1. **Java Core:**

<https://dzone.com/articles/when-to-use-abstract-class-and-intreface>

<https://www.javaworld.com/article/2139921/abstract-class-versus-interface-in-the-jdk-8-era.html>

<https://javaconceptoftheday.com/clone-method-java-lang-object-class/>

<https://www.codementor.io/savingfunda/20-java-interview-questions-and-answers-from-investment-banks-98ghl6frw>

<https://www.devfields.com/abstract-classes-and-interfaces/>

<https://javarevisited.blogspot.com/2011/06/volatile-keyword-java-example-tutorial.html>

<https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html> (Awesome Post on Hash Map)

<https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html>

<http://wordpress-264548-822440.cloudwaysapps.com/java-cyclic-barrier>

<http://wordpress-264548-822440.cloudwaysapps.com/java-countdown-latch>

<https://programming.guide/java/overriding-hashcode-and-equals.html>

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

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

<https://www.youtube.com/watch?v=2prdt2byiTs>

<https://www.geeksforgeeks.org/commonly-asked-java-programming-interview-questions-set-1/>

<https://www.javatpoint.com/corejava-interview-questions>

<http://www.corejavaguru.com/java/oop/4pillars>

<https://howtodoinjava.com/java-interview-questions/>

<https://way2java.com/collections/java-collections-interfaces-hierarchy/>

<https://dzone.com/articles/how-to-sort-objects-in-java>

<https://www.geeksforgeeks.org/java-util-hashmap-in-java/>

<https://www.geeksforgeeks.org/internal-working-of-hashmap-java/>

<https://www.geeksforgeeks.org/hashing-in-java/>

<https://www.tutorialspoint.com/data_structures_algorithms/heap_data_structure.htm>

<https://howtodoinjava.com/java/basics/jdk-jre-jvm/>

<https://dzone.com/refcardz>

<https://www.geeksforgeeks.org/marker-interface-java/>

<https://www.geeksforgeeks.org/serialization-in-java/>

<http://biercoff.com/big-o-complexity-cool-cheat-sheet/>

<https://javarevisited.blogspot.com/2017/08/top-10-java-concurrenthashmap-interview.html#axzz50U9xyqbo>

<https://www.toptal.com/java/interview-questions>

<http://infotechgems.blogspot.com/2011/11/java-collections-performance-time.html>

<https://www.journaldev.com/1321/java-string-interview-questions-and-answers>

Best Practices:

<https://howtodoinjava.com/java-best-practices/>

<https://www.callicoder.com/java-priority-queue/>

<https://howtodoinjava.com/best-practices/5-class-design-principles-solid-in-java/>

<https://jelastic.com/blog/10-object-oriented-design-principles-java-programmer-should-know-guest-post/>

Sorting Map:

<https://www.geeksforgeeks.org/sorting-hashmap-according-key-value-java/>

<https://www.geeksforgeeks.org/sorting-a-hashmap-according-to-values/>

<https://www.mkyong.com/java8/java-8-how-to-sort-a-map/>

<https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/>

*What are different types of memories in java.*

Java – 8:

<https://www.baeldung.com/java-8-interview-questions>

<https://www.baeldung.com/java-8-double-colon-operator>

<https://www.journaldev.com/8697/javase8-interview-questions-part1>

<https://www.tutorialspoint.com/java8/java8_interview_questions.htm>

<https://javahungry.blogspot.com/2018/10/java-8-interview-questions-and-answers.html>

<https://www.geeksforgeeks.org/static-vs-dynamic-binding-in-java/>

J2ee vs Spring:

<https://www.quora.com/What-are-the-differences-between-Java-EE-and-Spring>

Dead Lock:

<https://howtodoinjava.com/java/multi-threading/writing-a-deadlock-and-resolving-in-java/>

<https://dzone.com/articles/java-exception-handling-interview-questions-and-an-1>

<https://www.javamex.com/tutorials/exceptions/exceptions_hierarchy.shtml>

1. **Transaction Management:**

<https://dzone.com/articles/spring-transaction-management>

<http://www.codingpedia.org/jhadesdev/how-does-spring-transactional-really-work/>

<https://docs.spring.io/spring/docs/2.5.x/reference/transaction.html#tx-propagation>

<https://docs.spring.io/spring/docs/2.5.x/reference/transaction.html#transaction-declarative>

<https://www.tutorialspoint.com/spring/declarative_management.htm>

<https://www.byteslounge.com/tutorials/spring-transaction-propagation-tutorial>

<https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/transaction.html>

<https://dzone.com/articles/function-interface-functional>

<http://tutorials.jenkov.com/java-functional-programming/functional-interfaces.html>

<https://www.logicbig.com/tutorials/spring-framework/spring-data-access-with-jdbc/correct-use-of-declarative-transaction.html>

<https://codete.com/blog/5-common-spring-transactional-pitfalls/>

**Multi Transactional:**

<https://stackoverflow.com/questions/48954763/spring-transactional-with-a-transaction-across-multiple-data-sources>

<https://docs.spring.io/spring-data/commons/docs/1.6.2.RELEASE/api/org/springframework/data/transaction/ChainedTransactionManager.html>

<https://www.baeldung.com/spring-data-jpa-multiple-databases>

1. **Thread And Async:**

<https://dzone.com/articles/spring-and-threads-taskexecutor>

<https://dzone.com/articles/java-callable-future-understanding>

<https://dzone.com/articles/async-await-in-java>

<https://dzone.com/articles/spring-boot-creating-asynchronous-methods-using-as>

<https://www.deadcoderising.com/java8-writing-asynchronous-code-with-completablefuture/>

<https://academy.realm.io/posts/360andev-stacy-devino-async-tasks-threads-pools-executors-android/>

<https://howtodoinjava.com/java/multi-threading/how-to-use-blockingqueue-and-threadpoolexecutor-in-java/>

1. **HiBernate:**

<https://www.javatpoint.com/hibernate-and-spring-integration>

<https://www.baeldung.com/the-persistence-layer-with-spring-and-jpa>

<https://dzone.com/articles/springhibernate-application>

<https://www.journaldev.com/3524/spring-hibernate-integration-example-tutorial>

<https://www.baeldung.com/hibernate-5-spring>

<https://github.com/eugenp/tutorials/tree/master/persistence-modules/spring-hibernate-5/src/main/java/com/baeldung/spring>

<https://www.javarticles.com/2015/04/spring-hibernatetempate-example.html>

<https://o7planning.org/en/11665/spring-boot-hibernate-and-spring-transaction-tutorial>

<https://dzone.com/articles/all-hibernate-annotations-mapping-annotations>

<https://www.journaldev.com/17803/jpa-hibernate-annotations>

<https://www.baeldung.com/hibernate-lazy-eager-loading>

<https://dzone.com/articles/all-jpa-annotations-mapping-annotations>

<https://www.educba.com/hibernate-interview-questions/>

<https://www.mkyong.com/hibernate/hibernate-many-to-many-relationship-example-annotation/>

<https://www.baeldung.com/stored-procedures-with-hibernate-tutorial>

Hibernate Cache:

<https://javarevisited.blogspot.com/2017/03/difference-between-first-and-second-level-cache-in-Hibernate.html>

<https://www.baeldung.com/hibernate-second-level-cache>

<https://docs.jboss.org/hibernate/orm/4.0/devguide/en-US/html/ch06.html>

<https://howtodoinjava.com/hibernate/understanding-hibernate-first-level-cache-with-example/>

<https://github.com/eugenp/tutorials/tree/master/persistence-modules/spring-persistence-simple/src/main/java/com/baeldung/hibernate/bootstrap>

Paging:

<https://howtodoinjava.com/spring-boot2/pagination-sorting-example/>

<https://riptutorial.com/spring-data/example/28382/pagination-by-passing-parmeter-with-custom-query-in-spring-data-jpa>

<https://www.baeldung.com/spring-data-jpa-query>

<https://www.mkyong.com/hibernate/hibernate-criteria-examples/>

<https://www.baeldung.com/hibernate-pagination>

1. **JMS:**

<https://career.guru99.com/top-25-jms-java-message-service-interview-questions/>

<http://www.code2succeed.com/pub-sub/>

<https://www.jesse-anderson.com/2016/07/apache-kafka-and-google-cloud-pubsub/>

1. **Spring:**

<https://dzone.com/articles/understanding-the-basics-of-spring-vs-spring-boot>

<https://www.baeldung.com/spring-vs-spring-boot>

<https://www.springboottutorial.com/spring-boot-auto-configuration>

<https://www.journaldev.com/2888/spring-tutorial-spring-core-tutorial>

<https://www.tutorialspoint.com/spring/spring_bean_definition.htm>

<https://javabeginnerstutorial.com/spring-framework-tutorial/java-spring-bean-lifecycle/>

<https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring>

<https://dzone.com/refcardz/spring-annotations?chapter=2>

<https://www.baeldung.com/spring-autowire>

<https://howtodoinjava.com/spring-core/spring-ioc-vs-di/>

<https://stackoverflow.com/questions/6827752/whats-the-difference-between-component-repository-service-annotations-in>

<https://javarevisited.blogspot.com/2012/11/difference-between-setter-injection-vs-constructor-injection-spring-framework.html>

<https://blog.eduonix.com/java-programming-2/learn-design-patterns-used-spring-framework/>

<https://premaseem.wordpress.com/2013/02/09/spring-design-patterns-used-in-java-spring-framework/>

Bean Life Cycle

<https://howtodoinjava.com/spring-core/spring-bean-life-cycle/>

<https://www.concretepage.com/spring/spring-bean-life-cycle-tutorial>

<https://medium.com/teamarimac/implementing-throttling-in-java-spring-boot-ec4723cfce9f>

https://www.youtube.com/watch?v=RcmEZfgHJ\_0

1. Spring Boot Intercept:

<https://www.tutorialspoint.com/spring_boot/spring_boot_interceptor.htm>

<https://www.baeldung.com/running-setup-logic-on-startup-in-spring>

<https://o7planning.org/en/11689/spring-boot-interceptors-tutorial>

<https://www.springboottutorial.com/spring-boot-tutorials-for-beginners>

<https://stackoverflow.com/questions/35856454/difference-between-interceptor-and-filter-in-spring-mvc>

1. Spring MVC Dispatcher servlet

<https://dzone.com/articles/how-spring-mvc-really-works>

<https://stackoverflow.com/questions/35121399/how-to-configure-dispatcherservlet-in-a-spring-boot-application>

<https://howtodoinjava.com/spring5/webmvc/spring-dispatcherservlet-tutorial/>

Spring Servlet Filter:

<https://www.baeldung.com/spring-boot-add-filter>

<https://www.javadevjournal.com/spring-boot/spring-boot-add-filter/>

<https://www.baeldung.com/intercepting-filter-pattern-in-java>

<https://www.tutorialspoint.com/spring_boot/spring_boot_servlet_filter.htm>

<https://stackoverflow.com/questions/27743840/restrict-tomcat-in-spring-boot-to-certain-ip-addresses>

<https://www.logicbig.com/tutorials/spring-framework/spring-data-access-with-jdbc/data-access-exception.html>

<https://github.com/topics/cracking-the-coding-interview?o=desc&s>

<https://www.byte-by-byte.com/six-software-engineering-interview-questions/>

Repo: <https://github.com/eugenp/tutorials/tree/master/spring-boot>

1. Memory Leakes: https://www.toptal.com/java/hunting-memory-leaks-in-java

<https://stackify.com/java-memory-leaks-solutions/>

OuOfMemory, GC Overhead, StackOverflow, Memory Leaks

https://www.baeldung.com/java-gc-overhead-limit-exceeded

1. Rest:

**Explain how caching in a REST API takes advantage of the layered nature of the internet ?**

<https://www.pyarb.com/developer/java/principles-rest-architecture.html>

<https://www.javagists.com/rest-api-interview-questions>

<https://dzone.com/articles/top-rest-api-best-practices>

Exception in Service:

<https://www.baeldung.com/exception-handling-for-rest-with-spring>

<https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc>

Map<String, Object> params = new HashMap<>();

params.put("account", accountNumber);

return restTemplate.getForObject("/mvc/customer/getState/{account}/more/url", String.class, params);

UriComponentsBuilder builder = UriComponentsBuilder.*fromHttpUrl*(**retrieveMembershipsUrl**);

builder = builder.path(svocId);

**if**(StringUtils.*isNotBlank*(expand)) {

builder = builder.queryParam(**"expand"**, expand);

}

builder.toUriString()

1. Design Pattern:

<https://www.callicoder.com/java-singleton-design-pattern-example/>

1. Oracle Query

<https://artoftesting.com/interviewSection/sql-queries-for-interview.html>

1. SOAP Service

<https://www.baeldung.com/spring-boot-soap-web-service>

1. Micro services

<https://medium.com/clover-platform-blog/building-a-microservice-with-spring-boot-and-spring-cloud-1c8275d7d229>

<https://www.edureka.co/blog/microservices-with-spring-boot>

1. Load Balancing

<http://appsdeveloperblog.com/zuul-and-eureka-load-balancing-example/>

<https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/>

**Sum of Subarray Minimums**

**Sort Array By Parity**

[Partition Array into Disjoint Intervals](https://leetcode.com/problems/partition-array-into-disjoint-intervals)

*Find all top-left corners of squares in binary 2D matrix.(*xcept for the last row and the last column, every single cell will be a top left corner of a square*)*

Given a string, print all possible palindromic partitions

[*https://www.geeksforgeeks.org/recursion-practice-problems-solutions/*](https://www.geeksforgeeks.org/recursion-practice-problems-solutions/)

[*https://www.geeksforgeeks.org/palindrome-partitioning-dp-17/*](https://www.geeksforgeeks.org/palindrome-partitioning-dp-17/)

[*https://www.teamblind.com/article/Karat-interview-CgHsUTaU*](https://www.teamblind.com/article/Karat-interview-CgHsUTaU)

OAuth 2;

http://websystique.com/spring-security/secure-spring-rest-api-using-oauth2/

<https://howtodoinjava.com/spring-security-tutorial/>

<https://dzone.com/articles/securing-rest-services-with-oauth2-in-springboot-1>

Memory Leakes:

https://www.toptal.com/java/hunting-memory-leaks-in-java

https://stackify.com/java-memory-leaks-solutions/

Data Structures:

https://www.tutorialspoint.com/data\_structures\_algorithms/heap\_data\_structure.htm

**Behavior Questions:**

https://zety.com/blog/star-method-interview

Goldman Sachs:

MinHeap, Max Heap, Decorator Design Pattern

Different types of Index in Sql

Root class of Java, Main Methods of Object Class, what is NotifyAll() Method in java, Concurrent hashMap(0 in java

Find length of the shortest subarray greater than target

Find the smallestSubArraySum And return Sum

Java Script:

https://www.toptal.com/javascript/interview-questions

https://www.codementor.io/nihantanu/21-essential-javascript-tech-interview-practice-questions-answers-du107p62z

http://es6-features.org/#Constants

https://codetower.github.io/es6-features/

Java:

 List the collection framework interfaces and implementations.

Define a property of each. Difference between Set and HashMap?

 How can you get a synchronized LinkedList? Does this make the program thread safe?

 Compare - Arraylist and LinkedList. And when do you use each of these?

 How would you create a Hashmap data structure? How to handle collisions in a hashmap?

 What methods you need to override when using object as key in hashmap? Why do you need to override these?

 Concurrent HashMap.

 Different ways of creating threads in java.

Which one is the best. Difference between String and StringBuilder? What does immutable mean and how does String class become immutable?

 Algo:

 Given pointer to two nodes in a binary tree, find their common parent. Give its time complexity

 Write a sorting algorithm and explain the working with an example for 6 elements. Compare performance with other sorting algos.

 Others:

 What data structures are used in database indexing. How they are used for a date range query?

 Unit testing and why it is required. Why do a unit test on each module when doing a maven/ant build?

 Use of subversion – merging, updates, etc Performance monitoring tools.

 HR / Behavioral Why do you want to leave your present job? Why financial industry? What will you do if after starting you feel the financial industry is not of your interest? Why Goldman? Difference between what Goldman does and what other companies like Bank of America or Chase do? Did you ever work in a team which did not co operate? How did you handle this situation? Do you have experience working with global teams? Questions about role and responsibilities at present job.

JPA implementation.

Write a function to checkoff the input number is a power of 10

How did you communicate with other systems Sync And Async

1. How transaction Management happen in Spring boot and what will happen when exception happen
2. Benefits and draw backs of using new String and as literal
3. Topic can listen to multiple subscribers
4. What will happen if object is not override HashCode but equals
5. Why we need to use spring boot if we have Spring
6. How to send sensitive data through restService
7. About Garbage collector how it will work how to invoke them manually

Charles Schwab:

1. Inheritance, oop fundamentals, cohesion, association, get/put http protocols, a simple coding puzzle.
2. Why should we hire YOU?
3. SQL questions such as SQL injection Explain OOP
4. What is object oriented programming, what are the main principles of OOP, what is mvc, what is a linked list, what is a garbage dump.
5. How well do i work in a group environment
6. *Most questions were related cloud development. 12 factor  apps, devops, spring boot, Lot of spring annotation question. AI algorithms*
7. Types of join with a simple example, normalization, indexes
8. Java based questions(basics, web services, optimizations etc)
9. Validate a password given some restrictions
10. Given a number between 0 and 999999999, print out human readable text of the number. Ex 100 = One Hundred, 5225 = Five Thousand Two Hundred Twenty Five

https://stackoverflow.com/questions/3911966/how-to-convert-number-to-words-in-java

1. MergeSort, O(N) time complexity, Enterprise Library, REST APIs
2. The coding question was presidents.json file. This file was already given to me by Xoriant to ready for Chrles Schwab interview(don't know that is legal or not). You should create a restfull API which read from this file and convert json to object in java application and then show them all presidents to user in web browser by calling the end-point. Later they ask you to create additional end-point to show all alive presidents.
3. How would you like to leave your mark here, in five years time?
4. Please explain one occurrence of conflict you had and how you overcame it
5. Name a few design patterns. What are some antipatterns? What is big oh of various operations (looking up value in hash table, looking up in a sorted/unsorted list)?
6. Tell me about a hard decision you had to make
7. Why should we hire YOU?
8. SQL questions such as SQL injection Explain OOP
9. Describe a time that you went above and beyond on a project
10. What is a parallel query, when would you use it and under what conditions would it not be a good choice?
11. Cutting edge technologies that I had used.
12. what the difference b/w traditional application and enterprise application
13. what is difference between pointers and delegates what is dependency injection Question on left outer join unit testing examples
14. Write a program that takes a string and prints it out backwards.
15. fizzbuzz programming question
16. What do you know about web apis
17. What are the OOP principles? Why Schwab? Talk about projects that are on your resume. Know about data structures and why you should use one over another.
18. Implement a sorting algorithm, SQL joins, Stack vs LinkedList, Buffers vs StringBuffers, Multi-level inheritance, ArrayList vs Vectors, Garbage collection in Java, abstract classes vs interfaces, what is thread-safe , how can you implement thread-safe mechanism, synchronized vs thread-safe etc
19. Describe a time you encountered someone who had a differing view than you, how did you handle it? (Asked during phone interview.)
20. Say you encounter someone who disagrees with you in the work place and it's impacting your and their work, how do you solve this problem?
21. What is the difference between a strong leader and a strong manager?
22. What is a problem that you solved in the past year that you are proud of?

CTS

1. How we build a fault tolerant and graceful Services/ Application using spring boot
2. Is there any difference between @Service and @Repository
3. How to use Filter in Spring boot application
4. How to intercept http request before it reach controller
5. How to handle 5xx in generic way by throwing customers response
6. Rest services principles
7. Best practices of Rest Services
8. Best Practices to use Micro services
9. React flow
10. Bean Life cycle
11. How auto wiring works
12. Consumer Interface

How to Intercept a Rest Response/Request before reach to controller:

1st way:

@Configuration  
**public class** InterceptorConfig **implements** WebMvcConfigurer { (or)

**public class** InterceptorConfig **extends** WebMvcConfigurerAdapter {   
   
 @Autowired  
 **private** RequestLoggerInterceptor **requestLoggerInterceptor**;  
   
 @Override  
 **public void** addInterceptors(InterceptorRegistry registry) {  
 registry.addInterceptor(**requestLoggerInterceptor**);  
 }  
}

**public class** RequestLoggerInterceptor **extends** HandlerInterceptorAdapter { (OR)

**public class** RequestLoggerInterceptor **implements** HandlerInterceptor {  
   
  
@Override

public boolean preHandle(

HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

return true;

}

@Override

public void postHandle(

HttpServletRequest request, HttpServletResponse response, Object handler,

ModelAndView modelAndView) throws Exception {}

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response,

Object handler, Exception exception) throws Exception {}

@Override

**public void** afterConcurrentHandlingStarted(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {  
}

}

**What you have:**

We are looking for a passionate engineer who cares about writing clean, testable code. We expect you to have:

* 5+ years software development experience.
* Deep knowledge of Java and Spring ecosystem
* Experience building RESTful webservices or apps which consume RESTful interfaces.
* Experience with BDD, TDD.
* The ability to work across teams and organizations to get things done.
* Passion to learn new technologies and best practices for software development.
* Bachelor of Science in Computer Science or related fields.
* Experience with a PaaS like Cloud Foundry preferred.
* Experience with cloud frameworks & design patterns such as Microservices, 12-factor design, OOD et preferred
* Front end experience with a modern Javascript framework (Angular etc) preferred.
* Experience building applications with MongoDB, Aerospike or Redis as the storage mechanism preferred.
* Experience developing and supporting apps in a high visibility production environment preferred.

Spanner Template:

ProgramRepositoryImpl:

@Autowired  
**private** LoyaltyDAO<BaseEntity> **loyaltyDAO**;

**loyaltyDAO**.saveOrUpdate(programInclusionExclusionEntityList);

**loyaltyDAO**.query(ProgramStoreFeaturesEntity.**class**, LoyaltyQueries.***PROGRAM\_FIND\_ALL\_PROGRAMS\_IN\_STORE\_BY\_STORE\_NUMBER***, tags, params, options);

Get Query:

**public** <S **extends** T> List<S> query(Class<S> entityClass , String sql , @Nullable List<String> tags , @Nullable Object[] params , @Nullable SpannerQueryOptions options) {  
   
 Preconditions.*checkNotNull*(sql, **"Sql query is required to execute"**);  
   
 String finalSql = sql;  
   
 Statement buildStatementFromSqlWithArgs = SpannerStatementQueryExecutor.*buildStatementFromSqlWithArgs*(finalSql, tags, **null**, params);  
  
 **return this**.query(entityClass, buildStatementFromSqlWithArgs, options);  
  
}  
  
@Override  
**public** <S> List<S> query(Class<S> entityClass , Statement statement , SpannerQueryOptions options) {  
 **long** startTs = System.*currentTimeMillis*();  
 List<S> result = **super**.query(entityClass, statement, options);  
 **long** endTs = System.*currentTimeMillis*();  
 logPerformanceStats(statement.getSql(), endTs - startTs);  
 **return** result;  
}

Loyalty DAO:

**public class** LoyaltyDAO<T **extends** BaseEntity> **extends** LoyaltyBaseDAO<T> {

**public** LoyaltyDAO(DatabaseClient databaseClient , SpannerMappingContext mappingContext , SpannerEntityProcessor spannerEntityProcessor , SpannerMutationFactory spannerMutationFactory ,  
 SpannerSchemaUtils spannerSchemaUtils) {  
 **super**(databaseClient, mappingContext, spannerEntityProcessor, spannerMutationFactory, spannerSchemaUtils);  
}

LoyaltyDAOProvider

:

**public** LoyaltyDAOProvider(DatabaseId databaseid , SpannerMappingContext spannerMappingCnxt , SpannerEntityProcessor enityProcessor , **int** minSessions , **int** maxSessions) {  
 **try** {  
 SessionPoolOptions sessionPoolOptions = SessionPoolOptions.*newBuilder*().setMinSessions(minSessions).setMaxSessions(maxSessions).build();  
 *spannerDbClient* = SpannerOptions.*newBuilder*().setSessionPoolOption(sessionPoolOptions).build().getService().getDatabaseClient(databaseid);  
 *spannerMappingContext* = spannerMappingCnxt;  
 *spannerEntityProcessor* = enityProcessor;  
 *spannerSchemaUtils* = **new** SpannerSchemaUtils(*spannerMappingContext*, *spannerEntityProcessor*, **true**);  
 *spannerMutationFactory* = **new** SpannerMutationFactoryImpl(*spannerEntityProcessor*, *spannerMappingContext*, *spannerSchemaUtils*);  
   
 } **catch** (Exception ex) {  
 **throw new** LoyaltyPersistentException(**"Unable to create the SpannerTemplate object"**, ex);  
 }  
   
}  
  
@SuppressWarnings(**"rawtypes"**)  
**public** LoyaltyDAO<?> getSpannerTemplate() {  
   
 **return new** LoyaltyDAO(*spannerDbClient*, *spannerMappingContext*, *spannerEntityProcessor*, *spannerMutationFactory*, *spannerSchemaUtils*);  
}  
  
@SuppressWarnings(**"rawtypes"**)  
**public** LoyaltySequenceDAO<?> getSequenceSpannerTemplate() {  
 **return new** LoyaltySequenceDAO(*spannerDbClient*, *spannerMappingContext*, *spannerEntityProcessor*, *spannerMutationFactory*, *spannerSchemaUtils*);  
}

LoyaltyDAOConfig:

@Data  
@Configuration  
**public class** LoyaltyDAOConfig {  
   
 @Value(value = **"${datasource.spanner.instanceid}"**)  
 **private** String **instanceId**;  
 @Value(value = **"${datasource.spanner.databasename}"**)  
 **private** String **dataBaseName**;  
   
 @Value(**"${datasource.spanner.sessions.min}"**)  
 **private int minSessions**;  
   
 @Value(**"${datasource.spanner.sessions.max}"**)  
 **private int maxSessions**;  
   
 */\*\*  
 \* This method will return the DatabaseId required to create the database client  
 \* object to interact with spanner database.  
 \*   
 \** ***@return*** *\*/* **public** DatabaseId getSpannerDataBaseId() {  
 **return** DatabaseId.*of*(getProjectId(), **instanceId**, **dataBaseName**);  
 }  
   
 **public** String getProjectId() {  
 **return new** DefaultGcpProjectIdProvider().getProjectId();  
 }  
   
 @Bean  
 @Autowired  
 **public** LoyaltyDAOProvider getloyaltyDAOProvider() {  
 **return new** LoyaltyDAOProvider(getSpannerDataBaseId(), getSpannerMappingContext(), getSpannerEntityProcessor(getSpannerMappingContext()), **minSessions**, **maxSessions**);  
 }  
   
 @Bean  
 @Autowired  
 **public** LoyaltySequenceDAO<?> getLoyaltySequenceDAO(LoyaltyDAOProvider loyaltyDAOProvider) {  
 **return** loyaltyDAOProvider.getSequenceSpannerTemplate();  
 }  
   
 @Bean  
 @Autowired  
 **public** LoyaltyDAO<?> getLoyaltyDAO(LoyaltyDAOProvider loyaltyDAOProvider) {  
 **return** loyaltyDAOProvider.getSpannerTemplate();  
 }  
   
 @Bean  
 @Autowired  
 **public** SpannerMappingContext getSpannerMappingContext() {  
 **return new** SpannerMappingContext();  
 }  
   
 @Bean  
   
 **public** SpannerEntityProcessor getSpannerEntityProcessor(SpannerMappingContext mappingContext) {  
 **return new** ConverterAwareMappingSpannerEntityProcessor(mappingContext);  
 }  
   
 @Bean  
 @Autowired  
 **public** SpannerSchemaUtils getSpannerSchemaUtils(SpannerMappingContext mappingContext , SpannerEntityProcessor spannerEntityProcessor) {  
 **return new** SpannerSchemaUtils(mappingContext, spannerEntityProcessor, **true**);  
 }  
   
}

Behavior Questions:

* 1. Why we Hire you? (Or) Why you stand out among Others? (Or) Why we hire you not others

Answer: First And fore most I have knowledge and Work experience and qualification that required for the job and as mentioned in Job description. I have 7+ years of software development experience where I learnt best practices in coding, debugging skills and better way of solving issues these are really add a value to you company. Having cloud exposure. Along with all these qualification I have qualities most of the developers have like Hard work and Criticsl thinking and all.

* 1. Why this company?

Answer:

* 1. Why this Domain?

Answer:

* 1. Tell me situation where under lots of pressure?

Answer:

* 1. Tell me the work you proud?

Answer:

* 1. If you feel you are not good with this domain what you will do?

Answer:

* 1. How well you do in group Environment?

Answer:

* 1. How would you like to leave your mark in next 5 years?

Answer:

* 1. What asset you bring to company

Answer:

* 1. Please explain one occurrence of conflict you had and how you overcame it?

Answer:

* 1. Tell me about a hard decision you had to make?

Answer:

* 1. Describe a time that you went above and beyond on a project

Answer:

CDK:

1. Design an algorithm to generate meaningful words for phone numbers?
2. Design a API which takes JSON and filter result based on criteria

<https://www.javadevjournal.com/spring-boot/spring-boot-application-intellij/>

<https://spring.io/guides/gs/rest-service/>

Rect New Hooks:

<https://medium.freecodecamp.org/learn-the-basics-of-react-hooks-in-10-minutes-b2898287fe5d>

<https://blog.bitsrc.io/9-web-component-ui-libraries-you-should-know-in-2019-9d4476c3f103>

<https://medium.com/javascript-in-plain-english/state-management-with-react-hooks-no-redux-or-context-api-8b3035ceecf8>

Design Pattern:

<https://www.tutorialspoint.com/design_pattern/facade_pattern.htm>

<https://dzone.com/articles/creational-design-patterns-builder-pattern>

<http://techytuts.in/builder-design-pattern/>

<https://egkatzioura.com/2018/03/27/creational-design-patterns-factory-pattern/>

<https://www.baeldung.com/java-dao-pattern>

<https://www.javatpoint.com/factory-method-design-pattern>

Session facade, Business Delegates, Singleton, Factory, Service locator, Value Object and DAO pattern.

SOA VS Micro services

<https://medium.com/@kikchee/microservices-vs-soa-is-there-any-difference-at-all-2a1e3b66e1be>

Threads Java:

<https://winterbe.com/posts/2015/04/07/java8-concurrency-tutorial-thread-executor-examples/>

<https://winterbe.com/posts/2015/04/30/java8-concurrency-tutorial-synchronized-locks-examples/>

<https://winterbe.com/posts/2015/05/22/java8-concurrency-tutorial-atomic-concurrent-map-examples/>

Deadlock in java is a programming situation where two or more threads are blocked forever. Java deadlock situation arises with at least two threads and two or more resources.

**How to Detect Deadlock in Java**

To detect a deadlock in Java, we need to look at the *java thread dump* of the application, we can generate thread dump using *VisualVM* profiler or using *jstack* utility.

For analyzing deadlock, we need to look out for the threads with the state as *BLOCKED* and then the resources it’s waiting to lock. Every resource has a unique ID using which we can find which thread is already holding the lock on the object.

**How to avoid deadlock in java**

These are some of the guidelines using which we can avoid most of the deadlock situations.

* **Avoid deadlock by breaking circular wait condition**: In order to do that, you can make arrangement in the code to impose the ordering on acquisition and release of locks. If lock will be acquired in a consistent order and released in just opposite order, there would not be a situation where one thread is holding a lock which is acquired by other and vice-versa.
* **Avoid Nested Locks:** This is the most common reason for deadlocks, avoid locking another resource if you already hold one. It’s almost impossible to get a deadlock situation if you are working with only one object lock.
* **Lock Only What is Required:** You should acquire lock only on the resources you have to work on, if we are only interested in one of its fields, then we should lock only that *specific field not complete object*.
* **Avoid waiting indefinitely:** You can get a deadlock if two threads are waiting for each other to finish indefinitely using thread join. If your thread has to wait for another thread to finish, it’s always *best to use join with maximum time you want to wait for the thread to finish*.
* Use Synchronized collections

**Code Review:**

<http://javadecodedquestions.blogspot.com/2017/06/java-code-review.html>

Spring WebFlux

<https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html>

<https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#spring-web>

<https://dzone.com/articles/spring-webflux-getting-started>