Java & Spring Boot Knowledge Base (KB)

Core Java

- OOP: Encapsulation, Abstraction, Inheritance, Polymorphism

- Streams API: Functional-style operations; filter/map/reduce; Java 17 adds pattern matching

- HashMap: Buckets, collision handling, linked list → tree conversion in Java 8

- Exceptions: Checked vs unchecked; best practices for handling

- Multithreading: Thread, Runnable, ExecutorService, locks, thread-safe collections

- CompletableFuture: Async operations, chaining, combining futures

- Virtual Threads (Java 21): Lightweight threads for massive concurrency

- Records: Immutable data carriers, reduce boilerplate

- Sealed Classes: Restrict subclassing

- Pattern Matching: Simplifies type checks

Spring Boot

- Dependency Injection: Constructor/Setter/Field injection, @Autowired/@Service

- REST API: @RestController, @GetMapping/@PostMapping, ResponseEntity

- Exception Handling: @ControllerAdvice, @ExceptionHandler

- Transactions: Propagation levels (REQUIRED, REQUIRES_NEW, NESTED), isolation levels

- Security: OAuth2, JWT, Spring Security filters

- Profiles: Environment-specific configs, @Profile

- Spring Data JPA: EAGER vs LAZY, entity relationships

- Scheduling & Async: @Async, @Scheduled, thread pools

Databases & Caching

- SQL vs NoSQL: ACID vs horizontal scaling, flexible schema

- Redis Caching: @Cacheable, @CacheEvict, configuration best practices

- Indexing & Optimization: Use proper indexes, query tuning

- Transactions & Isolation: READ_UNCOMMITTED, READ_COMMITTED, REPEATABLE_READ, SERIALIZABLE

Messaging & Microservices

- Kafka vs RabbitMQ: Delivery, persistence, ordering, use cases

- Event Ordering: Kafka partition-level, RabbitMQ queue-level

- SAGA Pattern: Distributed transactions with local commits + compensating actions

- REST vs gRPC: Protocols, performance, streaming, binary vs JSON

- API Versioning: URL, Header, Query param strategies

System Design & Architecture

- Monolith vs Microservices: Deployments, scaling, resilience

- High Availability & Scalability: Load balancing, caching, clustering

- Idempotency: Safe retries, e.g., PUT with transaction ID

- Distributed Logging & Observability: Trace IDs, ELK/Prometheus/Grafana

- Design Example: Chat service microservices, PostgreSQL sharding, Kafka events, Redis cache, JWT auth

Advanced Concepts

- Locks & Synchronization: ReentrantLock, ReadWriteLock, synchronized blocks

- Thread-safe Collections: ConcurrentHashMap, CopyOnWriteArrayList

- Spring Cloud Config: Centralized config for microservices

- Feign Client: Declarative REST client

- Circuit Breakers: Resilience4j/Hystrix to prevent cascading failures

- Async Messaging: @KafkaListener, @KafkaTemplate

- Lazy vs Eager Loading: FetchType.LAZY vs FetchType.EAGER

- Batch Processing: Spring Batch

Testing & Tools

- JUnit 5 & Mockito: Unit testing, mocking

- Integration Testing: Testcontainers, in-memory DBs

- OpenAPI/Swagger: API documentation

- Performance & Profiling: Spring Boot actuator, JVM GC tuning, memory leak detection