-----------------------JAVA-------------------------------

1. What are the four pillars of OOPS?

Encapsulation, Abstraction, Inheritance, Polymorphism

1. Is multithreading recommended in production, and why?

Yes, for better resource utilization. But must handle concurrency carefully.

1. How to use the stream API for collection transformation?

Use functional operations like map, filter, and reduce for concise code.

1. Benefits of lambda expressions.

Simplifies code, enables functional programming, and reduces boilerplate.

1. Delete vs Truncate
2. Why avoid public static final for Singleton?

Singleton needs controlled lazy instantiation, not just static constants.

1. What happens if you swap @Service and @Repository?

Exception translation may break because @Repositorty handles persistence exceptions.

1. HTTP 403 vs 404

403: Access forbidden, 403: Resource not found.

1. Overview of JVM architecture includes class loader, Runtime Data Areas, Execution engine, and Garbage Collection.
2. What is the difference between Serializable and Externalizable? When would you use each?
3. How does a hash map handle collisions internally?
4. Explain the difference between checked and unchecked exceptions with an example.
5. How do you create a thread-safe singleton using an enum?
6. What is the difference between ServiceLoader and Dependency Injection?
7. How does the JVM optimize string concatenation with StringBuilder?
8. Explain the differences between Callable and Runnable
9. What is the purpose of ThreadLocal in Java? Give an example use case.
10. Singleton vs Immutable.
11. Predicate & Functional Interface
12. ConcurrentModificationException and how to avoid it.
13. Saga pattern in Microservices
14. Authentication vs Authorization
15. Spring Data: JPA vs JDBC vs JWT
16. Optional Class
17. Terminal vs Intermediate Operations(Stream)
18. API Gateway
19. Is JWT alone enough for Spring Security?
20. Spring Security real-time examples.
21. How do you handle versioning in REST APIs?
22. SOLID Principles and Design Pattern
23. Reverse a LinkedList
24. Distributed Transactions in Microservices
25. Service Discovery in microservices.
26. Inter-service communication strategies
27. Types of Garbage Collectors.
28. What is the difference between eager & lazy initialization in Java? How does spring support both?
29. What are bounded type parameters in Java generics? How are they defined and userId?
30. Explain the role of ThreadPoolExecutor. How do you configure core size, max size, and queue behaviour?
31. What is a memory-mapped file in Java(MappedByteBuffer)? When would you use one?
32. How does the @RequestScope, @SessionScope, and @ApplicationScope work in Spring
33. What are some common causes of the OutOfMemory Error in Java, and how would you diagnose them?
34. How does Java’s try-with-resources statement work? What is an Autocloseable interface?
35. What are default methods in interfaces? How do they affect multiple inheritance?
36. How does the Java Stream API handle parallelism internally? What are the best practices for parallel streams?
37. Explain how the Phantom Reference class works and its use cases.
38. How does Java’s CompletableFuture handle? Explain with an example.
39. What are the main differences between Hashtable and ConcurrentHashMap?
40. What is the role of the Unsafe class in Java? Why should it be used cautiously?
41. Describe how you would implement a thread-safe singleton in Java.
42. What are Java annotations' retention policies, and why are they important?
43. How do you implement a custom serializer/deserializer in Jackson for JSON processing?
44. What is the difference between the Predicate function and Consumer functional interfaces?
45. What is the difference between Class.forName() and ClassLoader.loadClass()?
46. How do you prevent memory leaks in Java application?
47. Explain the difference between clone and new when creating an object.
48. How does java.util.concurrent.atomic package help in multi-threading?
49. How does the finalize() method work, and why is it deprecated?
50. What are the sealed classes in Java? How do they help in design?
51. How does the Java compiler handle generic type erasure?
52. What are the pros and cons of reflection?
53. Use @WebMvcTest, @DataJpaTest, and Mockito for testing controllers and repositories.
54. Circuit Breaker pattern
55. Swagger, Mockito, TarbitMq, Kafka, Redis, Ehcache
56. Can You declare a private method in Java? When and why do you use it?
57. What’s the difference between a default and static method in an interface?
58. What are varargs in Java, and when should you use them?
59. Why does the main method accept String[] args? Can it be overloaded?
60. What happens if a static method is called on a null reference?
61. What is the contract between equals() and hashcode()? Why is it crucial in collections like HashMap/HashSet?
62. Real-world use case for overriding equals() and hashCode() in a class.
63. What is a marker interface? Can you create one? Give me a practical example.
64. Can a functional interface exist without any abstract method?
65. If both try and finally blocks have a return statement, which one gets executed?
66. How does the synchronized keyword work internally in Java?
67. What is thread starvation, and how can it be prevented?
68. What is a race condition, and how can you avoid it?
69. Concurrency vs Parallelism
70. What are the advantages of using ExecutorService over manual thread creation?
71. What is a daemon thread? When should it be used?
72. What is the Java memory model, and why is it crucial?
73. How does ReentrantLock work, and how is it better than synchronized?
74. What is CopyOnWriteArrayList? When should it be preferred?
75. How does CountDownLatch work?
76. Compare CountDownLatch vs CyclicBarrier with real-time examples.
77. What is ForkJoinPool? Where do we use it?
78. What are atomic variables? Why are they thread-safe?
79. How does BlockingQueue work? What is its type?
80. What is ReadWriteLock, and how does it improve performance?
81. Explain the use of a Semaphore with a use case.
82. How does the Java thread scheduler work? Can we really control thread priority?
83. What is the difference between optimistic and pessimistic locking?
84. How does thread context switching impact performance?
85. When does a Null pointer Exception occur in Java?
86. What is the record class in Java
87. What is the difference between the DTO class and the record class
88. What is static binding and dynamic binding?
89. How to check if a given string is null or not
90. What is covariance and contravariance in Java generics?
91. How do you implement lazy initialization in Java?
92. Describe how you can use annotations to enforce validation in Java applications.
93. What is the difference between Arrays.asList() and List.of()?
94. Describe how exception chaining works in Java.
95. How do you implement thread-safe lazy initialization with double-checked locking?
96. What is the use of WeakHashMap, and how does it differ from a regular HashMap?
97. What is the concept of method hiding in Java?
98. How does the JVM handle method overloading and overriding at runtime?
99. Explain the concept of class and instance initialization order in Java.
100. What is the difference between a Soft and a Strong reference in Java?
101. What are some ways to improve Java application startup time?
102. How do you handle circular dependencies in the Spring Framework?
103. What is tail recursion? Does Java optimize for tail-recursive calls?
104. What is the difference between Map.computeIfAbsent() and traditional get-then-put logic?
105. Explain the builder design pattern. When is it Preferable over telescoping constructors?
106. How does Java handle time zones with java.time.ZoneDateTime? What are some common pitfalls?
107. How would you prevent SQL injection in a Java-based web application?
108. What is double-checked locking? Why is the volatile keyword essential in this pattern?
109. How can you monitor and profile the memory usage of a Java application in production?
110. What are some common pitfalls of using floating-point numbers (float/double) in financial calculations? How can they be avoided in Java?
111. How does Java support functional programming concepts?
112. Explain the role of Class and ClassLoader in Java reflection.
113. What is a method reference in Java? How does it simplify lambda expressions?
114. What is the use of the Native keyword in Java?
115. What happens if an exception is thrown in the finally block?
116. How do you configure and use profiles in Spring Boot for different environments ( dev/test/prod)?
117. How do you secure a Spring Boot REST API using JWT and Spring Security?
118. Explain the use of Spring Cloud Config in a microservices environment.
119. How would you implement service discovery using Eureka or Consul?
120. How do you ensure resilience in Spring Boot Microservices (e.g, Circuit Breaker, Retry, Fallback)?
121. What are some common performance tuning practices in Spring Boot applications?
122. How would you implement asynchronous processing in Spring Boot?
123. How do you handle data consistency across distributed microservices?
124. How do you get JSON data from a client in an API? Can a GET API accept JSON data?
125. What is the difference between an Inner Join and an Outer Join in SQL?
126. Volatile vs Atomic
127. How Threads Signal Completion?
128. Spring Boot Design Pattern (API Gateway, Circuit Breaker, Saga Pattern, Database per Service Pattern, Strangler Fig Pattern, Bulkhead Pattern)
129. How does Java implement pass-by-value semantics?
130. Explain how you would handle thread interruption in Java?
131. How does the Split iterator interface work in Java streams?
132. How do you implement a concurrent LRU cache in Java?
133. Explain the difference between the Optional class
134. How do Java generics improve type safety?
135. Explain the use of the synchronized keyword with static methods
136. What is the role of the java.lang.invoke package
137. What are the new features of the switch statement?
138. How does Java handle checked exceptions differently from unchecked exceptions?
139. Explain the concept of “Effective Java” item: Favor composition over inheritance.
140. Describe how the var keyword works in Java 10 and its benefits.
141. How does Spring Boot auto-configuration work internally?
142. How do you override default configurations in Spring Boot?
143. What is the use of CommandLineRunner and ApplicationRunner?
144. How does an embedded server (Tomcat) work in Spring Boot?
145. How do profiles work in Spring Boot (@Profile use case)?
146. How does SecurityFilterChain work in Spring Boot 3+?
147. What is the UserDetailService, and how do you implement it?
148. How do you implement JWT token-based authentication in Spring Security?
149. How do you restrict access based on roles in Spring Boot?
150. How do you customize login/logout flows in Spring Security?
151. How do you create a custom annotation and intercept it using AOP?
152. How do you test REST controllers using MockMvc?
153. What is the difference between peek() and map() in Java Streams?
154. Describe how the instanceof pattern matching(Java 16+) works.
155. How does java’s var type inference work with generics?
156. Explain the difference between Comparator and Comparable.
157. What is the use of the @FunctionalInterface annotation?
158. How does Java’s switch expression differ from the traditional switch statement?
159. What is the difference between ClassCastException & ClassNotFoundException?
160. Explain how the synchronized collections differ from concurrent collections.
161. What is the difference between Executor and ExecutorService?
162. What are the main benefits of using Java Modules(in Java 9)?
163. Explain how Java’s Stream pipeline works internally.
164. What is the difference between try-catch-finally and try-with-resources?
165. How do you implement a thread-safe producer-consumer pattern in Java?
166. What are the differences between Path and File Classes in Java NIO?
167. How do you handle exceptions in lambda expressions?
168. Explain ConcurrentLinkedQueue
169. Explain how the stream.collect() method works with Collectors.
170. How does Java’s garbage collector handle circular references?
171. What are the advantages of enum types over constant variables?
172. Explain how you would create a custom thread pool in Java?
173. How do you prevent memory leaks caused by event listeners?
174. Describe the use of the @SafeVarargs annotation.
175. How do Microservices communicate with each other?
176. Consuming messages using Kafka in a real-world project
177. What is the difference between final, const, and immutable in Java
178. Difference between Web API and REST API?
179. 200,201,204, 400,404
180. What is Media Type Formatter?
181. What is attribute routing in Web API?
182. What is convention-based routing?
183. URL vs Header versioning- what is better?
184. Best practice to secure Web API endpoints?
185. How to restrict routes to specific HTTP methods?
186. How to enable CORS in Web API?
187. How to test a Web API using Postman?
188. What are Action Filters in Web API?
189. How to create custom Action Filters?
190. How to customize JSON formatting?
191. How to upload/download files via Web API?
192. Leverage Virtual Thread in Java 21
193. @ConfigurationProperties vs @Value
194. How do you manage communication between Services, async vs sync?
195. How do you perform distributed logging and tracing?
196. What are indempotent operations in microservices, and why are they important?
197. What is event-driven architecture, and where have you used it?
198. How do you avoid tight coupling between microservices?
199. How do you handle transaction management in microservices?
200. What are the common challenges with microservices in production?
201. How do you secure inter-service communication?
202. What is the sidecar pattern, and how does it work with service mesh(e.g., Isito)?
203. How do you implement rate limiting and throttling?
204. What’s your strategy for blue-green deployments / canary releases?
205. How do you handle partial failure and retries?
206. What is CQRS (Command Query Responsibility Segregation), and when is it useful?
207. How do you manage configurations across environments in microservices?
208. What is the role of the database per service, and how do you manage cross-service queries?
209. What are dead letter queues(DLQs), and when do you use them?
210. Share your experience debugging a complex issue in a distributed system
211. @Bean vs @Component
212. How do you manage microservices transactions in case each service has a different database?
213. What are the ACID properties?
214. How do you route your web request to the appropriate service in microservices?
215. What is a trigger in a database?
216. What are the properties of Spring Boot?
217. What is a container?
218. How do you deploy & test your application?
219. What is Themeleaf?
220. What is scalable? What do we actually do?
221. What are the characteristics of software?
222. What are the types of software?
223. What is the SDLC model? Explain the phases.
224. Suppose I have throws the runtimeException, what happens?
225. What is the waterfall model? What happens inside that?
226. What is black box & white box testing?
227. What is beta & alpha testing?
228. Load balancing in microservices using Spring Cloud Load Balancer
229. How does Spring Cloud Config help in centralized configuration management?
230. Feign Client vs WebClient. Which one to use and why?
231. Database per service vs shared database. Pros & cons
232. Role of Prometheus and Grafana in microservices monitoring.
233. When to use WebFlux for reactive microservices?
234. Kubernetes deployment strategies for microservices
235. What is a webhook, and how does it work
236. If the Jenkins master node fails or crashes, how would you troubleshoot it
237. Jenkins node management & configurations
238. Jenkins Plugin, Which plugin do you use and why?
239. How to retry plugin installation if it fails in Jenkins