Topics

***Core Java***

1. Define Association and its types i.e. Aggregation and Composition.
   1. With two given classes A & B, how will you implement aggregation and composition programmatically?
2. What do you mean by Immutable classes? Examples from JDK APIs.
   1. Why String is immutable?
   2. Unmodifiable vs Immutable collections.
   3. How can you make immutable classes?
   4. Cross questions on above topic, if non-immutable objects under immutable class and if that non-immutable class further contains non-immutable objects inside it.

🡪 What will be your approach to make the classes immutable in such cases?

1. Can you override instance or static variables? If yes/no, what is the concrete reason behind that?
2. Be prepared with different questions on function overloading. E.g.
   1. Can you overload methods with different return types?
   2. Can you overload with changing the argument types from primitive to wrapper type i.e

Void abc (int a) {}; Void abc (Integer a) {};

* 1. Be prepared with method overloading with Autoboxing and Widening

1. Serliazable vs Externalizable interface with possible cross questions.
2. What are Checked & non-checked exceptions? How you will create runtime exceptions? Explanation with real time scenarios to support the use of both types of exceptions.
3. What is JDK, JRE & JVM with internals of JVM? What are class loaders in java and how can you create your custom class loaders? Troubleshooting approaches for ClassNotFound exception and NoClassDefFoundError.
4. When to use volatile keyword and what is its use?
   1. Does declaring a variable as volatile ensures thread safety? If not then what to use?
5. When to use atomic variables? Which algorithm is used to identify the change? (CAS- Compare & Swap)
6. What is marker interface? Can you declare your own marker interface?
7. How to test a void method?
8. Can an abstract class have a constructor? When is that called?
9. What is difference between init and static block? What is the order of execution among init/static/constructor?

***JDK Memory Model***

1. How is memory managed in JAVA?
2. What is young generation and old generation?
   1. Eden/Survivor Spaces and how the transition happens between the memory spaces.
3. Minor and Major GC?
4. Various Garbage collection types and how to specify which garbage collection should be used?
5. CMS & G1 garbage collection and what does String Deduplication means?
6. How to take heap dump and analyse the same?
7. Concept of String Pool can be touched in order to judge the memory allocation for a String Object.

***Serialization/Externalization***

1. What does serialization means and why is it required?
   1. How can you achieve serialization?
   2. What are compatible and non-compatible changes for a serialized object?
2. What impact does Final, Transient & Static keywords have on member variables during serialization/deserialization?
3. What will happen if one the member of class does not implement Serializable interface?
4. What is the role of serialVersionUID?
5. What will happen if have a collection member variable in a class to be serialized?
6. Does constructor of class gets called during deserialization?
7. How you can avoid Deserialization process creating another instance of Singleton class?
   1. readResolve()

***Collections***

1. How HashSet and TreeSet internally works and what is the complexity to do operations like get, put on these collections?
   1. Will TreeSet allows null objects storage? If yes/no then why?
   2. How TreeSet able to maintain sorting of objects.
2. Scenarios where you want to use ArrayList and LinkedList with reasons.
3. How Collections.sort(list of objects) works internally, what will be the complexity to sort elements using above approach and how?
4. Internal working of HashMap (Any differences in internal implementation w.r.t JDK 1.7 and JDK 1.8), Concurrent HashMap & TreeMap. How LinkedHashMap able to maintain insertion order of elements.
5. Internals of Locking strategy used by Concurrent HashMap to provide synchronized behaviour in concurrent Environment.
6. Understanding of Concurrent APIs like CopyOnWriteArrayList, ConcurrentHashMap etc.
7. How can you sort hashmap on the basis of values with the help of JDK 1.8 Stream APIs and without that?
8. Contract between object’s class equals () & hashcode (). How these methods going to be used inside hashing technique. If you are using any object (like Employee class object) as a custom key inside the HashMap, how you will override these methods?
9. What type of classes should be used as keys for hashmap()?
10. Further questions around this like overriding hashCode() with constant or returning always true/false from equals() method?
11. What is dirty read in a Hashmap?
12. How does rehashing work in a Hashmap?
13. What is fail fast and failsafe?
14. Which collection implementation is failfast and which all are failsafe? (Concurrent modification exception)
15. Which all iterators are available as part of collection API?
16. What add-on feature does list Iterator provides in comparison to other iterator?
17. What are IdentityHashMap and WeakHashMap?

***JDK 1.8 Specific questions***

1. What is a stream? How does it differ from a collection?
2. What is the difference between intermediate, terminal & short-circuit operations?
3. What is the difference between map and flatMap stream operation?
4. What is stream pipelining in Java 8?
5. Program to check even/odd and prime numbers using stream in jdk 1.8
6. What is a functional interface? What are the rules of defining a functional interface?
7. Define default functional interfaces like: Function, Consumer, Supplier , Predicate, BiFunction, BinaryOperator, UnaryOperator
8. What is a lambda expression? What are its advantages? Where do we use a lambda expression?
9. What is a method reference with different types?
10. With interfaces having default methods, how JDK 1.8 able to sort out diamond problem?

***Multithreading & Java Concurrency***

1. What kind of common problems (which usually comes while doing concurrent operations) you have faced in multi-threading environment? How did you resolve it?
2. Mention some guidelines or best practices you have used while writing thread safe code.
3. How do you handle an unhandled exception in the thread?
4. What thread-scheduling algorithm is used in Java?
5. You have thread T1, T2, and T3, how will you ensure that thread T2 run after T1 and thread T3 run after T2?
6. Apart from Thread class instance join (), what are the other ways to do that? How join method is able to achieve it internally?
7. If you have to implement a high-performance cache which allows multiple readers but the single writer to keep the integrity how will you implement it?
8. Thread life cycle with difference between wait, sleep and yield methods.
9. Describe the purpose and use-cases of the fork/join framework.
10. How to generate and analyse Thread Dumps?
11. Difference between object lock and class lock?
12. What will be your design approach, if you have to design your own custom thread pool?
13. What will be your approach to handle uncaught runtime exception generated in run method?
14. What is CountDownLatch & CyclicBarrier? If you have to implement it by your own, what will be your approach?
15. Difference between synchronized and ReentrantLock in java?
16. What is executor framework in java? Explain the usage of Executor, ExecutorService inside that. Explain thread pool configuration in detail like CorePoolSize, MaximumPoolSize and KeepAliveTime.
17. What are the available implementations of ExecutorService in the standard library? What are the benefits of using ThreadPoolExecutor implementation of ExecutorService interface?
18. How static keyword does impacts the thread locks?
19. What is deadlock, livelock & thread starvation?
20. What are Futures object?
21. Completable futures?
22. What is thread local and how to implement it?
23. For which particular use case one should implement a thread local?
24. Blocking queue in JAVA and how that can be implemented?

***Spring***

1. What is Aspect, Advice, Pointcut, JointPoint and Advice Arguments in AOP?
2. What are the different types of Advices?
3. Explain the way you are doing transaction management and error handling in your spring applications.
4. What is the Bean life cycle in Spring Bean Factory Container and what are the callback methods in Spring?
5. What is Spring IoC Container? What is the difference between BeanFactory and ApplicationContext?
6. What is the default scope of beans in Spring? Explain all the scopes available in spring.
7. Difference between singleton scope bean and singleton class?
8. What is dependency injection and the types? When to use Setter and when to use Constructor dependency injection.
9. What do you understand by auto wiring and name the different modes of it?
10. What’s the difference between @Component, @Controller, @Repository & @Service annotations in Spring?
11. Use of @Required, @Autowired, @Resource & @Qualifier annotations
12. How to stop loading some beans in application context at start up?
13. How to resolve circular or cyclic dependency related issues like BeanCurrentlyInCreationException ?
14. Why it’s better to avoid constructor injection if there is any cyclic dependency?
15. How to Inject Prototype Scoped Bean in Singleton Bean so that the injected bean should behave like Prototype instead of outer Singleton bean.
16. Usage of ApplicationContextAware?

***Hibernate***

1. [What are the advantages of Hibernate over JDBC?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-vs-jdbc)
2. Is [SessionFactory and Session thread safe?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#session-factory-thread-safe)
3. How many SessionFactory & Sessions can we have for a connection?
4. [What is hibernate caching? Explain Hibernate first level cache?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-caching) How to configure second level caching? What is query level cache?
5. Difference between get() & load(), save() & persist(), merge() & update()
6. Different entity states and which operation can be performed in which state?

(Transient/Persistent/Detached)

1. What is HQL, Named SQL query and criteria query? How and when to use which one?
2. [What is cascading and what are different types of cascading?](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers#hibernate-cascading)
3. What is lazy & eager loading in hibernate? What is N+1 SELECT problem in Hibernate?
4. Explain different fetching strategies & Inheritance Mapping Strategies.
5. Which annotation can be used to avoid a field from an entity to be persisted in DB? (@Transient)
6. Have you implemented any connection pooling in your application? If yes, which connection pool you have used? Benefits and drawbacks of using connection pooling?
7. How to resolve LazyInitializationException & OptimisticLockException in Hibernate?
8. What is optimistic and pessimistic locking and which one should be used on scenario basis?
9. What is flush() method and when to use it?
10. How can you define relationships in different entities?
11. OneToOne/OneToMany/ManyToOne/ManyToMany?
12. What is the use of @MappedBy annotation?
13. By default how many tables will get created in different types of entity relationships?
14. How many tables are min required for a ManyToMany relationship?

***Rest:***

1. Difference between @Controller & @RestController?
2. Explain the term ‘Statelessness’ with respect to RESTful WEB service & Enlist advantages and disadvantages of ‘Statelessness’.
3. What are the best practices that are to be followed while designing RESTful web services?
4. What HTTP Status Code 200,201,204,304,400,401,404,409 & 500 states?
5. Explain different HTTP methods like Get, PUT, POST, Delete, Patch, Head & Options?
6. How you are doing API versioning in your application? What are the different approaches available for that with their benefits and drawbacks?
7. How can you make your Rest APIs secure?
8. Which approach you are using to define contract documents of your APIs?
9. What is @RestController?
10. What are the HTTP verbs?
11. What is the difference between POST and PUT?
12. Is it possible to fetch data using POST?
13. How to map a URI to a resource method?
14. What is a content-type?
15. How did you do authentication?
16. How to make a HTTPS URI?
17. Follow-up : What maps the URI to the method?
18. Difference between SOAP and REST
19. When to choose REST
20. What is the response code used indicate the successful processing of request.
21. What approach should be used while designing URI’s? Design one for GET/PUT/POST/DELETE.
22. How can you identify DDOS attack and what approach will you apply in order to handle the DDOS attack for your rest service?
23. Basics of any of security concepts like OAuth 2 & JWT?

***Database:***

1. How to optimize the performance of DB queries? What is query execution plan? How can you generate it?
2. What are indices? Advantages?
3. Is it good to have too many indices?
4. What is primary key?
5. Some SQL query to fetch the names of department and the count from the table where the employees’ count is more than 10.
6. How primary key, unique key and foreign keys are different.
7. How does index work.
8. What is the best data type to use as an index.
9. Difference between Stored Proc and Functions
10. What is Normalization? Why use normalization?
11. What are SQL joins and their types?
12. Delete, Truncate vs Drop command

***Design Pattern:***

1. Have you used any Creational, Structural or Behavioural design pattern in your application? Explain the use case as per your application?
2. With use case, explain at least one design pattern from Creational, Structural & Behavioural types.
3. Best way to implement Singleton Design Pattern.
4. What are different design principles? Can you specify any 4 design principles apart from the SOLID design principles?
5. Give example of decorator design pattern in Java? Does it operate on object level or class level?
6. What is façade design pattern and its usage?
7. What is flyweight design pattern and where is it used in JAVA API? (String Memory allocation).

Apart from the above topics, be prepared on JMS*(e.g. Durable vs Non-Durable Topics etc)*, Caching *(InMemory cache like Redis, MemCache & Distributed Cache)*, Profiling & Performance tuning at code, DB as well as JVM level*(i.e. defining proper heap size, efficient GC etc)*.

Knowledge about the Build tool like maven and continuous integration like Jenkins/TeamCity should be brushed up