Java 後端工程師面試

Java Core (20 分)

1. 面向對象編程（OOP）原則：封裝、繼承、多態、抽象。
2. Java 泛型：使用類型參數、有界類型和通配符泛型。
3. Java 多執行緒：創建執行緒、執行緒生命週期和執行緒間通信。
4. JVM 記憶體管理：堆、堆棧、永久代/存活區、垃圾回收算法。
5. 異常處理：檢查和未檢查異常、try-catch 塊、finally 和多重捕捉。
6. Java 串行化：Serializable 接口、自定義串行化使用 writeObject 和 readObject。
7. Java 集合框架：List、Set、Map、Queue 接口及其實現。
8. Lambda 表達式和函數式接口：使用 Predicate、Consumer、Supplier 和 Function。
9. Stream API：中間和終端操作、並行流和流管道。
10. 反射 API：在運行時訪問類、方法和字段、註解處理。
11. Java IO vs NIO：文件處理差異、基於通道的 I/O 和非阻塞 I/O。
12. Java 日期和時間 API：使用 LocalDate、LocalDateTime 和 Duration。
13. Java 網絡編程：套接字編程、URL 連接和 HTTP 客戶端。
14. Java 安全性：加密、數字簽名和安全編碼實踐。
15. Java 模塊：理解 JPMS（Java 平台模塊系統）和模塊化。
16. Java 枚舉：使用枚舉、序數值和自定義枚舉方法。
17. Java 注解：內建註解、自定義註解和註解處理。
18. Java 並發工具：CountDownLatch、CyclicBarrier、Semaphore 和 Exchanger。
19. Java 記憶體洩漏：原因、檢測和預防策略。
20. Java 性能調優：JVM 選項、分析工具和記憶體優化技術。

Spring 生態系統 (20 分)

21. Spring IoC 容器：依賴注入、bean 生命週期和作用域。
22. Spring Boot 自動配置：Spring Boot 自動配置 bean 的方式。
23. Spring Data JPA：儲存庫模式、CRUD 操作和查詢方法。
24. Spring Security：身份驗證、授權和保護 REST API。

25. Spring MVC：控制器方法、請求映射和視圖解析。
26. Spring Cloud：使用 Eureka 的服務發現、使用 Ribbon 的負載均衡。
27. Spring AOP：面向切面編程、跨切面關注點和建議類型。
28. Spring Boot Actuator：監控端點、健康檢查和指標收集。
29. Spring 設定檔：環境特定配置和設定檔激活。
30. Spring Boot Starter 依賴：使用啟動器簡化依賴管理。
31. Spring Integration：整合不同系統、消息和適配器。
32. Spring Batch：批處理、作業調度和步驟實現。
33. Spring Cache：緩存策略、註解和緩存管理器。
34. Spring WebFlux：反應式編程、非阻塞 I/O 和 WebFlux 框架。
35. Spring Cloud Config：微服務的集中配置管理。
36. Spring Cloud Gateway：API 閘道模式、路由和過濾。
37. Spring Boot 測試：使用 @SpringBootTest、MockMvc 和 TestRestClient。
38. Spring Data REST：將儲存庫公開為 RESTful 服務。
39. Spring Cloud Stream：與消息代理如 RabbitMQ 和 Kafka 的整合。
40. Spring Cloud Sleuth：微服務中的分佈式追蹤和日誌記錄。

微服務架構 (20 分)

41. 服務發現：Eureka、Consul 和 Zookeeper 的工作原理。
42. API 閘道：模式、路由和 API 閘道的安全性。
43. 熔斷器：使用 Hystrix、Resilience4j 實現彈性。
44. 事件驅動架構：事件源、消息代理和事件處理器。
45. RESTful API 設計：HATEOAS、無狀態設計和 REST 限制。
46. GraphQL：實現 GraphQL API、模式定義和解析器。
47. 微服務通信：同步 vs 非同步通信。
48. Saga 模式：跨服務管理分佈式事務。
49. 健康檢查：實現活性和就緒探針。
50. 契約優先開發：使用 Swagger 進行 API 契約。
51. API 版本控制：RESTful API 版本控制策略。

52. 速率限制：實現速率限制以防止濫用。
53. 熔斷器模式：實現回退和重試。
54. 微服務部署：使用 Docker、Kubernetes 和雲平台。
55. 服務網格：了解 Istio、Linkerd 及其優點。
56. 事件協作：Saga vs 協同模式。
57. 微服務安全性：OAuth2、JWT 和 API 閘道。
58. 監控和追蹤：Prometheus、Grafana 和 Jaeger 等工具。
59. 微服務測試：集成測試、契約測試和端到端測試。
60. 每個服務的數據庫：微服務中的數據管理和一致性。

數據庫和緩存 (20 分)

61. SQL 連接：內連接、外連接、左連接、右連接和交叉連接。
62. ACID 屬性：事務中的原子性、一致性、隔離性和持久性。
63. NoSQL 數據庫：文檔存儲、鍵值存儲和圖數據庫。
64. Redis 緩存：內存數據存儲、數據結構和持久化選項。
65. Memcached vs Redis：比較緩存解決方案。
66. 數據庫分片：水平分區和負載均衡。
67. ORM 框架：Hibernate、MyBatis 和 JPA 規範。
68. JDBC 連接池：DataSource 實現和連接生命週期。
69. 全文搜索：在 Elasticsearch 等數據庫中實現搜索。
70. 時間序列數據庫：InfluxDB、OpenTSDB 等時間序列數據。
71. 事務隔離級別：未提交讀取、已提交讀取、可重複讀取和可序列化。
72. 索引策略：B-tree、哈希索引和複合索引。
73. 數據庫複製：主從、主主設置。
74. 數據庫備份和恢復：數據保護策略。
75. 數據庫分析：SQL Profiler、慢查詢日誌等工具。
76. NoSQL 一致性模型：最終一致性、CAP 定理。
77. 數據庫遷移：使用 Flyway、Liquibase 進行模式變更。
78. 緩存策略：旁路緩存、讀取通過和寫入通過模式。

79. 緩存失效：管理緩存過期和失效。
80. 數據庫連接池：HikariCP、Tomcat JDBC 池配置。

並發和多執行緒 (20 分)

81. 執行緒生命週期：新、可運行、運行、阻塞、等待、終止。
82. 同步機制：鎖、同步塊和內在鎖。
83. 可重入鎖：同步塊的優點、公平性和超時。
84. 執行器框架：ThreadPoolExecutor、ExecutorService 和執行器池配置。
85. Callable vs Runnable：差異和用例。
86. Java 記憶體模型：可見性、發生前關係和記憶體一致性。
87. 易失關鍵字：確保變量變更在執行緒間的可見性。
88. 死鎖預防：避免和檢測死鎖。
89. 非同步編程：使用 CompletableFuture 進行非阻塞操作。
90. ScheduledExecutorService：按固定速率和延遲調度任務。
91. 執行緒池：固定、緩存和調度執行緒池。
92. 鎖條紋：使用條紋鎖減少鎖爭用。
93. 讀寫鎖：允許多個讀者或單個寫者。
94. 等待和通知機制：使用 wait/notify 進行執行緒間通信。
95. 執行緒中斷：處理中斷和設計可中斷任務。
96. 執行緒安全類：實現執行緒安全單例模式。
97. 並發工具：CountDownLatch、CyclicBarrier、Semaphore。
98. Java 8+ 並發特性：並行流、分叉/合併框架。
99. 多核編程：並行處理的挑戰和解決方案。
100. 執行緒轉傳和分析：使用執行緒轉傳識別問題。

Web 伺服器和負載均衡 (20 分)

101. Apache Tomcat 配置：設置連接器、context.xml 和 server.xml。
102. Nginx 作為反向代理：配置 proxy_pass、上游伺服器和負載均衡。
103. HAProxy 高可用性：設置故障轉移和會話持久性。

104. Web 伺服器安全性：SSL/TLS 配置、安全標頭和防火牆規則。
105. 負載均衡算法：輪詢、最少連接、IP 哈希。
106. 伺服器端緩存：使用 Varnish、Redis 或內存緩存。
107. 監控工具：使用 Prometheus、Grafana 和 New Relic 進行伺服器監控。
108. 生產日誌記錄：使用 ELK 堆棧或 Graylog 進行集中日誌記錄。
109. 水平 vs 垂直擴展：理解權衡和使用案例。
110. Web 伺服器性能調優：調整工作線程、連接超時和緩衝區。
111. 反向代理緩存：配置緩存標頭和過期。
112. Web 伺服器負載測試：使用 Apache JMeter、Gatling 等性能測試工具。
113. SSL 卸載：在負載均衡器處理 SSL/TLS 終止。
114. Web 伺服器硬化：安全最佳實踐和漏洞評估。
115. 動態 vs 靜態內容服務：優化伺服器配置。
116. Web 伺服器集群：設置集群以實現高可用性。
117. Web 伺服器驗證：實現基本、摘要和 OAuth 驗證。
118. Web 伺服器日誌格式：常見日誌格式和解析工具。
119. Web 伺服器資源限制：配置連接、請求和頻寬限制。
120. Web 伺服器備份和恢復：災難恢復策略。

CI/CD 和 DevOps (20 分)

121. Jenkins Pipeline as Code：編寫 Jenkinsfiles 進行 CI/CD 管道。
122. Docker 容器化：Dockerfile 創建、多階段構建和容器編排。
123. Kubernetes 編排：部署、服務、Pod 和擴展策略。
124. GitOps 原則：使用 Git 進行基礎設施和配置管理。
125. Maven 和 Gradle 构建工具：依賴管理、插件和構建生命週期。
126. 單元和集成測試：使用 JUnit、Mockito 和 TestNG 進行測試。
127. 代碼覆蓋工具：使用 Jacoco 衡量代碼覆蓋率。
128. 靜態代碼分析：使用 SonarQube 進行代碼質量檢查。
129. 基礎設施即代碼（IaC）：使用 Terraform、CloudFormation 進行基礎設施配置。
130. 藍綠部署：在部署期間最小化停機時間。

131. 金絲雀部署：逐步推出新功能。

132. 自動化測試在 CI 管道中：將測試與構建階段集成。

133. 環境管理：使用 Ansible、Chef 或 Puppet 進行配置管理。

134. CI/CD 最佳實踐：持續集成、持續部署和持續交付。

135. 回滾策略：在部署失敗時實現自動回滾。

136. 安全掃描：在管道中納入安全檢查如 SAST、DAST。

137. CI/CD 管道的微服務：管理多個服務的管道。

138. 監控 CI/CD 管道：在管道失敗和性能問題時發出警報。

139. DevOps 工具生態系統：了解 Docker、Kubernetes、Jenkins、Ansible 等工具。

140. 針對雲原生應用的 CI/CD：在雲平台上部署應用。

設計模式和最佳實踐 (20 分)

141. 單例模式：實現執行緒安全單例。

142. 工廠模式：創建對象而不指定具體類。

143. 策略模式：封裝算法並在它們之間切換。

144. SOLID 原則：理解和應用單一職責、開閉、里氏替換、接口隔離和依賴反轉。

145. 依賴注入：減少耦合並提高代碼可維護性。

146. 事件源模式：存儲事件以重建應用狀態。

147. CQRS 架構：分離命令和查詢責任。

148. 設計可擴展性：使用水平擴展、分片和負載均衡。

149. 代碼重構技術：提取方法、重命名變量和簡化條件。

150. 乾淨代碼實踐：編寫可讀、可維護和自文檔的代碼。

151. 測試驅動開發（TDD）：在實現之前編寫測試。

152. 代碼版本控制：使用 Git 分支策略如 GitFlow、基於主幹開發。

153. 設計可維護性：使用模塊設計、關注點分離。

154. 反模式避免：神類、意大利麵條代碼和緊耦合。

155. 設計安全性：實現最小權限、防禦深度。

156. 設計性能：優化算法、減少 I/O 操作。

157. 設計可靠性：實現冗餘、容錯和錯誤處理。

158. 設計可擴展性：使用插件、擴展和開放 API。

159. 設計可用性：確保 API 直觀且文檔完善。

160. 設計可測試性：編寫易於測試和模擬的代碼。

安全性 (20 分)

161. OAuth2 和 JWT：實現基於令牌的身份驗證。

162. 基於角色的訪問控制（RBAC）：為用戶分配角色和權限。

163. 安全標頭：實現內容安全策略、X-Frame-Options。

164. SQL 注入防護：使用預處理語句和參數化查詢。

165. 跨站腳本（XSS）保護：對輸入和輸出進行清理。

166. 加密和解密：使用 AES、RSA 進行數據保護。

167. 安全編碼實踐：避免常見漏洞如緩衝區溢出。

168. 實現審計跡：記錄用戶操作和系統事件。

169. 處理敏感數據：使用哈希算法安全存儲密碼。

170. 遵守法規：GDPR、PCI-DSS 和數據保護法規。

171. 實現雙因素身份驗證（2FA）：添加額外的安全層。

172. 安全測試：滲透測試、漏洞評估。

173. 安全通信協議：實現 SSL/TLS 進行數據加密。

174. 安全會話管理：管理會話令牌和超時。

175. 實現 Web 應用防火牆（WAF）：保護免受常見攻擊。

176. 安全監控和警報：使用 SIEM 進行威脅檢測。

177. 微服務中的安全最佳實踐：保護服務之間的通信。

178. 實現 CAPTCHA 以防止機器人攻擊：防止自動化攻擊。

179. CI/CD 管道中的安全性：在構建期間掃描漏洞。

180. 安全性由設計：從開發過程的開始納入安全性。

性能調優和優化 (20 分)

181. Java 應用程序分析：使用 JProfiler、VisualVM 進行性能分析。

182. 垃圾回收調優：調整 GC 參數以提高性能。

183. 數據庫查詢優化：索引、查詢重寫和使用解釋計劃。

184. 緩存策略：使用分佈式緩存、緩存失效機制。

185. 負載測試和壓力測試：識別性能瓶頸。

186. 優化 RESTful API：減少響應時間、最小化數據傳輸。

187. 降低網絡延遲：使用 CDN、優化 API 請求。

188. 連接池大小：確定數據庫和連接的最佳池大小。

189. 監控和警報設置：使用 Prometheus、Grafana 進行實時監控。

190. 識別和解決瓶頸：分析 CPU、記憶體和 I/O 使用情況。

191. 優化 Java 堆設置：為不同環境設置適當的堆大小。

192. 降低垃圾回收暫停：使用 G1GC、ZGC 進行低延遲應用。

193. 優化磁盤 I/O：使用 SSD、RAID 配置和文件系統優化。

194. 緩存 vs 存儲：決定何時緩存數據或將其存儲在數據庫中。

195. 優化日誌記錄：減少日誌開銷並管理日誌量。

196. 優化並發訪問：有效使用鎖並最小化爭用。

197. 分析記憶體使用：識別記憶體洩漏並優化對象分配。

198. 優化執行緒池大小：在過少和過多執行緒之間取得平衡。

199. 優化數據結構：根據具體用例選擇合適的數據結構。

200. 性能指標和 KPI：定義和跟蹤應用的關鍵性能指標。