LLD INTERVIEW TEMPLATE (JAVA / OOP)

1. Understand the Problem Clearly
- Ask clarifying questions.
- Identify scope (MVP or full scale).
- Understand data persistence and concurrency needs.

2. List Requirements
Functional:
- Core system features and behaviors.

Non-Functional:
- Performance, scalability, concurrency, fault-tolerance.

3. Identify Actors and Their Use Cases
Example:
| Actor | Use Cases |
|---|---|
| User | Book, Cancel |
| Admin | Configure system |
| System | Generate IDs, Notify |

4. Derive Key Entities (Classes)
- Nouns -> Classes (User, Ticket, Vehicle)
- Verbs -> Methods (book(), cancel())

5. Define Class Responsibilities (SRP)
Use a table to keep roles clear.

6. Draw Class Diagram (UML-style)
- Attributes + Methods
- Arrows: composition, aggregation, inheritance

7. Design Patterns (Optional)
| Pattern | Usage Example |
|---|---|
| Singleton | FeeCalculator |
| Strategy | Multiple pricing rules |
| Factory | Dynamic object creation |
| Observer | Notify on events |

8. Discuss Tradeoffs & Extensibility
- Easy to add new features?
- Handle concurrency or persistence?
- Separation of concerns?

9. Code MVP (if time)
- Define main classes and key methods.
- Keep code readable and logical.

10. Summarize Clearly
"Here is the system design with SRP, scalability, and key extensibility points. Open to feedback."

FLOW SUMMARY:
1. Understand problem
2. List requirements
3. Identify actors/use cases
4. Identify key classes
5. Define responsibilities
6. Draw class diagram
7. Think design patterns
8. Discuss tradeoffs
9. Code MVP if needed
10. Summarize

Good luck! You've got this!