**Preparation Topics |**

Focus more on Problem solving,core Java and Multithreading.

Below are the topics for preparation :

|  |  |
| --- | --- |
| **Modules** | **Sub-Topics** |
| Core Java | OOPs, classes, interfaces, inner classes, Generics etc |
| Serialization, custom serialization |
| Collections (Linked HashSet, HashMap, TreeMap etc) |
| Multithreading (volatile, ThreadPool, Locks etc) |
| Data Structures | Lists, Trees, Graphs, Queues, Stacks etc |
| Problem Solving, Time complexity analysis |
| Design | Design principles and data structures in context of some real time problem |
| SQL | Basic SQL Queries |
| DB modelling, Query Tuning, Indexes, Qry Plans etc |
| Spring/ Other frameworks | Spring Core, Dependency Injection, Transaction Management etc |
| Soft Skills | Communication Skills, Adaptable, Never say die attitude |

Special focus on the below :

**Data Structures & Problem Solving:**

**1.    Data Structures:**

a.    Should be well versed with various data structures: Stacks, Queues, Trees, Graphs etc

b.    Should be able to compare various data structures e.g Stack vs Queue etc

c.     Should be able to create data structures e.g.

                                         i.    Queues using Stacks

                                        ii.    Max Depth of a Tree

                                       iii.    No of nodes in a Tree

                                       iv.    Reverse a list

                                         v.    Find Middle of a list

                                       vi.    Any many more these kinds of question….

**2.    Problem Solving:**

a.    Should be well versed with common Searching and Sorting Algorithms

b.    Should be able to think in terms of time complexity analysis

c.     Should be able to solve Problem Solving questions like:

                                         i.    Find duplicates in an array

                                        ii.    Find if String is a Palindrome

                                       iii.    Find max sub array within an integer array

                                       iv.    Any many more these kinds of question….

**Design**

1.    Should be comfortable with design principles like

a.    Programming to interfaces

b.    Composition over Inheritance

c.     Open-Closed Principle etc etc

2.    Should be comfortable with common design patterns like

a.    Singleton, factory etc

b.    Strategy, Adapter, Decorator, Command etc etc

3.    Should be able to apply design principles and design patterns in common real time problems e.g:

a.    Design an Air Traffic Controller System

b.    Or a Car Parking Lot system

c.     Or Producer Consumer Problem etc etc

4.    The focus here is on the how well design principles are adhered to and the appropriateness of the design patterns

Below are the areas for Multithreading :

a.    Must understand Object Level vs Class Level Locking

b.    Must understand difference among sleep, wait, yield, join etc.

c.     Must understand volatile keyword and its usage

                                         i.    Focus should be its difference with synchronization and atomic classes

d.    Must understand alternate strategies to synchronized keyword

e.    Must understand Executors Framework, ThreadPool etc

                                         i.    Should be able to design a ThreadPool without using executors framework

                                        ii.    Special focus should be on middleware to manage tasks while creating a threadpool e.g blocking queue.

                                       iii.    Should know how to size a threadpool

f.     Should be comfortable with the inner design of Multithreading Collection APIs like:

                                         i.    Concurrent Hash Map:

1.    Its internal Segmentation, Locking mechanism; Fail Fast, Fail Safe Iterators; Concurrent Modification Exception; Level of concurrency

                                        ii.    Synchronizers: Semaphores, Cyclicbarrier, Countdownlatches

1.    Should understand the internal details of these apis and should be able to create these apis on your own

g.    Need to know various ways to create a Thread: ThreadClass; Runnable; Callable etc

h.    Need to know what is deadlock; what to do if deadlock happens; how to analyze thread-dumps

Also go through the below links :

Kindly go through the following links and prepare for your next round :

I request you to prepare on the below areas:-

> <http://javahungry.blogspot.com/2013/08/hashing-how-hash-map-works-in-java-or.html>  
>  
> <http://javahungry.blogspot.com/2014/06/how-treemap-works-ten-treemap-java-interview-questions.html>  
>  
> <http://javahungry.blogspot.com/2015/02/how-concurrenthashmap-works-in-java-internal-implementation.html>  
>  
> <http://javahungry.blogspot.com/2015/10/how-treeset-works-internally-in-java-interview-questions.html>  
>  
> <http://javahungry.blogspot.com/p/threads.html>  
>  
> <http://javarevisited.blogspot.in/2011/04/synchronization-in-java-synchronized.html>  
>  
> <http://javarevisited.blogspot.in/2011/06/volatile-keyword-java-example-tutorial.html>  
>  
> <http://mrbool.com/working-with-java-executor-framework-in-multithreaded-application/27560>  
>  
> <http://stackoverflow.com/questions/10828863/what-the-use-of-custom-class-loader>  
>  
> <http://stackoverflow.com/questions/10901752/what-is-the-significance-of-load-factor-in-hashmap>  
>  
> <http://stackoverflow.com/questions/11011291/treeset-internally-uses-treemap-so-is-it-required-to-implement-hashcode-method>  
>  
> <http://stackoverflow.com/questions/137975/what-is-so-bad-about-singletons>  
>  
> <http://stackoverflow.com/questions/13855013/understanding-java-memory-management>  
>  
> <http://stackoverflow.com/questions/2087469/sort-a-file-with-huge-volume-of-data-given-memory-constraint>  
>  
> <http://stackoverflow.com/questions/27325997/how-does-countdownlatch-works-in-java>  
>  
> <http://stackoverflow.com/questions/56860/what-is-the-liskov-substitution-principle>  
>  
> <http://stackoverflow.com/questions/8161896/example-code-to-show-how-java-synchronized-block-works>  
>  
> <http://tutorials.jenkov.com/java-concurrency/synchronized.html>  
>  
> <http://tutorials.jenkov.com/java-util-concurrent/cyclicbarrier.html>  
>  
> <http://www.dynatrace.com/en/javabook/how-garbage-collection-works.html>  
>  
> <http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/G1GettingStarted/index.html>  
>  
> <http://www.programcreek.com/2013/03/hashmap-vs-treemap-vs-hashtable-vs-linkedhashmap/>  
>  
> <https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html>  
>  
> <https://en.m.wikipedia.org/wiki/Creational_pattern>  
>  
> <https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)>  
>  
> <https://lostechies.com/derickbailey/2009/02/11/solid-development-principles-in-motivational-pictures/>  
>  
> <https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>