General

General expectation from java service developer is that they should be strong in Data structure , fundamental, problem solving and they should know micro services, DevOps , messaging services and cloud ,other expectation are when complex problem /use case comes in front of them they should attempt and try to solve the problem by applying their design /problem solving skills, the idea of asking complex question is to see are developer attempting to solve the problem which they have never solved , if they are trying to solve what approach are they taking , how are they breaking a complex problem into small problem and applying design principle and pattern to solve it.

Sometimes, we are asking open ended question, the purpose of asking open ended question is, developer should answer based on his past experience, if developer answers in totality with confidence the probability of having hand-son become very high

Sometimes we are asking the ambiguous questions, the idea of asking ambiguous question is to check if developer is asking questions to clarify the requirement if requirement is not clear and then try to solve the problem instead of without understanding the requirement jumping on the solution.

Problem Solving

The idea of asking problem solving is to evaluate developer design skills , sometimes client not expecting concrete implementation , they want to understand what approach developer taking to solved the design problem , are they giving up or the attempting to solve the problem

Fundamental

In fundamental client expecting , developer should know purpose of all the keyword , static binding , dynamic binding , overloading rule , overriding rule in terms of access modifier, exception handling, impact of dynamic linking on performance , how to improve performance by using final keyword , what's default implementation of hash code and equal , cloning, immutability, advantage of immutability , importance of final in security , Exception handling rules

Data Structure

In Data structure the expectation is developer should know all basic data structure and how those basic data structure internally working , based on use case developer should answer question which data structure is best fit for given use case , how hashing concept is working in hash set and map , how to improve map /set performance by optimizing hash code , what are time complexity of different operation on data structure , how re sizing is happening in data structure , how to use comparable comparator , how to implement different sorting algorithm . How to optimize data structure by changing the capacity and load factor, could able to implement one data structure by using another, could able to answer how to implement data structure

Concurrent API

developer should know how concurrent hash map internally managing a lock how segmentation is working ,how many thread can work on concurrent hash map , benefit of using concurrent hash map over hash table and synchronize map , what kind of business use case can be implemented by using

concurrent hash map , how blocking queue is working what kind of problem can be solved by using blocking queue , when we should use linked blocking queue and when array blocking queue what's implementation of blocking queue , how to use blocking queue in inter thread communication, what's fail safe iterator

how to implement thread pool , what's advantage of thread pool , how many type of thread pool do we have , how we can use executor service , how to use executor service to implement parallel /pipe line processing. , what kind of business problem can be solved by cyclic barrier and count down latch and how its working, how to use semaphore, what's CAS concept (Compare and set), how atomic API is working internally

Multithreading

the expectation here is developer should know basic of multithreading , should know how wait , notify , sleep , join is working , how locking is working , what's class label lock , what's object lock, how to implement inter thread communication by using wait and notify , how volatile is working , how happen before concept is working in terms of volatile , how to implement thread pool in java 4 , how important is immutability in multithreading , what's code can create deadlock , what code can create starvation,

Serialization

Developer should know purpose of serialization, purpose of serial version UID, if serial version UID is not define how JVM generating it, how to customize serialization behavior, how to serialize transient variable how to improve performance by customizing serialization behavior

Memory management

Developer should know java memory model , should know heap , how garbage collection is working , how to optimize memory , should aware where class meta data storing in memory , should know reason of Perm gen Exception , reason of Out of memory exception , should aware how to do memory profiling , how to identify which code consuming memory

Design Pattern

Developer should know at least 2 to 3 design pattern thoroughly, while explaining use case implementation should use some of the design pattern , must know best way of implementing singleton pattern , factory pattern, strategy pattern, builder pattern, flyweight pattern, decorator and adapter pattern. Should know at-least 1 example of these patterns implementations from JDK.

Design Principle

Developer should know SOLID concept very well, whenever explaining solution design principle should reflect in his solution, how important code for interface concept is

Object oriented Concept

Developer should know Encapsulation, Polymorphic, Composition, Inheritance, when should use inheritance when should we use composition.

Database

Developer should able to write some of query on join and aggregation, should aware index, type of index and how indexing is working, should aware all key

Spring

Knows basics of spring like dependency injection (inversion of control), auto wiring (both XML and annotations), bean life cycle, profiling, transaction management and externalization of properties.

Rest

Basic understanding of REST principles (Uniform interface, Stateless interactions, Cacheable, Client-Server, Layered System, Code on Demand. HTTP protocol (HTTP methods, Headers, Error codes) and concept of resources for REST.

Micro-Services

What is MicroServices? How it is different from Monolithic architecture? What are the advantages in Micro services w.r.t Monolithic architecture? Small application using Spring boot.