**Core Java (Concepts, Collections, Multi-threading, Exception Handling)**

1. **What are the main differences between HashMap and ConcurrentHashMap?**
   * **Answer:** HashMap is not thread-safe and may lead to concurrency issues, whereas ConcurrentHashMap is thread-safe and allows concurrent modifications by multiple threads without locking the entire map.
2. **How does the Java Memory Model handle multi-threading?**
   * **Answer:** The Java Memory Model (JMM) ensures visibility, atomicity, and ordering of operations. It uses synchronized blocks, volatile variables, and locks to manage thread access to shared variables.
3. **What is the difference between checked and unchecked exceptions?**
   * **Answer:** Checked exceptions are checked at compile-time (e.g., IOException), whereas unchecked exceptions are checked at runtime (e.g., NullPointerException).
4. **How does the ExecutorService help in Java multi-threading?**
   * **Answer:** ExecutorService provides a thread pool to efficiently manage and execute multiple tasks asynchronously instead of creating new threads manually.

**Spring Boot (Microservices, REST API, Security, JPA, Testing)**

1. **What is the role of Spring Boot in microservices?**
   * **Answer:** Spring Boot simplifies microservices development by providing built-in features like embedded servers, auto-configuration, and production-ready monitoring tools (Actuator).
2. **How does Spring Security implement authentication and authorization?**
   * **Answer:** Spring Security uses filters, authentication providers, and role-based access control (RBAC) to handle authentication (username/password, OAuth) and authorization (roles, permissions).
3. **What is the difference between JPA and Hibernate?**
   * **Answer:** JPA is a specification for ORM, while Hibernate is an implementation of JPA that provides additional features like caching and lazy loading.
4. **How do you test a Spring Boot REST API?**
   * **Answer:** Use tools like JUnit, Mockito, and Spring Boot Test annotations (@WebMvcTest, @SpringBootTest) to write unit and integration tests.

**Database (CRUD Queries, Performance Tuning)**

1. **What is the difference between INNER JOIN and LEFT JOIN?**
   * **Answer:** INNER JOIN returns only matching records from both tables, whereas LEFT JOIN returns all records from the left table and matching records from the right table.
2. **How can you optimize a slow SQL query?**
   * **Answer:** Indexing, query optimization techniques like avoiding SELECT \*, using proper joins, and analyzing query execution plans.
3. **What are ACID properties in databases?**
   * **Answer:** Atomicity, Consistency, Isolation, and Durability ensure reliable transaction processing.
4. **What is database connection pooling, and why is it important?**
   * **Answer:** Connection pooling reuses existing database connections instead of creating new ones, reducing overhead and improving performance.

**Problem-Solving & Algorithms (Data Structures, Sorting, Complexity Analysis)**

1. **What is the time complexity of QuickSort in the worst case?**
   * **Answer:** The worst-case time complexity of QuickSort is **O(n²)** when the pivot selection is poor.
2. **How does a HashMap achieve constant-time lookup?**
   * **Answer:** HashMap uses a hashing function to compute an index and stores key-value pairs in an array, allowing O(1) average time complexity.
3. **What is the difference between Breadth-First Search (BFS) and Depth-First Search (DFS)?**
   * **Answer:** BFS explores level by level (queue-based), while DFS explores deep before backtracking (stack-based).
4. **How does dynamic programming differ from greedy algorithms?**
   * **Answer:** Dynamic programming solves subproblems and stores results (memoization), whereas greedy algorithms make local optimal choices without considering future consequences.

**Troubleshooting & Debugging (Log Analysis, Issue Diagnosis)**

1. **What are common logging levels in Java?**
   * **Answer:** TRACE, DEBUG, INFO, WARN, ERROR, and FATAL indicate different levels of severity.
2. **How can you debug a memory leak in a Java application?**
   * **Answer:** Use tools like VisualVM, JProfiler, or Java Flight Recorder to analyze heap dumps and identify objects with excessive memory usage.
3. **What is thread dump analysis, and when is it useful?**
   * **Answer:** A thread dump provides a snapshot of all active threads and their states, helping diagnose deadlocks and performance bottlenecks.
4. **How do you troubleshoot high CPU utilization in a microservices environment?**
   * **Answer:** Monitor logs, use APM tools (New Relic, Prometheus), analyze thread dumps, and check database query performance.

**System Design & Architecture (Microservices Patterns, Distributed Systems, Scalability)**

1. **What is the CAP theorem in distributed systems?**
   * **Answer:** The CAP theorem states that a distributed system can provide only two out of three guarantees: Consistency, Availability, and Partition Tolerance.
2. **How does event-driven architecture work in microservices?**
   * **Answer:** Services communicate asynchronously using event brokers like Kafka or RabbitMQ, reducing direct dependencies.
3. **What are the benefits of API Gateway in microservices?**
   * **Answer:** API Gateway handles authentication, request routing, rate limiting, and load balancing in a microservices environment.
4. **What is the difference between horizontal and vertical scaling?**
   * **Answer:** Horizontal scaling (scaling out) adds more machines, while vertical scaling (scaling up) increases resources in a single machine.

**Code Quality & Best Practices (Design Patterns, Unit Testing, Resource Management)**

1. **What is the Singleton pattern, and when should it be used?**
   * **Answer:** The Singleton pattern ensures only one instance of a class exists and is used for managing shared resources like database connections.
2. **What are mocks and stubs in unit testing?**
   * **Answer:** Mocks simulate real objects with expectations, while stubs provide predefined responses to method calls.
3. **What is Dependency Injection (DI) in Spring?**
   * **Answer:** DI allows Spring to manage object creation and dependencies, improving testability and reducing tight coupling.
4. **Why is SonarQube used in code quality checks?**
   * **Answer:** SonarQube detects code smells, vulnerabilities, and ensures adherence to coding standards.

**Performance Optimization (Space/Time Complexity, Query Optimization, Parallel Processing)**

1. **How do you optimize time complexity in an algorithm?**
   * **Answer:** Choose efficient data structures, use divide-and-conquer techniques, and avoid redundant calculations.
2. **What is caching, and how does it improve performance?**
   * **Answer:** Caching stores frequently accessed data in memory (e.g., Redis) to reduce database calls and improve response time.
3. **What is parallel processing, and when should it be used?**
   * **Answer:** Parallel processing executes multiple tasks simultaneously, improving performance in compute-intensive applications.
4. **How does indexing improve database performance?**
   * **Answer:** Indexing speeds up search operations by maintaining a sorted structure for quick lookups.

**Set 1: General Java and Core Concepts**

**1. Differences Between Singleton, Factory, and Builder Design Patterns**

**Answer:**

* **Singleton** ensures only one instance of a class exists.
* **Factory** provides a method to create objects without specifying the exact class.
* **Builder** is used for constructing complex objects step by step.

**2. How Does QuickSort Differ from MergeSort in Terms of Time and Space Complexity?**

**Answer:**

* **QuickSort:** Avg case O(n log n), worst-case O(n²), space O(log n).
* **MergeSort:** Always O(n log n), space O(n).

**3. Explain How HashMap Works in Java**

**Answer:**

* Uses an array of buckets and a hash function to store key-value pairs.
* Handles collisions using chaining (linked list or tree).
* Retrieves values in O(1) average time complexity.

**4. What is the Difference Between HashSet, LinkedHashSet, and TreeSet?**

**Answer:**

* **HashSet:** Unordered, O(1) operations.
* **LinkedHashSet:** Maintains insertion order, O(1) operations.
* **TreeSet:** Sorted order, O(log n) operations.

**Set 2: Advanced Java and Spring Boot**

**5. How Would You Identify and Resolve Performance Bottlenecks in a Spring Boot Microservice?**

**Answer:**

* Use Spring Actuator, JMeter, VisualVM, YourKit for analysis.
* Optimize database queries, implement caching (Redis, Ehcache).
* Use connection pooling (HikariCP) and enable lazy loading.

**6. Explain the Difference Between Eager and Lazy Loading in Hibernate**

**Answer:**

* **Lazy Loading (Default):** Loads data only when accessed. Used when related data is rarely needed.
* **Eager Loading:** Loads all related entities upfront. Used when related data is always required.

**7. What is Dependency Injection and How Does Spring Implement It?**

**Answer:**

* Injects dependencies instead of creating them manually.
* Spring uses Constructor Injection, Setter Injection, and Field Injection.

**8. How Does Spring Security Handle Authentication and Authorization?**

**Answer:**

* Uses filters, JWT tokens, and OAuth2.
* Authentication managed via UserDetailsService.

**Set 3: Practical and System Design**

**9. How Would You Design a URL Shortener Like Bit.ly?**

**Answer:**

* Use Base62 encoding to generate short unique IDs.
* Database partitioning to handle large-scale requests.
* Implement caching (Redis) for frequently accessed URLs.
* Rate limiting to prevent abuse.

**10. How Would You Design a Real-Time Notification System for a Banking App?**

**Answer:**

* Use Firebase, APNs for push notifications.
* Implement event-driven architecture (Kafka, RabbitMQ).
* Maintain user preferences (email, SMS, push).

**11. How Would You Optimize a Slow-Running REST API?**

**Answer:**

* Use caching (Redis, CDN, HTTP Cache-Control).
* Optimize database queries (Indexing, Query tuning, Avoid N+1 problem).
* Implement pagination for large datasets.
* Asynchronous processing for long-running tasks.
* Use Gzip compression for smaller payloads.

**12. How Would You Manage Transactions in Spring Boot With Multiple Database Operations?**

**Answer:**

* Use @Transactional for atomic operations.
* Define appropriate propagation levels (REQUIRED, REQUIRES\_NEW, NESTED).
* Handle rollback scenarios effectively.

**Set 4: REST API and Microservices**

**13. Explain RESTful APIs and How They Differ from SOAP**

**Answer:**

* REST (Representational State Transfer) is an architectural style using HTTP methods (GET, POST, etc.).
* Unlike SOAP, REST is lightweight, uses JSON or XML, and is stateless.
* REST constraints: Statelessness, Uniform Interface, Cacheability, etc.

**14. How Would You Secure REST APIs in Spring Boot?**

**Answer:**

* Implement JWT-based authentication (OAuth2, OpenID Connect).
* Use Spring Security for role-based access control (@PreAuthorize).
* Enable HTTPS & TLS encryption.
* Rate limit requests using API Gateway.

**15. How Do Microservices Communicate With Each Other?**

**Answer:**

* REST, gRPC, Kafka, RabbitMQ, Service Mesh (Istio).

**16. How Does API Gateway Work in a Microservices Architecture?**

**Answer:**

* Manages authentication, rate limiting, and routing.
* Examples: Kong, Zuul, API Gateway.