**LEVEL 1:**

**Java Basic**- Mutability, interfaces, equals, Comparable

**Data Structure** – Map, List, set ,tree, concrete

**Exception handling**

**Garbage collection** – locks, executor, Thread, safety

**Strong Design Principles**

**Design Patterns**

**Scenario / Problem solving question (Design Pattern / Method )**

**LEVEL 2/3:**

**Design problems / Design in architectural questions**

**Open source –** spring basic

**Cache (Local / Distributor)**

**Transacting messaging**

**RDBMS**

**Webapp**

**Agile**

**TDD / BDD**

These are the some general questions :

**Core Java**

1. How HashMap works internally? explain about [hashCode()](http://codepumpkin.com/?p=855/#HashCode)  and  [equals()](http://codepumpkin.com/?p=855/#Equals) methood?  ([Answer](http://codepumpkin.com/?p=855))--done
2. hashCode() method internally calls hash(int h) function. What is the use of hash() function? What does it do?--done
3. What is rehashing How HashMap does rehashing? internally?--done
4. Write a java program to implement a custom Map implementation for MarketPlace which internally maintais key-value pairs of stocks and its 10 last updated values.

Why ConcurrentHashMap provides better performance than SynchronizedMap? --done

1. How CopyOnArrayList works?--done
2. What is the difference between LinkedHashSet and LinkedList?

**Spring**

1. Which all Spring modules have you used?
2. Explain Spring MVC life cycle.
3. Explain various bean scopes of Spring?
4. What is singleton bean scope?
5. If I am having Spring Configuration file as below :

**Design Pattern**

1. What is the difference between Spring Singleton Bean scope and Singleton design pattern.
2. How you will implement [Singleton Design Pattern](http://codepumpkin.com/?p=398)?
3. What is double check in lazy intialization singleton pattern implementation?

**SQL**

1. Database table design for What is index in database table? When do we need to create index on table?
2. Difference between clustered and non-clustered index
3. Design Bank's customers and thier account.
4. When creating extra mapping table is useful.
5. Write a SQL query to find 3rd largest salary

**Basic questions :**

1 - sort an array containing lots of 0s, 1s and 2s

2 - Implementation of Integer.parseInt("123");

3 - Implementation of Arraylist in deep detail.

4 - Diff bw Hashmap and concurrent Hashmap and detailed implementation logic --done

5 - How to make a code block ThreadSafe

6 - Reenterent lock

7 - Diff bw Synchronized block and Reenterent lock

1.Creating a stack using queues.

2.What is immutability?--done

3.What is the hashcode and equals contract in Java--done

4.What if thread in a thread pool throws an Exception.

5.Difference between abstract classes and interfaces.

6.Which design patterns do you use?

More Questions

1. Explain OOPs concepts by taking example of any collection
2. How does Arraylist works internally?
3. What is the difference between LinkedList and ArrayList? When to use LinkedList/ArrayList and Why?
4. What is Serialization and why we use?
5. In How many ways Class can be instantiated?--done
6. What is immutable and how we can achieve?--done
7. What is singleton and how we can achieve?--done
8. What is AtomicInteger and when to use?
9. What is CompareAndSwap/CompareAndGet?
10. How Garbage Collector works?
11. Which are syntactically correct
12. List l = new ArrayList<String>();
13. List<Object> l = new ArrayList<String>();
14. List<String> l = new ArrayList<Object>();
15. List list = new ArrayList<Object>();

list.add(“abc”);

1. List<String> l = new ArrayList();
2. What will be the output:

Class A{

public static void print(){

System.out.println(“A”);

}

}

Class B extends A{

public static void print(){

System.out.println(“B”);

}

public static void main(String[] args){

A a = new B();

a.print();

}

}

1. How does BlockingQueue work internally?
2. Thread Life Cycle : <https://www.baeldung.com/java-thread-lifecycle>--done
3. <https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/--done>
4. <https://stackoverflow.com/questions/18425026/shutdown-and-awaittermination-which-first-call-have-any-difference--done>
5. <https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html#awaitTermination%28long,%20java.util.concurrent.TimeUnit%29> –done
6. <https://www.geeksforgeeks.org/reentrant-lock-java/> --done
7. <https://www.youtube.com/watch?v=NUazC4EUG50>

How one can implement their own lock?

|  |  |
| --- | --- |
| **multi threading and concurrency** (hands on coding implemented  from scratch) | 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). –done Understanding of volatile keyword--done  OOPs, classes, interfaces, inner classes, Generics; Enums; access Modifier, etc  **Collections (Linked HashSet, HashMap, TreeMap etc)**  **pass by value/pass by reference**  **Multithreading (volatile, ThreadPool, Locks etc); Collection API** (Concurrent HashMap); Sync  -Can explain **difference b/w various collections** ArrayList, LinkedList, HashSet.  Knows about hashed & sorted collections. How to sort objects in java.  **-JMM or JVM** – java memory management  XML, JMS |
| Data Structures | Lists, Trees, Graphs, Queues, Stacks etc  Problem Solving, Time complexity analysis |
| **Springs** | Should have working knowledge of Dependency Injection. Good handle on spring core + basic understanding of AOP - Should have worked on two or more modules of spring, e.g. AOP, MVC. Star candidate will have the Knowledge of almost all spring modules like web-services, remoting, used spring boot |
| **ORM (Hibernate, JPA, iBatis)** | Good understanding of ORM, inheritance mapping, object life-cycle, fetching strategies & transaction management |
| **J unit** | Intermediate + knowledge of mocking frameworks. Good practices in unit testing. How to write a highly unit-test friendly code? DI, coding to interfaces |
| **Design patterns** | Knowledge of commonly used design patterns like singleton, factory, strategy, observer, command. Star Candidate - Have used singleton, strategy, observer, adaptor, template, command, factory & other patterns. Can apply appropriate DP in a given situation  **Design principles and data structures in context of some real time problem** |
| **Algorithm** | Solution to a simple problem - generally involves using map/list DS;  (array duplicity problem/ palindrome/ middle value of list, find element from list etc.) |
| SQL | Basic SQL Queries  -**SQL -** Understands various types of joins, group by & having clauses  DB Modeling, Query Tuning, Indexes, Query Plans etc  Store Procedures |
| **serialization or cloning** | Good handle on serializable, externalizable, transient variables & customizing serialization. Good handle on shallow & deep cloning. |
|  |  |
| Soft Skills | Communication Skills, Adaptable, Never say die attitude |

1. Core Java :  
   1.1 Polymorphism - overloading vs override, specifications for override, etc.   
   1.2 Inheritance - Abstract Classes, Interfaces, Casting classes, etc.   
   1.3 Generics - Upper bound, lower bound etc.   
   1.4 Multi-threading and Concurrency (must have) - wait() sleep() notify(), creating threads, Executors, volatile, Atomic package, Java Memory Model, Producer Consumer Problem, etc.   
   1.5 Collections (must have) - ArrayList, Vector, HashMap, ConcurrentHashMap, HashSet etc.

Coding Exercise Evaluation Criteria

1. Code Completeness/ Correctness
2. Code Structure and quality: Modularity, usage of OO principles, size of classes/functions,
3. Choice of data structures
4. Unit Test cases
5. Coding productivity (more time you take to submit the exercise, lesser you will score)
6. class/function/variable names, package/class structure
7. Design Patterns and Design Skills   
   2.1 Creational Design Patterns - factory method, abstract factory etc.   
   2.2 Behavioral Design Patterns   
   2.3 Hands on Skills for coding exercise
8. Database
9. Unit Tests
10. Best Practices - Agile, Test Driven Development, DevOps, Continuous Integration

**Try to answer with the help of real time examples**

1. How do we synchronize a static method to prevent data corruption in concurrent update ?
2. How will you introduce multi-tasking in your java application ?
3. Design a program to search files inside a directory using multi-threadig?
4. How will you implement a LRU timed cache in Java ?
5. There is a folder containing multiple SQL files. Each file’s name contain a sequence number which determines the execution order of that particular SQL script file. Design an API that will take such folder and execute the SQL in correct order.
6. Difference between wait() and sleep() method ?
7. What is a deadlock situation ? How will you handle deadlock in development and production environment ?
8. What is mechanism for inter-thread communication?
9. Would adding multi-threading to sorting algorithm improve its performance?
10. In what practical scenario’s multi-threading actually improves the performance of Java application ?
11. Do we need to synchronize getter and setter both to prevent concurrency related issues on a collection ?
12. Explain the key classes in java.util.concurrent package ?
13. What is purpose of ConcurrentHashMap ?
14. What is purpose of a Future ? How will you use it ?
15. How will you create your own custom Thread Pool ?
16. What is a BlockingQueue ?