null then you have reached to the lowest level of tree structure and you can exit the method.
    a. Below is the method if they ask to implement the same.
    b. public void traverse(Employee emp, int level) {
        i.
        ii. for(int i=0;i<level;i++) {
            1. System.out.print("\t");
        iii. }
        iv.
        v. System.out.println(" - "+emp.firstName);
        vi. if(!emp.reportees.isEmpty()) {
            1.
            2. Iterator<Employee> empItr = emp.reportees.iterator();
            3. level++;
            4. while(empItr.hasNext())
                a. traverse(empItr.next(), level);
        vii. }
        viii.
    c. }
209.

210. Apart from the above 2 programs, they only asked me one question, that why should I prefer MongoDB over a conventional database? (as there was MongoDB mentioned in my profile) and for that below link can be referred:
    a. https://urldefense.com/v3/__https://dzone.com/articles/mongodb-vs-rdbms__;!!EWETVAMtCZQyTG4!Wu8_qepFoNlz9Wh4dWQqKvt6eWOjlG_QhvB80BBVehSmCX1OPlUm93BjY2ZynEtsqvOz1Wo1ZFaI_bJ208a_pyE$
    b.
211. There is no static constructor java because a constructor, by definition, cannot be static. What you are referring to is called a "static initialization block." A constructor implies that you are constructing an object. You cannot have a constructor for a class because a class is not an instance of itself.
212. list down Object class methods -> toString(),equals,hashcode,clone,wait,notify
213. cache fetures ->
214. Heap and stack -> Object store in heap and reference of that object is store in stack.
215. why string is immutable? -> string pool
216. Hashset vs treeSet ->
217. Reverse a string using recursion -> use substring and charAt
218. create an immutable class -> final class, private field with final, setter not create and use constructor for set value.
219. output of this code:-
    a. class Test {
220. public static void main(String args[]) {
221. System.out.println(test());
222. }
223. static float test() {
224. static float x = 0.0;
    a. return ++x;
225. }
226. }
227. 9. SQL query for below table:-
228.
229. a. EmployeeDetails
    a. EmpId FullName ManagerId DateOfJoining City
230.
231. b. Project
    a. EmpId Project Salary Variable
232.
    a. "write select query to GET :: EmpId, Project
    b. select EmployeeDetails.EmpId, Project.Project from EmployeeDetails Join Project on EmployeeDetails.EmpId==Project.EmpId;
233.
    a. "write select query to GET :: EmpId, Project, Salary+Variable, JoiningYear(YYYY)
    b. select EmployeeDetails.EmpId, Project.Project, (Project.Salary+Project.variable) as totalSalary, EmployeeDetails.DateOfJoining from Project join EmployeeDetails

234. wait and notify method. -> thread wait for sometime, notify once thread is available for process
235. Object lifecycle -> create-intialize-utilize-delete-cleanUp-garbageCollections
236.
237. oop concept in detail
238. use of static
239. inheritance?
240. Overloading overriding
241. project pe kya kam kiya
242. junit pata hai?
243. which DB used in project? how you used it?
244. find unique string
245. palindrome
246. which collection used
247. arraylist vs linkedlist
248.
249. Rest API def -> communication between server and client with HTTP protocol, scalable, stateless,lossely coupled
250. Rest vs SOAP ->
251. Why we use java 8 not 7.
252. @RestController explain.(@Controller + @ResponseBody)->return value of a method should be directly serialized and returned as the HTTP response body
253.
254. @Respository why we use??
255. Can we send XML in @RestController?-> @GetMapping(value = "/data", produces = MediaType.APPLICATION_XML_VALUE)
256. What is a bean prototype? -> instance of bean created on each request from the IOC container
257. when we use bean prototype singleton? -> single instance create and share across multiple request.
258. bean lifecycle,scops. -> instantiation-populate properties-initialization-bean use-destruction
259. what is database partitioning? -> large table into smaller table
260. spring boot configuration? -> YAML,@Configuration
261. What are streams and why introduce them to java? -> readability and more features like filters and maps.
262. ACID properties.. -> Atomicity(treated individual transaction),Consistency(commit data, if interrupt then roll back),Isolation(multiple transaction does not interfere with each other),Durability(once data is committed in DB , even app is crash or fail that will not effect)
263. Spring AOP -> @Before ,@After
264. database indexing
265. what is cohesiveness? -> High cohesiveness recommend
266. About security concepts like SQL injection , XSS? -> manipulate an application's SQL query by injecting malicious SQL code.Lack of input validation
267. XSS-> maliciouse web script
268. what does API gateway do?
269.

270. CORS error
271. CSRS token
272. Spring security
273. Authentication manager -> It is responsible for coordinating the authentication process and managing a collection of Authentication Providers.
274. Authentication provider -> responsible for actual authentication process with cred
275. Dependency injection
276. inversion control
277. @Component
278. @Table vs @Entity
279. What is the process to create auto create table ->DDL true
280. JPA vs Hibernate
281. Able to create multiple bean with same name? no type can be same and for that we can use qualifier annotation to assign
282. @Qualifier use
283. Can we create object of abstract class? no we can only extend
284. latest version of java? 21
285. can we able to create default method in interface? yes
286. Security context spring security?
287. Native query ->
288. dispatcher servlet ->JSP
289. @bean and object create diff -> term diff in diff framework, OOPS , java based framework
290. JIT
291. What is web services? -> multiple transport protocol support and JSP thymleaf also
292. what is microservice?
293. what is stateless and statefull
294. bean scops -> prototype and singleton
295. what is CI CD?
296. what purpose volatile varibale serve in java? -> multi threading access
297. compare serialization with deserialization -> convert into bytecode and bytecode to code
298. what do you understand by OutOfMemoryError in java? -> heap spaces are exhausted then it will happen.
299. What is base class of all class? -> Object class
300. what is class? -> template for creating a object, class define properties,behaviour
301. why we need wrapper class? -> converting data types, type casting, oops ,nullable
302. Type casting ->
303. Hashmap and set basics
304. Given table remove duplicate entries
305. enum-> group of constant , type safety, you can use space in constant
306. String manipulation question
307. How we can create Thread 2 ways write code And Difference
308. List of status code and meanings
309. =>
310. 200 OK: The request was successful.

311. 201 Created: The request has been fulfilled, resulting in the creation of a new resource.
312. 204 No Content: The server successfully processed the request, but there is no additional content to send.
313. 400 Bad Request: The server could not understand the request due to malformed syntax or invalid request message framing.
314. 401 Unauthorized: The request has not been applied because it lacks valid authentication credentials.
315. 403 Forbidden: The server understood the request but refuses to authorize it.
316. 404 Not Found: The requested resource could not be found on the server.
317. 500 Internal Server Error: A generic error message indicating that the server encountered an unexpected condition that prevented it from fulfilling the request.
318. 503 Service Unavailable: The server is not ready to handle the request. Common causes include a temporary overloading or maintenance of the server.
319.
320. can we make overriding method private in child class if parent class method is public.->No
321. Unique key and primary key SQL. -> primary key at table level unique and unique key is column level
322. Which have faster insertion sql or nosql.
323. which have faster fetch in sql or nosql.
324. interface extends interface.
325. hashmap collision
326. IS-A and HAS-A relationship ->Inheritance,Class dependency
327. Constructor injection
328. setter injection -> return void
329. SQL
330. with Where can not use with aggregation
331. with Having can use in that case
332. Nativequery
333. ->String nativeQuery = "{ fieldName: 'fieldValue' }";
334. List<YourEntity> result = mongoTemplate.find(new BasicQuery(nativeQuery), YourEntity.class);
335.
336. BoolQueryBuilder for elasticsearch.
337. ->BoolQueryBuilder boolQuery = QueryBuilders.boolQuery();
338. boolQuery.must(QueryBuilders.matchQuery("fieldName1", "fieldValue1"));
339. boolQuery.must(QueryBuilders.rangeQuery("fieldName2").gte(10).lte(100));
340.
341. Elasticsearch syntax
342. -> elasticsearchoperations.search(Query,Class.class).get().map(Searchit::getContent).collect(Collectors.toList());
343. ->Query method return NativeSearchQuerybuilder().withQuery(boolQuery).withPageable(pagere).build();
344.
345. @Query Eg
346. ->

```
    // Example of using @Query to write a JPQL query
    @Query("SELECT b FROM Book b WHERE b.author = ?1")
    List<Book> findByAuthor(String author);

    // Example of using @Query to write a native SQL query
    @Query(value = "SELECT * FROM books WHERE publication_year = :year", nativeQuery = true)
    List<Book> findByPublicationYear(int year);
```

Try without catch is not possible
Try with finally possible

1 interface implement by 2 class. we can use @Qualifier to @Autowired.

String reverse-->

a. -string does not have any method for reverse.
b. ->eg>
c. String originalString = "Hello, World!";
d. char[] charArray = originalString.toCharArray();

a. int left = 0;
b. int right = charArray.length - 1;

a. while (left < right) {

```
    // Swap characters at left and right indices
    char temp = charArray[left];
    charArray[left] = charArray[right];
    charArray[right] = temp;

    // Move indices towards the center
    left++;
    right--;
```

a. }

a. String reversedString = new String(charArray);
b. System.out.println(reversedString);

a. -> another way is string builder have reverse method.

Set allow to write single null value
treeset does not allow single null value
Map null key is allow, null value is also allow
List null value allow
Functional interface->single abstract method
String buffer and builder have append method not string have.
Diff Thread class and runnable interface
Diff between treeset and hashset

383. super and this keyword
384. Normalization in DB
385. what is yield and stop diff? -> to allow other thread to execute/terminate thread
386. what is instance creation? -> Memory allocation, constructor invocation, initialization
387. what is vector? -> thread safe
388. exit is keyword?
389. thread priorities -> higher priorities first chance
390. why we use spring boot?-> spring framework 6.1 boot 3.2
391. what are the updates in latest spring boot version? -> reactive programming support
392. write the main method which is invoke the spring boot app
393. JPQL
394. role of spring data rest?
395. String s=new String() - store in heap memory not in string pool because it is string object and it is not constant
396. String s1="hello" - store in string pool because it is string literals and it is constant
397. String s3=s+s1 will store in heap not in string pool because of it will runtime check
398. if both are literals then it will check at compile time and it will store in string pool.
399. String s="hello"-> s.equals("hello")-valid, "hello".equals(s)- valid (we can use this to handle nullpointer exception)
400. Diff between spring and spring boot
401. Diff between webservice and microservice.

Mota data ()

402. new features of java 17 from 8
403. hashtable and hashMap
404. design pattern
405. why to use spring boot
406. what to do if we want create new object every time
407. problem solving skill
408. in kafka if load on producer is increased very much then what are the approaches we can do

Cars 24 ()

409.
```
String s1 = "CheckString";
String s2 = new String("CheckString");
String s3 = s2.concat("Now");
String s4 = "CheckStringNow";
String s5 = new String("CheckStringNow");

System.out.println(s1 == s2);
System.out.println(s3 == s4);
System.out.println(s4 == s5);
System.out.println(s3 == s5);
```

410. POJO Person with 3 fields: name , id , phoneNumber
411. Try catch block with 2 exception in one catch and method which will always execute (finally)
412. From list find first id where id is 5 using stream and java 8

413. Using java 8 find first duplicate id from a list
414. In sql from two table get customer name with number of orders
415. Example 1:Input: [100, 200, 1, 3, 2, 4]
Output: 4
Explanation: The longest consecutive subsequence is 1, 2, 3, and 4.

ASINT/Measureone

416. What is secured http or https ? Explain ?
417. How have you managed multi-threading in your project ?
418. What is Agile ? Diff between Agile VS Waterfall model ? What if you do not follow Agile ?
419. How do you use Kafka in your project ?
420.

```
class Main {

    static {
        System.out.println("A");
    }

    public static void main(String[] args) {
        System.out.println("B");
    }

    static {
        System.out.println("C");
    }
}
```

Output ?

421.

```
class Abc {
    public static void main(String[] args) {
        try {
            System.out.println(1 / 0);
        } catch (ArithmeticException ar) {
            System.out.println("arithm");
        } catch (Exception e) {
            System.out.println("exception");
        } finally {
            System.out.println("finally");
        }
    }
}
```

422.

```
class Abc {
   public static void main(String[] args) {
      System.out.println('j' + 'a' + 'v' + 'a');
   }
}
```

Output ?

423. What is Marker Interface ?
424. What is transient keyword ?
425. What is Serialier ?
426. Diff between DDL VS DML ?
427. 2 VS Delete SQL Diff ?
428. What is try-with-resources ?
429. Which Design Principles you know ?
430. Diff between Cohesion VS Coupling ?
431. What is recursion in java ? write pseudo code.

432. List down starter name which use in spring boot.
433. Inheritance, interface
434. Diff spring and spring boot
435. Rest api design
436. Can we use 2 database in single application
437. Explain exception and error and eg
438. Explain compiletime and runtime error
439. Junit brief
440. How to test external database call
441. Dependency injection
442. Object class method equals use.
443. Hashmap and treemap
444. Arraylist and linkedlist
445. Abstract class have constructor?and some extend it and call it super then what happen?

ASINT/Lucent

446. What is IOC (Inversion of Control) and dependency injection?
447. How have you used Kafka in your projects?
448. What are the disadvantages of using Kafka?
449. In kafka if consumer consumes mess but due to some error it was unable to store data in the database . So will this data be lost?
450. Why did you choose Kafka over RabbitMQ?
451. Explain the try-catch block. In what order are multiple catch blocks executed?
452. What happens if there is only a try block without catch or finally?
453. Which design patterns are you familiar with? Can you explain the Singleton pattern with a code example?
454. What are the types of exceptions in Java (e.g., MethodNotFoundException)?
455. Explain method overloading and overriding with examples.

456. List all Java annotations(explain annotations you used). Can a @Service annotation be used in a repository class? Why or why not?
457. What is a HashMap, and how does it work internally?
458. What is the difference between a LinkedList and an ArrayList?
459. What are the different ways to iterate over a HashMap?
460. Explain synchronization in Java. When should it be used?
461. What is the difference between an interface and abstraction?
462. If we can have default methods in interfaces, how is it different from abstract classes?
463. Can a default method in an interface be public or final?
464. Can we have class private and if yes then can explain in deep?
465. Can a static method in a class be called using an object? Why or why not?
466. Can we invoke a private constructor? (Hint: Using the Constructor class).
467. Explain the super and this keywords. Can you call a parent class method using super?
468. What are access modifiers in Java? What is the difference between protected and default?
469. What is @interface in Java? How is it used?
470. What is the difference between throws and throw in Java?
471. How does a HashMap work internally in Java?
472. Does for(: :) compile in Java? Explain its usage.
473. What is Agile ? Diff between Agile VS Waterfall model ? What if you do not follow Agile ?
474. What are the principles of exception handling in Java? How have you implemented it in your projects?
475. How do you write unit tests in your projects? Provide examples.
476. Why is Kafka important in your projects? Explain its role and benefits.
477. What will be the output of System.out.println('j' + " " + 'a');?
478. ::::::::Quick sort:::::::::
479.
480. class Main {
481. public static void main(String[] args) {
482. System.out.println("Try programiz.pro");
483. int[] arr={9,6,3,4,3,4,2,1};
484. int low=0;
485. int high=arr.length-1;
486. int[] result=qs(arr,low,high);
487. for(int i=0;i<result.length;i++){
488. System.out.print(arr[i]+" ");
489. }
490. }
491. public static int[] qs(int[] arr,int low,int high){
492. // 1. pivot index find
493. // 2. pivot left side sort
494. // 3. pivot right side sort
495. if(low<high){
496. int pIndex=pivotIndexFind(arr,low,high);
497. System.out.println(pIndex);

```
            qs(arr,low,pIndex-1);
            qs(arr,pIndex+1,high);
        }
        return arr;
    }
    public static int pivotIndexFind(int[] arr,int low,int high){
        int i=low;
        int j=high;
        int pivot=arr[low];
        while(i<j){
            while(arr[i]<=pivot && i<=high-1){
                i++;
            }
            while(arr[j]>pivot && low+1<=j){
                j--;
            }
            if(i<j){
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
        int temp = arr[low];
        arr[low]=arr[j];
        arr[j]=temp;
        return j;
    }
}
```

Spring container
ApplicationContext and BeanFactory
Types of ApplicationContext
Types of Dependency injection
Types of Bean scopes
Bean life cycle , @PostConstruct and @PreDestory
BeanPostProcessor-> allows custom logic before and after bean initialization.
List of Spring boot starter dependencies(spring-boot-starter-web,spring-boot-starter-data-jpa,spring-boot-starter-security)
@SpringBootApplication: Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan.
Why field injection is discouraged?
What happened when nested transaction failed?(if propagation is REQUIRED then outer transaction is also rolled back, if REQUIRED_NEW then only inner transaction is rolled back while outer transaction is continue
https://chatgpt.com/c/67acd8fc-91a0-8000-83fc-5c672e4aadd1
/////Kafka////

539. Spring boot with kafka we can use kafka template producing and @KafkaListner to use kafka consumer consuming them. (@KafkaListener(topics = "my-topic", groupId = "my-group") put above method
540. Using a dead-letter topic is a great way to handle failures. You could also configure retries with SeekToCurrentErrorHandler or DefaultErrorHandler in Spring Kafka
541. **At most once**: Messages are delivered at most one time, but they might be lost if a failure occurs before processing.
542. **At least once**: Messages are guaranteed to be processed at least one time, but duplicates may occur.
543. **Exactly once**: Ensures each message is processed only once, avoiding duplicates while preventing message loss. This is achieved using Kafka's idempotent producer and transactional processing.
544. How to implement exactly once semantic.->Set enable.idempotence=true in the producer properties. This prevents duplicate messages.
545. https://chatgpt.com/c/67af4703-e4a0-8013-9dec-dfd40bd91677
546. Transactional Outbox pattern
547. /// Infosys ////
548. What is diff between queue and topic
549. How do you handle exception in queue
550. How you use opensearch in your project
551. Test cases annotation
552. What is batch scheduling where you use
553. What is stream and explain with demo
554. What is circuit breaker pattern and explain
555. What is diff between spring and spring boot
556. What is flux and mono
557. What is boilerplate code in spring that spring boot does not have.
558. IOC and DI concepts
559. How to configure properties in your service.
560. Eureka explain
561. Did you use Env variable to set property value?
562. How to handle diff property for diff env like dev and prod? Use profiles
563. Explain the difference between HashMap and ConcurrentHashMap. When would you use one over the other?
564. How does Spring Boot manage dependency injection, and what are the different types of bean scopes available?
565. What is the difference between @Transactional and programmatic transaction management?
566. How would you handle circular dependencies in Spring Boot?
567. Explain the difference between ExecutorService and ForkJoinPool in Java.
568. Can you explain the concept of lazy and eager fetching in JPA? How do you decide which one to use?
569. What are the key challenges of microservices architecture, and how do you address them?
570. How do you handle inter-service communication in microservices? Which approach do you prefer: REST, gRPC, or messaging (Kafka)?

571. Explain how you would design a high-availability system for a payment gateway like Fiserv.
572. What strategies would you use to scale a microservice handling millions of API requests per second?
573. How would you ensure data consistency in a distributed system using Kafka?
574. How does Kafka ensure message durability and fault tolerance?
575. What is a Dead Letter Queue (DLQ), and how have you implemented it in your projects?
576. How do you handle schema evolution in Kafka messages?
577. How does Kafka's Exactly-Once Processing work? What configurations are needed to achieve it?
578. How does Redis improve performance in a microservices architecture? Can you explain some real-world use cases?
579. Compare MongoDB and MySQL. When would you choose one over the other?
580. How do you implement database connection pooling in Spring Boot?
581. What Boilerplate Code Does Spring Boot Remove? -> (Configuration, Logging, Dependency management, Database config, testing, transaction management)
582. @Embedded and @Embedable ,when to use
583. @Transient use
584. What is session and session factory?
585. OOPS principles - exact meaning of each term (example: Abstraction)
586. Comparable and Comparator program
587. Generics in collection - and its code to add two objects
588. Array size syntax - which code will compile and run
589. Ways to create/handle spring bean
590. Thread basic concept - life cycle
591. Database queries - DDL & DML
592. SQL Joins - query
593. Final , finally, finalize diff -> (finalize - Called by the **Garbage Collector** before an object is destroyed.)
594. In a single table find duplicate student name which is repeating in same class
595. Int arr[] = {5,7,1,9,8,2,13,5} find min sum of adjacent number and also return the two numbers which are minimum sum.
596. JVM concepts deeply
597. Which needed to run the code jvm jre or jdk
598. SQL queries on highest salary on department
599. Lambda expression with deeply
600. Streams concept with largest occurring value find out
601. What is pipeline
602. CriteriaQuery
603. Stored procedure
604. you have been provided with a table called ‘salaries’ that has three columns: EmployeeID, Department, and Salary.write sql query to list the single highest salary in each department
605. Write data structure of Stack,Queue,LinkedList
606. in which failover scenario when kafka consumer is not consuming topic data
607. what is key difference in encapsulated class and normal class.
608. what is the reason why abstract class can not create instance.

609. what if inherited interface have same method name?

Learning Links

- Spring & SpringBoot - https://docs.google.com/document/d/1Z0UNLA_SSuH4IkpQcXTaBDj_3oXPhXQFEaP7aTcfWU4/edit?tab=t.0
- Spring - https://docs.google.com/document/d/1C2PBzuP7s5tCELYjTUshBqqkxUVuOK-u6sXItO9Ophw/edit?tab=t.0
- DSA Roadmap - https://docs.google.com/document/d/1D_fyCBlQaI2DYI74YRczIV0BTx7skMYBlmoQcAEYeIs/edit?tab=t.0
- Agile - https://docs.google.com/document/d/1D_fyCBlQaI2DYI74YRczIV0BTx7skMYBlmoQcAEYeIs/edit?tab=t.0
- Javascript - https://www.interviewbit.com/javascript-interview-questions/