1. Core Java

1. **What are the key differences between HashMap and ConcurrentHashMap?**
   - `HashMap` is not thread-safe, while `ConcurrentHashMap` allows concurrent modifications.
2. **Explain the difference between `final`, `finally`, and `finalize` in Java.**
   - `final`: Prevents modification of variables, methods, or classes.
   - `finally`: Used in try-catch-finally blocks for cleanup.
   - `finalize()`: Called by the garbage collector before an object is destroyed.
3. **What is the difference between `String`, `StringBuffer`, and `StringBuilder`?**
   - `String`: Immutable.
   - `StringBuffer`: Mutable, thread-safe (synchronized).
   - `StringBuilder`: Mutable, not thread-safe (faster).
4. **Explain the difference between checked and unchecked exceptions.**
   - Checked exceptions must be handled (IOException), while unchecked exceptions (RuntimeException) don't require explicit handling.
5. **What is the difference between shallow copy and deep copy?**
   - A shallow copy copies object references, whereas a deep copy clones the actual objects.
6. **What is the purpose of the `volatile` keyword in Java?**
   - Ensures visibility of changes to a variable across threads.
7. **How does garbage collection work in Java?**
   - JVM automatically removes unused objects using various GC algorithms like G1, Parallel, and CMS.
8. **What is the difference between an interface and an abstract class?**
   - An interface has only method declarations, while an abstract class can have both declarations and definitions.
9. **What is a lambda expression in Java?**
   - A lambda expression provides a concise way to implement functional interfaces (`(a, b) -> a + b`).

10. **What are Java Streams?**
    - Streams process data in a functional style (`stream().map().filter().collect()`).

2. Spring Framework (Spring Boot, Spring MVC, Spring Security, etc.)

1. **What is the difference between `@Component`, `@Service`, and `@Repository`?**
    - All are Spring-managed beans, but `@Service` is for business logic, and `@Repository` is for database interactions.
2. **Explain dependency injection in Spring.**
    - Spring injects dependencies via Constructor, Setter, or Field Injection using `@Autowired`.
3. **How does Spring Boot simplify Spring configuration?**
    - Spring Boot provides auto-configuration, embedded servers, and starter dependencies.
4. **What is the use of `@Transactional` in Spring?**
    - Ensures database operations are atomic.
5. **What are Spring Boot Starters?**
    - Pre-configured dependencies for specific functionalities (e.g., `spring-boot-starter-web`).
6. **How does Spring Security handle authentication and authorization?**
    - Uses `UserDetailsService`, JWT, OAuth2, and Role-Based Access Control.
7. **What is `@RestController` in Spring Boot?**
    - A combination of `@Controller` and `@ResponseBody` to return JSON responses.
8. **What is Circuit Breaker in Spring Boot?**
    - Prevents failures from cascading using libraries like Resilience4J.
9. **What is the difference between `@RequestParam` and `@PathVariable`?**
    - `@RequestParam` extracts query parameters, whereas `@PathVariable` extracts path variables.

10. **How does Spring Boot handle application configuration?**
    - Using `application.properties` or `application.yml`.

3. Microservices Architecture

1. **What is a microservice?**
    - A small, independently deployable service that communicates via APIs.
2. **What are the advantages of microservices?**
    - Scalability, flexibility, fault isolation, and ease of deployment.
3. **How do microservices communicate?**
    - Using REST, gRPC, Kafka, or RabbitMQ.
4. **What is API Gateway?**
    - A single entry point for client requests, handling authentication, logging, and routing.
5. **What is Service Discovery?**
    - A mechanism where services dynamically register and discover each other using tools like Eureka.
6. **What are distributed transactions?**
    - Transactions spanning multiple microservices, handled using Saga patterns.
7. **What is a Sidecar pattern?**
    - Deploying auxiliary services alongside main microservices.
8. **How does Spring Boot support microservices?**
    - Using Spring Cloud (Eureka, Feign, Ribbon, Resilience4J, Config Server).
9. **What is a CQRS pattern?**
    - Separates read and write operations for better scalability.
10. **What is Blue-Green Deployment?**
    - A strategy to reduce downtime by maintaining two production environments.

4. Database and ORM (JPA, Hibernate, SQL, NoSQL)

1. **What is JPA?**

- Java Persistence API for ORM.

2. **What is the difference between `fetchType.LAZY` and `fetchType.EAGER`?**
   - `LAZY`: Fetches only when accessed.
   - `EAGER`: Loads related entities immediately.
3. **What is the N+1 query problem in Hibernate?**
   - When one query loads data, triggering additional queries for each related entity.
4. **How do you resolve the N+1 problem?**
   - Use `JOIN FETCH` or `@BatchSize`.
5. **What is ACID in databases?**
   - Atomicity, Consistency, Isolation, Durability.
6. **What is the difference between SQL and NoSQL?**
   - SQL is relational, while NoSQL supports flexible, schema-less structures.
7. **What is optimistic vs pessimistic locking?**
   - Optimistic: Assumes no conflicts.
   - Pessimistic: Locks data for exclusive use.
8. **How does indexing improve database performance?**
   - Reduces search time by maintaining a sorted data structure.
9. **What is a composite key?**
   - A primary key made up of multiple columns.
10. **What is `@OneToMany` and `@ManyToOne` in JPA?**
    - Defines a one-to-many and many-to-one relationship between entities.

5. Concurrency and Multithreading

1. **What is a thread pool?**
   - A collection of worker threads for executing tasks efficiently.
2. **What is the difference between `synchronized` and `Lock`?**
   - `synchronized` is implicit locking, while `Lock` provides better control.

3. **How does `ThreadLocal` work?**
    - Stores data per thread for isolation.
4. **What are daemon threads?**
    - Low-priority threads that run in the background.
5. **What is a race condition?**
    - When multiple threads access shared resources unpredictably.
6. **What is `Callable` in Java?**
    - Similar to `Runnable`, but returns a result.
7. **What is `CompletableFuture`?**
    - Supports asynchronous programming with Java 8+ features.
8. **What is deadlock in Java?**
    - A state where two or more threads block each other indefinitely.
9. **What is the difference between `notify()` and `notifyAll()`?**
    - `notify()`: Wakes up one waiting thread.
    - `notifyAll()`: Wakes up all waiting threads.
10. **What is Fork/Join Framework?**
    - A framework for parallel processing using task splitting.

6. Design Patterns & Best Practices

1. **What is the Singleton pattern?**
    - Ensures only one instance of a class exists, commonly implemented using `private static` and `getInstance()`.
2. **What is the Factory pattern?**
    - Creates objects without exposing instantiation logic using an interface.
3. **How does the Strategy pattern work?**
    - Defines a family of algorithms and lets clients choose the desired implementation at runtime.
4. **What is the Observer pattern?**
    - Enables an object (subject) to notify multiple observers of state changes.

5. **Explain Dependency Injection and its benefits.**
   - A technique where dependencies are injected instead of hardcoded, promoting flexibility and testability.
6. **What is the difference between the Builder and Prototype patterns?**
   - `Builder`: Step-by-step object construction.
   - `Prototype`: Cloning an existing object.
7. **What is the Adapter pattern?**
   - Acts as a bridge between incompatible interfaces.
8. **How does the Command pattern work?**
   - Encapsulates requests as objects for better undo/redo operations.
9. **What are SOLID principles?**
   - A set of five principles ensuring maintainable and scalable code.
10. **What is the Circuit Breaker pattern?**
    - Prevents system failures from cascading by stopping calls to a failing service.

7. Cloud & DevOps (Docker, Kubernetes, CI/CD)

1. **What is Docker, and how is it used in Java applications?**
   - A containerization tool that packages applications with dependencies for consistent environments.
2. **What is Kubernetes, and how does it relate to Docker?**
   - Kubernetes orchestrates containerized applications, handling scaling and networking.
3. **What is a Pod in Kubernetes?**
   - The smallest deployable unit that contains one or more containers.
4. **What are ConfigMaps and Secrets in Kubernetes?**
   - Used to store configuration data and sensitive information separately.
5. **What is a Helm chart?**
   - A package manager for Kubernetes applications.
6. **How does a CI/CD pipeline work?**
   - Automates build, test, and deployment processes for faster releases.

7. **What are the differences between a rolling update and a blue-green deployment?**
   - `Rolling update`: Gradual updates with zero downtime.
   - `Blue-green`: Two identical environments with traffic switching.
8. **What is Infrastructure as Code (IaC)?**
   - Managing infrastructure using code (Terraform, Ansible).
9. **What is the purpose of a Service Mesh?**
   - Manages service-to-service communication in microservices (Istio, Linkerd).
10. **What is observability in DevOps?**
    - Monitoring system health using logging, tracing, and metrics.

8. Messaging & Event-Driven Systems

1. **What is Apache Kafka?**
   - A distributed event streaming platform for real-time data processing.
2. **How does Kafka differ from RabbitMQ?**
   - Kafka is log-based (event streaming), while RabbitMQ is queue-based (message brokering).
3. **What is a Kafka topic and partition?**
   - A topic is a message category, and partitions allow parallel processing.
4. **What is a Kafka Consumer Group?**
   - A group of consumers that share the load of processing a topic.
5. **How does exactly-once delivery work in Kafka?**
   - By enabling idempotent producers and transactional consumers.
6. **What is Event Sourcing?**
   - Storing state changes as a sequence of events instead of overwriting state.
7. **What is a Dead Letter Queue (DLQ)?**
   - A queue for messages that fail to process after multiple attempts.
8. **How does Kafka handle scalability?**
   - By increasing partitions and distributing consumers.

9. **What is a Stream Processor in Kafka?**
    - A service that processes events in real time (`Kafka Streams`, `Flink`).
10. **How does RabbitMQ ensure message durability?**
    - Persistent queues, message acknowledgments, and clustering.

9. Testing (Unit, Integration, Performance)

1. **What is the difference between unit and integration testing?**
    - `Unit`: Tests individual components.
    - `Integration`: Tests how components interact.
2. **What is Mockito, and how is it used?**
    - A Java mocking framework for simulating dependencies in tests.
3. **What is Spring Boot Test?**
    - Provides testing utilities for Spring applications (`@SpringBootTest`).
4. **What is the difference between `@MockBean` and `@Mock` in Spring?**
    - `@MockBean`: Creates a mock bean in the Spring context.
    - `@Mock`: Pure Mockito mock, outside the Spring context.
5. **What is Testcontainers?**
    - A Java library for running database tests in Docker containers.
6. **How do you test REST APIs in Spring Boot?**
    - Using `MockMvc` for simulated HTTP requests.
7. **What is JMeter used for?**
    - Performance and load testing.
8. **What is contract testing in microservices?**
    - Ensuring API agreements between services (`Pact`, `Spring Cloud Contract`).
9. **What is Cucumber used for?**
    - Behavior-driven development (BDD) testing framework.
10. **How do you handle flaky tests?**

    - Retry mechanisms, better mocks, and improved test isolation.

10. Performance Optimization & Scalability

1. **How do you improve Java application performance?**
    - Optimized data structures, caching, efficient threading, and profiling.
2. **What is profiling in Java, and how is it done?**
    - Analyzing runtime performance using tools like JProfiler and VisualVM.
3. **What is the purpose of caching in applications?**
    - Reducing database calls and improving response times (`Redis`, `Ehcache`).
4. **What is connection pooling?**
    - Reusing database connections to improve efficiency (`HikariCP`).
5. **What is lazy loading, and why is it useful?**
    - Deferring object loading until needed to optimize performance.
6. **What is pagination, and why is it important?**
    - Loading data in chunks to prevent excessive memory usage.
7. **What is Load Balancing?**
    - Distributing traffic across multiple servers to improve availability.
8. **What is rate limiting?**
    - Controlling API usage to prevent abuse and overloading.
9. **What is a CDN, and how does it help performance?**
    - A Content Delivery Network caches static assets globally for faster access.
10. **What is the CAP theorem?**
    - A distributed system can only guarantee two of the three: Consistency, Availability, Partition Tolerance.

---

1. Core Java - Expert Level

1. **How does the Java Memory Model (JMM) work, and how does it handle visibility and ordering of variables?**

   - JMM defines how threads interact through memory, ensuring atomicity, visibility (happens-before), and ordering constraints.

2. **Explain the difference between biased locking, lightweight locking, and heavyweight locking in Java.**
   - Biased: Single-thread optimization.
   - Lightweight: CAS-based spinning lock.
   - Heavyweight: OS-based monitor lock.
3. **What are the differences between `ForkJoinPool` and `ExecutorService`?**
   - `ForkJoinPool`: Optimized for recursive parallelism using work-stealing.
   - `ExecutorService`: General-purpose thread pool management.
4. **How does the Java Garbage Collector handle memory fragmentation?**
   - Through compaction (G1 GC) and region-based allocations.
5. **What is the difference between `ReentrantLock` and `synchronized`?**
   - `ReentrantLock` provides better flexibility, fairness, and condition variables.
6. **Explain how VarHandles improve concurrency performance over Atomic classes.**
   - Direct low-level memory access using `unsafe` operations.
7. **What is the difference between `CompletableFuture.allOf()` and `CompletableFuture.anyOf()`?**
   - `allOf()`: Waits for all futures.
   - `anyOf()`: Completes when any future finishes.
8. **How does the `StampedLock` improve read performance compared to `ReentrantReadWriteLock`?**
   - It allows optimistic reads to reduce contention.
9. **What are memory barriers, and how does Java enforce them?**
   - Instructions preventing CPU reordering, enforced using `volatile`, locks, and `Unsafe.fullFence()`.
10. **How do you implement a lock-free data structure in Java?**
    - Using `AtomicReference` with CAS operations (`compareAndSet()`).

2. Spring Boot & Spring Framework - Expert Level

1. **How does Spring Boot auto-configuration work internally?**
   - Uses `@ConditionalOnClass`, `@ConditionalOnProperty`, and `spring.factories` to enable beans dynamically.
2. **What is the difference between `@ComponentScan` and `@Import`?**
   - `@ComponentScan`: Scans packages for beans.
   - `@Import`: Manually registers specific configurations.
3. **How does Spring Boot support reactive programming?**
   - Uses `WebFlux`, `Project Reactor`, and `Mono/Flux`.
4. **How does Spring Security handle OAuth2 authentication?**
   - Using `OAuth2LoginConfigurer`, `JwtDecoder`, and `OAuth2AuthorizedClientService`.
5. **Explain the purpose of `@Primary`, `@Qualifier`, and `@Bean` annotations in dependency injection.**
   - `@Primary`: Default bean resolution.
   - `@Qualifier`: Specific bean selection.
   - `@Bean`: Defines custom bean creation.
6. **How do you integrate Spring Boot with GraphQL?**
   - Using `spring-boot-starter-graphql` with resolver methods.
7. **What is a `BeanPostProcessor`, and how does it differ from a `BeanFactoryPostProcessor`?**
   - `BeanPostProcessor`: Modifies beans after instantiation.
   - `BeanFactoryPostProcessor`: Modifies bean definitions before instantiation.
8. **How does Spring Boot handle database migrations?**
   - Uses `Flyway` and `Liquibase` for schema versioning.
9. **What is a `DelegatingFilterProxy` in Spring Security?**
   - Bridges Java EE filters and Spring-managed security filters.

10. **How does `@Transactional` work under the hood?**
    - Uses dynamic proxies (`JDK` or `CGLIB`) to manage transactions.

3. Microservices Architecture - Expert Level

1. **How do microservices communicate in a reactive, event-driven architecture?**
    - Using Kafka, RabbitMQ, and WebSockets for async messaging.
2. **What is a Sidecar pattern in microservices?**
    - Deploying auxiliary services (logging, monitoring) alongside primary services.
3. **What are the benefits of a Service Mesh in microservices?**
    - Provides observability, security, and traffic management (`Istio`, `Linkerd`).
4. **How does API Gateway handle rate limiting and circuit breaking?**
    - Using `RateLimiter` filters (Redis-based) and `Resilience4j` circuit breakers.
5. **What is the Strangler Fig pattern?**
    - Incremental migration from monolith to microservices.
6. **How do you handle distributed transactions in microservices?**
    - Using SAGA (choreography or orchestration) and 2PC (Two-Phase Commit).
7. **What is Consul, and how does it handle service discovery?**
    - A distributed service registry using health checks and key-value storage.
8. **What is the role of OpenTelemetry in microservices?**
    - Provides distributed tracing and observability.
9. **What is an API Composition pattern?**
    - Aggregates multiple microservice responses in a single API call.
10. **How does Spring Cloud Sleuth work for tracing?**
    - Adds trace IDs to logs for distributed tracing.

4. Cloud & DevOps - Expert Level

1. **What is Kubernetes Horizontal Pod Autoscaler (HPA)?**

   - Automatically scales pods based on CPU, memory, or custom metrics.

2. **How does GitOps differ from traditional CI/CD?**
   - Uses Git as the single source of truth for deployments (`ArgoCD`, `FluxCD`).
3. **What is a Kubernetes StatefulSet, and when should it be used?**
   - Manages stateful applications with stable network identities.
4. **How does Nginx handle reverse proxying and load balancing?**
   - Uses `proxy_pass`, `upstream`, and `sticky sessions`.
5. **What is an eBPF, and how does it enhance security in cloud environments?**
   - Allows kernel-level monitoring without modifying code (`Falco`, `Cilium`).
6. **What are the differences between AWS Fargate and Kubernetes?**
   - `Fargate`: Serverless container execution.
   - `Kubernetes`: Custom orchestration control.
7. **How do you configure multi-tenancy in Kubernetes?**
   - Using namespaces, RBAC, and network policies.
8. **What is the difference between Canary and Blue-Green deployments?**
   - `Canary`: Gradual rollout to a subset.
   - `Blue-Green`: Full switch between environments.
9. **What is Chaos Engineering?**
   - Introducing controlled failures to test system resilience (`Gremlin`, `Litmus`).
10. **How does Terraform differ from Ansible in IaC?**
    - Terraform is declarative, Ansible is procedural.