1. Design patterns used in your project?

2. Design a Train & reservation system. Give class structure and design UML

3. How would you implement a synchronized Stack or a Queue in Java? Explain performance of your solution.

4. Describe CopyOnWriteArrayList? Where is it used in Java Applications?

5. How would you find kth highest number in a list of n Numbers? Explain time complexity of your solution.

6. How Heap space is divided in Java. How does Garbage Collector clean up the unused Objects?

7. What are database transaction Isolation levels?

8. What are different Bean Scopes in Spring?

Collections-

Good understanding of: - Commonly used collections like, ArrayList, Vector, LinkedList, Hashmap, HashTable, HashSet. - Able to write code using these collections and iterating through them. - equals and hashcode contract and its implications. - Usage of Comparable and Comparator. - Failsafe and failfast iterators and their impact when used on various collections	Good Understanding of: - Other collections apart from those mentioned in level 1 like : LinkedHashSet, TreeSet, PriorityQueue, LinkedHashMap, TreeMap, etc. and is able to explain their usage through examples like when to use what. - HashMap concepts like Hashing, Collision, Rehashing, LoadFatcor, etc - Is able to write code to sort objects using Comparable and Comparator.

Multithreading

Good understanding of: - Synchronization, sleep, wait, notify, notifyAll - DeadLock, Runnable At least basic understanding of: - Executor Framework - Concept of Callable and Future	Good understanding of: - ReentrantLock and Condition interface, and their usage. - Various implementations of ExecutorService interface. - Concurrent collections like BlockingQueue, ConcurrentMap their implementation details and usage. - Theoretical knowlege of fork/join, Latch, Barrier - Understanding of volatile keyword. - Concept of ThreadLocal. Has used atleast 3 classes of concurrent package in the past projects and is able to explain with examples.

Java Memory Mgmt

Understands finalize, initial/ max heap size, garbage collection basics, Class Loading basics	able to explain GC algos; concurrent/ parallel GC; full GC/ partial GC; Custom Class Loader

New JDK Features

Knowledge of basic Java 8 features - Default and Static methods in interfaces, Lambda Expressions, Streams, Functional interface, Method references. Also consider if the candidate has good understanding of functional programming in languages like Scala, etc.	Experience in writing code using Streams and Lambdas. The candidate should be able to write code using these features.

Design principles

Person should be able to talk about each of these with examples. ENCAPSULATION , ABSTRACTION, POLYMORPHISM, INHERITANCE.	Explain cases with examples where inheritance should be avoided and composition is preferred. (This should be done by providing a scenario / problem statement to the candidate)

Design Patterns	Knowledge of commonly used design patterns like singleton, factory, strategy, observer, command. Extensive knowledge of design patterns. Have used singleton, strategy, observer, adaptor, template, command, factory & other patterns. Can apply appropriate DP in a given situation.
OOPS & Design principles	talk about each of these with examples. ENCAPSULATION , ABSTRACTION, POLYMORPHISM, INHERITANCE. Should be able explain how he achieved object oriented programming in his / her last project with examples from the tracks he / she worked on. Should be able to explain low coupling and high cohesion and how it was achieved in last project across modules. What is programming to interfaces, with examples? Design principles for reusability, extensibility, maintainability: designing to interfaces, single-responsbility classes, OCP, coding to interfaces etc.
Data structures	Can implement stacks & queue with ease. Knowledge of hashed collections, & BST. Knowledge of hashed collections & trees. Can design & implement Hashtable & BST. Dictionary/ TRIE/ Shortest Path in Graph/ B+ trees etc.
Algorithms	Understands O(N) analysis; provides a reasonably optimal solution to a simple problem - generally involves using map/list DS; (array duplicity problem/ palindrome/ middle value of list, find element from list etc.)
CoreJava Fundamentals	(access modifiers, classes, interfaces, keywords, try-catch bocks, finalize(), inner-classes, generics, enums) Language fundamentals: Static, final, access modifiers, equals/hashcode, finalize, pass by value, primitive objects, immutability Interfaces, marker interfaces, Abstract/ inner/ static classes, anonymous classes Data Structures (Collection, List, Set, Map), comparator, comparable Serialization, object cloning , transient (what, why, how), Serializable/Externalizable Multi-threading (synchronized, wait/notify, volatile) JDK 1.5 (generics, enums, annotations) Exception handling (try/catch/finally, runtime/compile-time) JDBC (initiatilization, Connection, PreparedStatement, transaction mgmt., ResultSet) JUNIT (various asserts, testing exception conditions, testing collaboration among classes, testing using mocking frameworks)

Collections (lists, sets, maps, equals(), hashcode(), hashed collections, sorted collections)	aware of the contract between hashcode and equals. Can explain difference b/w various collections - ArrayList, LinkedList, HashSet. Knows about hashed & sorted collections. How to sort objects in java,
Multi-threading	familarity with inter-thread communication (wait, notify, join, interrupt), executor service, callable, futures, & synchronization. Strong handle on multithreading. Can talk about concurrent collections, syncronizers (semaphores, latches, barriers). Understanding of volatile keyword.
IO, Serialization & Cloning	Some experience with IO package. Knows about basic classes of the package. Knows about buffered collections. Good handle on IO package. Good handle on serializable, externalizable, transient variables & customizing serialization. Good handle on shallow & deep cloning.
Java Memory Management	Understands finalize, initial/ max heap size, garbage collection basics, Class Loading basics. able to explain GC algos; concurrent/ parallel GC; full GC/ partial GC; Custom Class Loader
SQL	Can create a query involving joins of two or more table. Understands various types of joins, group by & having clauses.
Database & JDBC	Knows about tables, PK, FK, various database objects - tables, sequences, views, database modeling, transaction management, ER diagrams + connection pools + various kind of statements in JDBC
Server Side Development	JMS, Messaging Web Services Spring ORM (Hibernate, JPA, iBatis)
Misc	Unit Testing Build scripts Software processes