Java Interview Topics

1. Class Loader: <http://javapapers.com/core-java/java-class-loader/> - load class on demand
   * Bootstrap Class Loader – java/lib/\*.jar - java.lang, java.ref etc…
   * Extension Class Loader – jre/lib/ext – jar
   * System Class loader – project specific class-path
2. OOPS Concept - <https://dzone.com/articles/10-oops-concepts-in-java>
   * Abstraction, Encapsulation, Inheritance, Polymorphism, Association, Composition, Aggregation, Delegation, Coupling, Cohesion
3. Collection
   * HashMap - <https://howtodoinjava.com/java/collections/hashmap/how-hashmap-works-in-java/>
4. Garbage Collection
   * Strong Reference, Weak Reference, Soft Reference, Phantom Reference
   * Throughput Garbage Collector, Concurrent Mark Sweep Collector, The Incremental low pause Collector

<http://javapapers.com/core-java/java-weak-reference/>;

<https://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html>

1. Java Heap Memory
   * Young Generation (Eden, Survivor0, Surivior1), Old Generation (Tenured), Permanent

<http://javarevisited.blogspot.com.au/2013/01/difference-between-stack-and-heap-java.html>

<https://javarevisited.blogspot.com/2011/05/java-heap-space-memory-size-jvm.html>,

<https://javarevisited.blogspot.com/2011/11/hotspot-jvm-options-java-examples.html>

1. Thread

* Wait (), sleep (), yield ()

<http://javarevisited.blogspot.com/2011/12/difference-between-wait-sleep-yield.html>

<http://stackoverflow.com/questions/9700871/what-is-difference-between-sleep-method-and-yield-method-of-multi-threading>

* Thread local

<http://veerasundar.com/blog/2010/11/java-thread-local-how-to-use-and-code-sample/>, <https://javarevisited.blogspot.com/2012/05/how-to-use-threadlocal-in-java-benefits.html>

* Blocking Queue – Linked Blocking Queue, Synchronized Blocking queue, Array bound Blocking Queue
* Executor Service – Thread Pool Executor, Fork Join Pool, Scheduled Thread Pool Executor

1. Big-O-Notation
   * Constant O(1)
   * Logarithmic O(log n) – Binary Search
   * Linear O(n)
   * Linear Logarithmic O(n log n) – Quick Sort
   * Quadratic O(n2), Cubic O(n3)
   * References - <http://www.javacodegeeks.com/2011/04/simple-big-o-notation-post.html>; <http://bigocheatsheet.com/>

<https://www.youtube.com/watch?v=V6mKVRU1evU>; <https://www.youtube.com/watch?v=6Ol2JbwoJp0&index=2&list=PLs9r0oS_gM97_1Gb9unMgAxychOZy-0zL>

1. Programming Sample ( Binary Tree) <http://www.java67.com/2016/10/how-to-print-leaf-nodes-of-binary-tree-without-recursion-in-java.html>
2. One way and mutual SSL - <https://www.ossmentor.com/2015/03/one-way-and-two-way-ssl-and-tls.html>, <http://tutorialspedia.com/an-overview-of-one-way-ssl-and-two-way-ssl/>
3. OAuth 2.0 - <https://auth0.com/docs/protocols/oauth2>, <https://aaronparecki.com/oauth-2-simplified/>, <https://developers.google.com/identity/protocols/OAuth2>

**Micro Service:**

* **Apache Kafka**: <https://www.tutorialspoint.com/apache_kafka/index.htm>

<https://kafka.apache.org/intro.html>

<https://www.slideshare.net/ConfluentInc/microservices-in-the-apache-kafka-ecosystem?next_slideshow=1>

* **Akka:** <https://akka.io/docs/>

<https://www.slideshare.net/sap1ens/akka-microservices-architecture-and-design>

* **Vertx** - http://www.sczyh30.com/vertx-blueprint-microservice/

**Design Concept –**

1. <http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns>
2. <http://javapapers.com/oops/association-aggregation-composition-abstraction-generalization-realization-dependency/>
3. http://codebetter.com/jeremymiller/2006/04/11/six-design-patterns-to-start-with/
4. Loose Coupling and High cohesion
5. Proxy Pattern - <http://java.dzone.com/articles/design-patterns-proxy>
6. Adapter Pattern -
7. Browser Cache - <https://www.mnot.net/cache_docs/>, http://www.mobify.com/blog/beginners-guide-to-http-cache-headers/

**Web-Services**

1. Understanding of UDDI and WSDL -    <http://www.ibm.com/developerworks/library/ws-wsdl/>
2. RPC & Document Style - <http://java.globinch.com/enterprise-java/web-services/soap-binding-document-rpc-style-web-services-difference/>
3. SOAP & RESTful - <http://java-success.blogspot.com/2012/02/java-web-services-interview-questions.html>; <http://javajeetechnology.weebly.com/webservices-interview-questions.html>

<http://connectedcircuits.wordpress.com/2013/02/16/what-to-use-soap-or-rest/>

<http://www.restapitutorial.com/lessons/idempotency.html>

<http://restcookbook.com/HTTP%20Methods/idempotency/>

**Isolation & Propagation**: <http://stackoverflow.com/questions/8490852/spring-transactional-isolation-propagation>

**Pessimistic locking and optimistic locking:**

1. <http://stackoverflow.com/questions/129329/optimistic-vs-pessimistic-locking>
2. <http://mjabr.wordpress.com/2011/06/10/differences-between-pessimistic-and-optimistic-locking/>

**SQL Interview**: <http://www.programmerinterview.com/index.php/database-sql/find-nth-highest-salary-sql/>, <http://www.dba-oracle.com/t_delete_duplicate_table_rows.htm>

**Hibernate**: - http://www.mkyong.com/

1. <http://stackoverflow.com/questions/5459176/difference-between-hibernate-update-by-session-update-and-hibernatetemplate-merg>
2. <http://www.developersbook.com/hibernate/interview-questions/hibernate-interview-questions-faqs.php>

**JPA**: - https://www.youtube.com/watch?v=KNdeDSUrM28

**Jenkins**: http://www.vogella.com/tutorials/Jenkins/article.html

**Java script**: <http://www.frontendjournal.com/most-common-technical-interview-question-for-frontend-developers/>

Find below sample question which I remember; if you come across new questions please do share.

1) String [Optimum Code - Big O notation]

- Program to identify whether 2 string are Anagram

- Program to generate Anagram for given string

- Program to print character count for give string (optimized code)

- Given a string consists of different types of brackets, write a function to determine the string is balanced. For example, " ([])" and "[]{}" are balanced but "([)]" and "](){" are not

- Program to generate Run length encoder for given string (Example : aabeeeesefrfffff)

- Difference between InstanceOf and getClass

- Design an algorithm and write code to remove the duplicate characters in a string without using any additional buffer.

- Implement indexOf, identify first duplicate value in given string, identify first non duplicate value in given string

2) Binary Tree / Binary Search tree

- Implement a function to check if a tree is balanced

- Write function to print Kth smallest node

- Write function to identify Lowest Common Ancestor

- Write a program to 2 binary tree are identical

- Given a tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum

3) Thread

- Explain Life cycle, dead lock, concurrent scenario

- Program on thread : complete life cycle based on scenario (multi thread)

- Program on dead lock

- Concentrate more on synchronized method/block, when to add wait() or notifyall() etc...

4) Sort

- Program on Quick Sort

- Tricky question on swap function based on usecase

- Program on Bubble sort

5) Collection

- Difference between hashmap & Hash table & Concurrent Hash map

- HashMap implementation : put(), get(); what data structure it uses

- Program on Binary Search

- Program to find intersection of 2 collection object

- Linked List (Singly/Doubly) : Delete Middle Node, Reverse linked list, delete node based on value

- Program : Stack & Queue Implementation

- Program to implement Queue using 2 stack

6) Recursive function (Question are not straight forward)

- Program on Palindrome

- Program on Fibonacci

- Program on Prime number

7) Design Pattern

- Programming on SingleTon

- Programming on Factory

- Difference between Adapter and Proxy

- Difference between Builder & X.... pattern

8) Web Service

- SOAP v/s REST

- Aware on WSDL file, how to handle service exception, retry mechanism

- REST : GET, POST, DELETE, UPDATE - Jersey Implementation using annotation

9) SQL Script

- Joins, Sub Query, Correlated query : All are based on usecase

10) Hibernate, Maven/ANT - if you mentioned in resume the 1 or 2 questions

11) Exception Handling, Serialization - need to write same program