Java バックエンドエンジニア面接

Java Core (20 points)

1. OOP の原則の理解: カプセル化、継承、多態性、抽象化。
2. Java のジェネリクス: 型パラメータ、制限付き型、ワイルドカードジェネリクスの使用。
3. Java のマルチスレッド: スレッドの作成、スレッドのライフサイクル、スレッド間通信。
4. JVM のメモリ管理: ヒープ、スタック、PermGen/Survivor スペース、ガベージコレクションアルゴリズム。
5. 例外処理: チェック例外、アンチェック例外、try-catch ブロック、finally、マルチキャッチ。
6. Java のシリアライズ: Serializable インターフェース、カスタムシリアライズで writeObject と readObject。
7. Java Collections Framework: List、Set、Map、Queue インターフェースとその実装。
8. ラムダ式と関数型インターフェース: 謂語、消費者、供給者、関数の使用。
9. Stream API: 中間操作、終端操作、並列ストリーム、ストリームパイプライン。
10. リフレクション API: クラス、メソッド、フィールドのランタイムアクセス、アノテーション処理。
11. Java IO vs NIO: ファイル処理の違い、チャネルベースの I/O、非ブロッキング I/O。
12. Java Date and Time API: LocalDate、LocalDateTime、Duration の使用。
13. Java ネットワーキング: ソケットプログラミング、URL 接続、HTTP クライアント。
14. Java セキュリティ: 暗号化、デジタル署名、セキュアなコーディング実践。
15. Java モジュール: JPMS（Java Platform Module System）とモジュール性の理解。
16. Java 列挙型: 列挙型の使用、順序値、列挙型のカスタムメソッド。
17. Java アノテーション: 組み込みアノテーション、カスタムアノテーション、アノテーション処理。
18. Java 並行ユーティリティ: CountDownLatch、CyclicBarrier、Semaphore、Exchanger。
19. Java メモリリーク: 原因、検出、防止戦略。
20. Java パフォーマンス調整: JVM オプション、プロファイリングツール、メモリ最適化技術。

Spring エコシステム (20 points)

21. Spring IoC コンテナ: 依存性注入、ビーンのライフサイクル、スコープ。
22. Spring Boot の自動設定: Spring Boot がビーンを自動的に設定する方法。
23. Spring Data JPA: リポジトリパターン、CRUD 操作、クエリメソッド。
24. Spring Security: 認証、認可、REST API のセキュリティ保護。

25. Spring MVC: コントローラーメソッド、リクエストマッピング、ビュー解決。
26. Spring Cloud: Eureka によるサービス発見、Ribbon によるロードバランシング。
27. Spring AOP: アスペクト指向プログラミング、横断的関心事、アドバイスの種類。
28. Spring Boot Actuator: モニタリングエンドポイント、ヘルスチェック、メトリクス収集。
29. Spring プロファイル: 環境固有の設定とプロファイルのアクティベーション。
30. Spring Boot スターター依存関係: スターターの使用による依存関係管理の簡素化。
31. Spring Integration: 異なるシステムの統合、メッセージング、アダプター。
32. Spring Batch: バッチ処理、ジョブスケジューリング、ステップ実装。
33. Spring Cache: キャッシュ戦略、アノテーション、キャッシュマネージャー。
34. Spring WebFlux: リアクティブプログラミング、非ブロッキング I/O、WebFlux フレームワーク。
35. Spring Cloud Config: マイクロサービス向けの中央集中型設定管理。
36. Spring Cloud Gateway: API ゲートウェイパターン、ルーティング、フィルタリング。
37. Spring Boot テスト: @SpringBootTest、MockMvc、TestRestClient の使用。
38. Spring Data REST: リポジトリを RESTful サービスとして公開。
39. Spring Cloud Stream: RabbitMQ や Kafka などのメッセージブローカーとの統合。
40. Spring Cloud Sleuth: マイクロサービスにおける分散トレーシングとロギング。

マイクロサービスアーキテクチャ (20 points)

41. サービス発見: Eureka、Consul、Zookeeper の動作。
42. API ゲートウェイ: パターン、ルーティング、API ゲートウェイのセキュリティ。
43. サーキットブレーカー: Hystrix、Resilience4j を使用した耐障害性の実装。
44. イベント駆動型アーキテクチャ: イベントソーシング、メッセージブローカー、イベントハンドラー。
45. RESTful API 設計: HATEOAS、ステートレス設計、REST 制約。
46. GraphQL: GraphQL API の実装、スキーマ定義、リゾルバー。
47. マイクロサービス通信: 同期通信と非同期通信。
48. サガパターン: サービス間の分散トランザクション管理。
49. ヘルスチェック: ライヴネスとレディネスプローブの実装。
50. コントラクトファースト開発: Swagger を使用した API 契約。
51. API バージョニング: RESTful API のバージョニング戦略。

52. レート制限: 悪用を防ぐためのレート制限の実装。
53. サーキットブレーカーパターン: フォールバックとリトライの実装。
54. マイクロサービスデプロイ: Docker、Kubernetes、クラウドプラットフォームの使用。
55. サービスメッシュ: Istio、Linkerd の理解とその利点。
56. イベント協調: サガパターンとコレオグラフィーパターン。
57. マイクロサービスセキュリティ: OAuth2、JWT、API ゲートウェイ。
58. モニタリングとトレーシング: Prometheus、Grafana、Jaeger などのツール。
59. マイクロサービステスト: 統合テスト、契約テスト、エンドツーエンドテスト。
60. サービスごとのデータベース: マイクロサービスにおけるデータ管理と整合性。

データベースとキャッシュ (20 points)

61. SQL ジョイン: インナー、アウター、左、右、クロスジョイン。
62. ACID 特性: トランザクションにおける原子性、整合性、分離性、持続性。
63. NoSQL データベース: ドキュメントストア、キー値ストア、グラフデータベース。
64. Redis キャッシュ: メモリ内データストア、データ構造、永続化オプション。
65. Memcached vs Redis: キャッシュソリューションの比較。
66. データベースシャーディング: 水平パーティショニングとロードバランシング。
67. ORM フレームワーク: Hibernate、MyBatis、JPA 仕様。
68. JDBC 接続プール: DataSource 実装と接続ライフサイクル。
69. フルテキスト検索: Elasticsearch などのデータベースでの検索実装。
70. タイムシリーズデータベース: InfluxDB、OpenTSDB などの時間ベースデータ。
71. トランザクション分離レベル: 読み取り未確定、読み取り確定、繰り返し読み取り、シリアライズ可能。
72. インデックス戦略: B-tree、ハッシュインデックス、複合インデックス。
73. データベースレプリケーション: マススレーブ、マスターマスター設定。
74. データベースバックアップとリカバリ: データ保護のための戦略。
75. データベースプロファイリング: SQL プロファイラ、スロークエリログなどのツール。
76. NoSQL 整合性モデル: 最終一貫性、CAP 定理。
77. データベースマイグレーション: Flyway、Liquibase を使用したスキーマ変更。
78. キャッシュ戦略: キャッシュアサイド、リードスルー、ライトスルーパターン。

79. キャッシュ無効化: キャッシュの期限切れと無効化の管理。
80. データベース接続プール: HikariCP、Tomcat JDBC プール設定。

並行処理とマルチスレッド (20 points)

81. スレッドライフサイクル: 新規、実行可能、実行中、ブロック、待機、終了。
82. 同期メカニズム: ロック、同期ブロック、内部ロック。
83. 再入可能ロック: 同期ブロックよりの利点、公平性、タイムアウト。
84. Executor フレームワーク: ThreadPoolExecutor、ExecutorService、スレッドプール設定。
85. Callable vs Runnable: 違いと使用例。
86. Java メモリモデル: 可視性、happens-before 関係、メモリ整合性。
87. volatile キーワード: スレッド間での変数変更の可視性を確保。
88. デッドロック防止: デッドロックの回避と検出。
89. 非同期プログラミング: CompletableFuture を使用した非ブロッキング操作。
90. ScheduledExecutorService: 固定レートと遅延でタスクのスケジューリング。
91. スレッドプール: 固定、キャッシュ、スケジュールスレッドプール。
92. ロックストライピング: ストライプロックを使用したロック競合の軽減。
93. 読み書きロック: 複数の読み取りまたは単一の書き込みを許可。
94. 待ち通知メカニズム: wait/notify を使用したスレッド間通信。
95. スレッド中断: 中断の処理と中断可能なタスクの設計。
96. スレッドセーフクラス: スレッドセーフなシングルトンパターンの実装。
97. 並行ユーティリティ: CountDownLatch、CyclicBarrier、Semaphore。
98. Java 8+ 並行機能: 並列ストリーム、フォークジョインフレームワーク。
99. マルチコアプログラミング: 並列処理の課題と解決策。
100. スレッドダンプと分析: スレッドダンプを使用した問題の特定。

Web サーバーとロードバランシング (20 points)

101. Apache Tomcat 設定: コネクタ、context.xml、server.xml の設定。
102. Nginx リバースプロキシ: proxy_pass、アップストリームサーバー、ロードバランシングの設定。
103. HAProxy の高可用性: フェイルオーバーとセッション持続性の設定。

104. Web サーバーセキュリティ: SSL/TLS 設定、セキュリティヘッダー、ファイアウォールルール。
105. ロードバランシングアルゴリズム: ラウンドロビン、最小接続数、IP ハッシュ。
106. サーバーサイドキャッシュ: Varnish、Redis、メモリキャッシュの使用。
107. モニタリングツール: Prometheus、Grafana、New Relic を使用したサーバーモニタリング。
108. プロダクションでのロギング: ELK スタックや Graylog を使用した集中型ロギング。
109. 水平スケーリング vs 垂直スケーリング: トレードオフと使用例。
110. Web サーバーパフォーマンス調整: ワーカースレッド、接続タイムアウト、バッファの調整。
111. リバースプロキシキャッシュ: キャッシュヘッダーと期限の設定。
112. Web サーバーロードテスト: Apache JMeter、Gatling などのパフォーマンステストツール。
113. SSL オフロード: ロードバランサーでの SSL/TLS 終了処理。
114. Web サーバーハードニング: セキュリティベストプラクティスと脆弱性評価。
115. 動的コンテンツ vs 静的コンテンツの提供: サーバー設定の最適化。
116. Web サーバークラスタリング: 高可用性のためのクラスター設定。
117. Web サーバー認証: 基本認証、ダイジェスト認証、OAuth 認証の実装。
118. Web サーバーログ形式: 一般的なログ形式と解析ツール。
119. Web サーバーリソース制限: 接続数、リクエスト、バンド幅の制限設定。
120. Web サーバーバックアップとリカバリ: 災害復旧のための戦略。

CI/CD と DevOps (20 points)

121. Jenkins パイプラインコード: CI/CD パイプラインのための Jenkinsfiles の作成。
122. Docker コンテナ化: Dockerfile 作成、マルチステージビルド、コンテナオーケストレーション。
123. Kubernetes オーケストレーション: デプロイメント、サービス、ポッド、スケーリング戦略。
124. GitOps の原則: Git を使用したインフラと設定管理。
125. Maven と Gradle ビルドツール: 依存関係管理、プラグイン、ビルドライフサイクル。
126. ユニットテストと統合テスト: JUnit、Mockito、TestNG を使用したテストの作成。
127. コードカバレッジツール: Jacoco を使用したコードカバレッジの測定。
128. 静的コード解析: SonarQube などのツールを使用したコード品質チェック。
129. インフラコード (IaC): Terraform、CloudFormation を使用したインフラプロビジョニング。
130. ブルーグリーンデプロイメント: デプロイメント中のダウンタイムの最小化。

131. キャナリーデプロイメント: 新機能の段階的なロールアウト。
132. CI パイプラインでの自動テスト: ビルドステージにテストの統合。
133. 環境管理: Ansible、Chef、Puppet を使用した設定管理。
134. CI/CD ベストプラクティス: 継続的インテグレーション、継続的デプロイメント、継続的デリバリー。
135. ロールバック戦略: デプロイメント失敗時の自動ロールバックの実装。
136. セキュリティスキャン: SAST、DAST などのセキュリティチェックをパイプラインに組み込む。
137. マイクロサービス向け CI/CD パイプライン: 複数サービスのパイプライン管理。
138. CI/CD パイプラインのモニタリング: パイプライン失敗とパフォーマンス問題のアラート。
139. DevOps ツールエコシステム: Docker、Kubernetes、Jenkins、Ansible などのツールの理解。
140. クラウドネイティブアプリケーション向け CI/CD: クラウドプラットフォームへのアプリケーションデプロイ。

設計パターンとベストプラクティス (20 points)

141. シングルトンパターン: スレッドセーフなシングルトンの実装。
142. ファクトリーパターン: 具体的なクラスを指定せずにオブジェクトを作成。
143. ストラテジーパターン: アルゴリズムのカプセル化と切り替え。
144. SOLID 原則: シングルレスポンス、オープン/クローズ、リスコフ置換、インターフェース分離、依存性逆転の理解と適用。
145. 依存性注入: 結合を減らし、コードの保守性を高める。
146. イベントソーシングパターン: アプリケーション状態を再構築するためのイベントの保存。
147. CQRS アーキテクチャ: コマンドとクエリの責任を分離。
148. スケーラビリティ設計: 水平スケーリング、シャーディング、ロードバランシングの使用。
149. コードリファクタリング技術: メソッドの抽出、変数のリネーム、条件の簡素化。
150. クリーンコード実践: 読みやすく、保守しやすく、自己文書化されるコードの書き方。
151. テスト駆動開発 (TDD): 実装前にテストを書く。
152. コードバージョニング: GitFlow、Trunk-Based Development などのブランチ戦略。
153. 保守性設計: モジュール設計、関心事の分離。
154. 避けるべきアンチパターン: God クラス、スパゲッティコード、緊密な結合。
155. セキュリティ設計: 最小権限、防御の深さの実装。

156. パフォーマンス設計: アルゴリズムの最適化、I/O 操作の減少。
157. 信頼性設計: 冗長性、容錯性、エラー処理の実装。
158. 拡張性設計: プラグイン、拡張機能、オープン API の使用。
159. 使用性設計: API が直感的でよく文書化されていることを確保。
160. テスト可能性設計: テストしやすくモックしやすいコードの書き方。

セキュリティ (20 points)

161. OAuth2 と JWT: トークンベースの認証の実装。
162. ロールベースアクセス制御 (RBAC): ユーザーに役割と権限を割り当てる。
163. セキュリティヘッダー: Content Security Policy、X-Frame-Options の実装。
164. SQL インジェクション防止: 準備されたステートメントとパラメータ化クエリの使用。
165. クロスサイトスクリプティング (XSS) 保護: 入力と出力のサニタイズ。
166. 暗号化と復号: AES、RSA を使用したデータ保護。
167. セキュアなコーディング実践: バッファオーバーフローなどの一般的な脆弱性を避ける。
168. オーディットトレイルの実装: ユーザーアクションとシステムイベントのログ記録。
169. 敏感データの取り扱い: ハッシュアルゴリズムを使用したパスワードの安全な保存。
170. 法令順守: GDPR、PCI-DSS などのデータ保護法令。
171. 二要素認証 (2FA) の実装: セキュリティの追加層を追加。
172. セキュリティテスト: 侵入テスト、脆弱性評価。
173. セキュアな通信プロトコル: データ暗号化のための SSL/TLS の実装。
174. セキュアなセッション管理: セッショントークンとタイムアウトの管理。
175. Web アプリケーションファイアウォール (WAF) の実装: 一般的な攻撃からの保護。
176. セキュリティモニタリングとアラート: SIEM などのツールを使用した脅威検出。
177. マイクロサービスにおけるセキュリティベストプラクティス: サービス間通信のセキュリティ保護。
178. CAPTCHA の実装: ボット攻撃の防止。
179. CI/CD パイプラインにおけるセキュリティ: ビルド中の脆弱性スキャン。
180. セキュリティバイデザイン: 開発プロセスの初期段階からセキュリティを組み込む。