**Refresher (Java, Spring Boot, Databases)**

**Basics of Core java**

* Exception handling.
  + Checked exception vs unchecked exception
  + Overriding the methods which throws checked or unchecked exceptions
* Generics
* Serialization
* **OOPs concepts**
* Immutable class
* Abstract class and interfaces
* **Encapsulation vs Abstraction**

**Data structures / Java Collections**

* Internals of HashMap, HashSet, TreeMap, TreeSet, LinkedList
* LRU cache implementation (Hint LinkedHashMap)
* Why there is a hash method inside the HashMap class?

(Hint: It’s not for key.hashcode() % bucket\_size)

* Uses of loadfactor and treefy threashold.
* Can HashSet contain a duplicate value?

(Yes, it can when equals and hashcode methods don’t follow the contract)

Find out what is the contract between them?

* Time complexity of all the operations on HashMap, HashSet, TreeMap, TreeSet, Stack and Queue for Best case, worst case and average case.

**SQL, DB2, Stored Procedures**

* Find the most recently hired emps in each department.

<https://www.w3resource.com/sql-exercises/employee-database-exercise/sql-subqueries-exercise-employee-database-52.php>

Follow up:

Can you optimize it further if we are fetching data in batches?

Can you solve it using proprietary functions like row\_number()

* Practice joins

<https://www.w3resource.com/sql/joins/sql-joins.php>

* Write a query in SQL to list the details of the departments where maximum number of employees are working.
* Write a query in SQL to find one employee from each department who is getting the highest salary within his department. (Don’t forget to handle corner case where multiple / no employees are getting highest salary in a department.)
* Write a query in SQL to list the details of the departments where maximum number of employees are working. (1 department with max employee or multiple departments having same number of employees)

**Microservices using spring webmvc, spring boot**

* **What is the use of service registry and discovery?**

* **Get familiar with usage of zookeeper, eureka and consul as a service discovery. Added advantage if you know trade off of all these solutions.**
* What are the key benefits of using Spring Boot?
  + Auto Configuration
  + Dependency management using starter parent
  + Actuator and Health Checks
  + Embedded Web Server
* How to implement Fault tolerance and circuit breaker?
* CAP theorem.
* What is the role of API Gateway in microservice architecture?
* How do you decide that a microservice should expose HTTP or JMS listener?
* **Microservices design patterns specially SAGA patterns, API Composition pattern and CQRS pattern**

**Spring Framework**

* AOP concepts like Join point, Advice etc.

**Problem Solving**

1. Implement a queue using stacks. Prepare for follow up questions also.
2. Find first and last positions of an element in a sorted array. (Observe the pattern like here array is sorted and it may contain duplicates)

<https://www.geeksforgeeks.org/find-first-and-last-positions-of-an-element-in-a-sorted-array/>

1. Detect and Remove Loop in a Linked List.

<https://www.geeksforgeeks.org/detect-and-remove-loop-in-a-linked-list/>

1. Add two numbers represented by linked lists.
2. Inorder Tree Traversal without Recursion.
3. How to design an LRU cache. Which data structure you would like to use?

Hint: LinkedHashMap Or HashMap + LinkedList

1. Find the k most frequent words from a file.

<https://www.geeksforgeeks.org/find-the-k-most-frequent-words-from-a-file/>

1. Print N’th node from the end of a Linked List.

<https://www.geeksforgeeks.org/nth-node-from-the-end-of-a-linked-list/>

1. Find K largest (or smallest) elements in an array.

There are multiple ways to solve this but using Min/max heap is the better solution.

<https://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/>

**Algorithms**

* Tim sort (Used by Java for objects)
* Quick Sort (Double Pivot Quick Sort is used by Java for primitives)
* Merge sort

**Multithreading / Concurrency**

* **Intrinsic and extrinsic locks**
* **How wait() and notify() works internally?**
* **Spurious wakeup and SpinLock**
* **Synchronization puzzle like print 1 to N sequentially from 2 threads one should be printing even numbers and one should be printing odd numbers.**
* **Deadlock prevention techniques**
* When we should prefer volatile int/long over the Atomic Integer / Atomic Long?
* When we should prefer Atomic Integer / Atomic Long over the intrinsic and extrinsic locks?
* How to Design a website visitor counter. Come up with at least 3 solutions and be aware about the pros and cons of each one.
* **Get Familiar with Semaphores, CountDownLatch and CyclicBarrier.**

<https://www.baeldung.com/java-semaphore>

<https://www.baeldung.com/java-cyclicbarrier-countdownlatch>

* Assume you have to replace the thread pool of any web application server using your custom thread pool executor. Try to build it and submit some tasks to it.

Do load testing of the same.

<https://howtodoinjava.com/java/multi-threading/java-thread-pool-executor-example/>

* Get familiar with class level and object level locks.

<https://howtodoinjava.com/java/multi-threading/object-vs-class-level-locking/>

**Design Principles / Patterns**

* SOLID Design Principles

Real world use cases, good to relate it with the work you have done

* Singleton Design pattern

Java vs Spring IOC, how to break it?

* Strategy Design Pattern
* Decorator Design Pattern
* Adaptor Design Pattern
* Builder Design Pattern
* Observer Design Pattern
* Template Design Pattern

**JVM Internals**

* **Diagnose Out of memory error**.
* Heap dumps, memory dumps