

Java Questions and Answer asked in interview

L1 – Capgemini

Q. What's your current project, tech stack you are using and day-to-day roles and responsibility?

Q. If I want to communicate from one service to another services what are the approach we can use? – Service Mesh

Q. RestTemplate is a class or an Interface? – Class. Is it Synchronous or Asynchronous one? – RestTemplate is Synchronous | AsyncRestTemplate is Asynchronous

Q. After fixing the bug, how many stages we need to check the fix?

Q. Suppose I want to make some enhancement what is the process? – Interested in JIRA creation – Feature branch creation – merge with develop branch – deployment in DEV-> QA-> UAT -> PROD

Q. How to create feature branch from git bash? - git checkout -b feature-branch-name

Q. How to send the data via GET mapping to RestController? – He is interested in figuring out the answer on @PathVariable @RequestParam.

Q. What are all the pre-defined functional interfaces in java 8?

Q. What is the use of default method? Why they introduced default method? What code goes under default method? - Default methods can contain reusable code that can be shared among multiple classes implementing the interface.

Q. What is the default capacity of ArrayList? -10

Q. What is the annotation of Global Exception in spring boot? - @ControllerAdvice, @ExceptionHandler, @ResponseStatus

Q. What's the difference between @Configuration and @Bean? – Class level and method level | Defined within Configuration class

Q. Can you please explain the thread life cycle? - New -> Runnable -> Running -> [Blocked/Waiting/Timed Waiting] -> Terminated

Q. Is @FunctionalInterface is required while defining Functional Interface?

TCS

Q. Exception handling hierarchy and how to throw custom exception? - public class CustomException extends Exception {

public CustomException(String message) {

super(message);

}

}

Q. **MVC – flow.** – *DispatcherServlet* [receives the request] -> map it to *HandlerMapping* [map it to appropriate controller based on the URL passed] -> Processing of data, service-layer along with repository -> prepare data for the view -> into *Model* -> *ViewResolver* [based on logical name] -> View populated with data -> Send back as HTTP response to client

Q. Explain yourself

Roles and Responsibilities:

As a Staff Engineer, I was responsible for leading the development of enterprise-level Java applications, ensuring code quality and adherence to coding standards. I collaborated with teams to analyze requirements, design scalable solutions, and implement features using Java SE/EE technologies.

Tech Stack:

My tech stack includes Java 8/11, Spring Boot, JPA, Maven, BitBucket, DB2, OpenShift and LightSpeed. I have experience with RESTful APIs, microservices architecture, and testing frameworks like JUnit and Mockito.

Projects Worked On:

Currently I am working in one of the Citi Commercial banking portal. I led the backend development using Spring Boot and JPA, integrated with external APIs for data retrieval, and implemented security measures using Spring Security. We achieved a 30% performance improvement by optimizing database queries and implementing caching strategies as compared to Legacy system which uses IBM Mainframe and Wrapper zOS calls. This project improved scalability and reduced downtime during updates using strategy at deployments.

Q. Data Structure use – ArrayList – why we use arrayList?

L1 - Zemoso Technology

Q. Major and minor GC? – targets old generation and collects long-lived object | target young generation and collect short-lived object [Eden space, S0, S1]

Q. Security implementation in your application. – LDAP, Basic Authorization/Authentication, CSRF and CORS.

Q. Java 8 coding questions – Map based questions.

Q. How to improve the performance of the application – Optimize Code [appropriate usage of data structure], DB query optimization, Caching, Resource optimization, optimize external API calls, Monitoring and Scaling.

Q. How to convert monolithic to micro-services. – Decompose functionality, Define service boundaries, apply domain driven services approach to single out functionality, RestAPI or GraphQL, Orchestration [Kubernetes], ELK, Service discovery, Scalability.

Q. How to solve $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \dots = \frac{x}{4}$ find the value of x. -
double sum = 0.0;

```
for (int i = 0; i < terms; i++) {  
    double term = (i % 2 == 0) ? 1.0 / (2 * i + 1) : -1.0 / (2 * i + 1);  
    sum += term;
```

```
}
```

```
return sum;
```

Q. What is livelock in multithreading? – Situation where one thread wants to acquire other lock held by one thread which wants to acquire lock held by other thread they are not locked but unable to make progress.

Q. Can I again start the thread again if it is already completed? - No

Q. What are the feature of Java 17? (Current version in my project) – Sealed Classes, enhanced Switch statements, Applet API – Deprecation, Pseudo Random Number for cryptographic and non-cryptographic use, Jarkarta APIs

Q. Does the backend developer need to know about frontend development? – Better API design, Communication and Collaboration with team across stacks, Debugging and Monitoring.

Q. Why time API was again introduced in Java 8 if it is already there? – address shortcoming of java.util.* time classes like Zoned API introduced, Daylight saving introduction, Immutable classes, collaborate with lambda and Stream APIs,

Q. What are the data Structure you have used? – Arrays, ArrayList, HashMap, HashSet, Stack and Queues, BST, MinHeap-MaxHeap, Graphs.

Q. Binary Tree and Binary Search Tree?

Q. How to find the height of BST?

L2 - Zemoso Technology

Q. You seems to have product ownership? Ask about Resume under Axis bank – Understand and draft a better answer.

Q. You seems to have written reduction in cash burnout by 20% - how you have achieved it? Resume under ICICI bank – understand and draft a better answer.

Q. If you were asked to make spring boot better from developer perspective what are the 3 improvement that you will make? – 1. Enhanced Configuration

Management 2. Improved Developer tooling and integration 3. Streamlined dependency management.

Q. What do you understand by connection pooling? - Connection pooling is a technique used to manage database connections efficiently in a multi-threaded environment, such as web applications. It involves creating a pool of reusable database connections that can be shared among multiple clients, rather than opening and closing a new connection every time a database access is needed. This approach significantly improves performance and resource management.

Q. Suppose you were asked to design food delivery app, what are all the microservices you will design? Which microservices do you think needs to be scalable? – User Service, Restaurant Service, Inventory service, Notification Service, User Review and Rating Service, Delivery and tracking service, Analytics Service, Search Service, Payment Service, Order Service. [*User Service, Payment Service, Order Service, Restaurant Service*]

Q. Any new technology you learned in last six months?

Q. Did you use chatGPT, in what ways? Gen AI?

L1 – Nagarro

Q. Can you give me a brief about yourself?

Q. What are all the challenges in MS architecture? – Complexity in Distributed System [Network latency and Fault tolerance], Management of transaction across MS become complex, Monitoring and logging, Deployment complexities.

Q. How will you handle communication among MS?

Q. What's the difference between RestTemplate and Feign Client? –

RestTemplate - Imperative, explicit HTTP request handling, better suited for detailed control and customization of HTTP requests

Feign Client - Declarative, Annotation based implementation, best suited for frequent inter-service communication.

Q. How would be handle fault tolerance in MS architecture?

Q. What is circuit breaker? Which library you used to implement Circuit Breaker?

– **Closed** [services are allowed to pass through], **Open** [request failed immediately] and **Half Open** [A limited number of requests are allowed to pass through to check if the service has recovered. If successful, the circuit breaker transitions back to the closed state; otherwise, it returns to the open state.]

Q. How would you implement distributed tracing in MS architecture? – Jaeger, Zipkin and Micrometers

Q. What are the drawbacks of Saga Design Pattern? – Complexity in the communication in MS if multiple services involved, Maintenance increases, Long running transactions, coordination overhead.

Q. Have you worked on Multi-threading? What is Future Object in Multi-threading? - Future is an interface that represents the result of an asynchronous computation. It allows you to interact with the result of a computation that is executed asynchronously, typically in a multi-threaded environment

Q. How will you create threadPool in Multi-Threading?

Q. What is AtomicInteger in Multi-threading? - AtomicInteger provides a simple and efficient way to handle atomic operations in multi-threaded applications, ensuring thread safety without the complexity and overhead of explicit synchronization | Methods – incrementAndGet(), GetandIncrement(), get(), set(int val)

Q. What is PATCH in Rest API?

Q. What is CORS errors? - CORS errors occur when a web application attempts to make a request to a different origin and the server does not permit it | Common causes include missing or incorrect Access-Control-Allow-Origin headers, improper handling of preflight requests, and issues with credentials | Fixing CORS errors involves configuring the server to include the appropriate CORS headers in its responses, ensuring that preflight requests are handled correctly, and properly managing credentials.

Q. Have you worked on K8s or Docker? What's the difference between Secret and Sealed Secret or Seal Secret? - Kubernetes Secret is an object that stores

sensitive data such as passwords, OAuth tokens, and SSH keys. Secrets are base64-encoded and can be used by pods to access confidential information without embedding it in the pod specification or container image | Sealed Secrets are a Kubernetes-native way to encrypt secrets before storing them in a git repository or other version-controlled environment. This ensures that sensitive information is not exposed, even if the repository is compromised.

Q. How you provide value in Secrets? - You can create a Secret by defining a YAML manifest and applying it to your cluster. Values in the Secret must be ***base64-encoded***.

Q. How will you define HorizontalPodScaling in K8s? - Horizontal Pod Autoscaling (HPA) in Kubernetes automatically adjusts the number of pods based on observed metrics, such as CPU utilization. It helps ensure efficient resource utilization and application performance by scaling pods in or out as needed. Defining HPA involves specifying the target resource, the scaling metrics, and the desired scaling limits.

apiVersion: autoscaling/v2beta2

kind: HorizontalPodAutoscaler

metadata:

name: my-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: my-deployment

minReplicas: 1

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

Q. You have to define CPU and Memory, what parameter you will define in deploymentConfig? - To define CPU and memory resource requests and limits in a Kubernetes Deployment configuration, you use the resources field within the container specification. This allows you to specify the amount of CPU and memory resources that the container will request and the maximum amount it is allowed to use.

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-deployment

spec:

replicas: 3

selector:

matchLabels:

app: my-app

template:

metadata:

labels:

app: my-app

spec:

containers:

- name: my-container

image: nginx

resources:

requests:

memory: "64Mi"

cpu: "250m"

limits:

memory: "128Mi"

cpu: "500m"

Q. What is the difference between cluster IP and node port? – Within cluster |
Outside cluster

Key Differences

Feature	ClusterIP	NodePort
Access Scope	Only accessible within the cluster	Accessible from outside the cluster
Default Port	Cluster-internal IP and port	Each Node's IP and a static NodePort
Use Case	Internal communication between services	Exposing services to external clients
Configuration	Default Service type	Requires explicit <code>type: NodePort</code>
Traffic Flow	Directly to pods via ClusterIP	Through NodePort to ClusterIP, then to pods

Q. Which ORM you are using Hibernate or JPA? I have an Employee Entity, I want to fetch employees whose age is greater than 35 without writing any query? - `List<Employee> findByAgeGreaterThan(int age);`

Q. What are the transaction propagation? – Achieved via `@Transactional` |
Propagation level – **REQUIRED** [by default – Whole transaction will be based on first declared `@Transactional`], **REQUIRED_NEW** [discard the on-going `@Transactional` and create a new for the scope], **NESTED** [Create a save point at each called `@Transactional`], **MANDATORY** [If parent should call `@Transactional`

otherwise child will throw an exception], **NEVER** [opposite of MANDATORY], **NOT_SUPPORTED, SUPPORTED** [Work in @Transactional or if not present]

Q. How will you create a query using JPA? – JPQL [`@Query("SELECT e FROM Employee e WHERE e.age > 35")`]

`List<Employee> findEmployeesOlderThan35();`] | **Criteria API | Native Query** [`@Query(value = "SELECT * FROM Employee WHERE age > 35", nativeQuery = true)`]

`List<Employee> findEmployeesOlderThan35();`] | **Spring Data JPA Query Method** [`// Method naming convention for query generation`]

`List<Employee> findByAgeGreaterThan(int age);`]

Q. What is the difference between nativeQuery and Entity Query? - In summary, JPQL is ideal for most use cases due to its database independence and type safety, while native queries are useful for more complex scenarios where direct SQL access is necessary.

Q. What are the new feature in Java 8?

Q. What are the new thing introduced in DateTime api as a part of Java 8?

Q. What is predicate?

Q. have you worked on Junit? What is Mockito.verify()? How will you mock a void method?

L2 – Nagarro

Q. Introduce yourself?

Q. What do you understand by mutable and immutable? Suppose I have one POJO class consisting of one is Integer and other one is hashmap, how to make this POJO class immutable?

Q. What is equals and hashCode contract?

Q. Suppose you have list of numbers, I wants to remove the duplicate elements and sort the elements in reverse order. How to do it?

Q. What is functional Interface?

Q. What is Singelton design pattern?

Q. Suppose you have public void method(), how you can write Junit method for it?

```
java
@RunWith(SpringRunner.class)
public class MyClassTest {
    @Test
    public void testMyMethod() {
        // Arrange
        MyClass myClass = new MyClass();

        // Act
        myClass.method();

        // Assert
        // Add assertions to verify the expected behavior
        // For example:
        // assertTrue(myClass.someField == expectedValue);
    }
}
```

Q. How you are going to start three threads? Parallel call calling three different services? - You can start three threads in Java to make parallel calls to three different services using the ExecutorService and Callable interface

Q. How many ways we can define dependency injection? In which scenario you will use different dependency injection? – Field, setter, constructor | Setter – not mandatory, can be mutable, remove cyclic dependency | Constructor – Mandatory, immutable, can lead of cyclic dependency.

Q. How can you define two DB in one spring boot application?

1. Configure Multiple Data Sources

2. Create DataSource Configuration
3. Create JPA EntityManagerFactory Beans
4. Use EntityManager in Repositories
5. Use Multiple Transaction Managers (Optional)

Q. How to do Global exception in spring boot?

Q. What is the difference between @Service and @Component annotation? –
Specialized annotation | Stereotype annotation.

Q. Can I create Rest API by using @Controller?

Q. How would you define Entity class? - @Entity, @Table, @Id, @Column, No arg constructor, All arg constructor

Q. What is lazy and eager loading?

Q. What is named Query in hibernate?

Q. Can we use nativeQuery in JPA? So then how we are going to map the response in our POJO class? –

- Use `@SqlResultSetMapping` and `@ConstructorResult` for a more complex mapping.
- Use `@SqlResultSetMapping` with field-based mapping for simpler cases.
- Use `@Query` in Spring Data JPA for straightforward native query mappings.

Q. What is the groupBy clause in Oracle?

Q. How many ways we can communicate between two MS?

Q. What is API gateway?

Q. Are you hardcoding the URLs for making RestAPI calls?

Q. Suppose A MS calling B MS calling C MS, if C fails, how you are going to handle this failure scenario?

Q. Any experience with Message driven or event driven communication?

Q. How you are deploying your application?

L1 – Epam System

Q. Could you please describe me SOLID principle?

5. Dependency Inversion Principle (DIP)

- **Definition:** High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

Q. Could you please tell me the difference between intermediate and terminal operation in stream API? What is lazy evaluation in stream API?

Q. How hashmap works internally?

- `HashMap` uses an array of buckets.
- Keys are hashed to determine the bucket index.
- Collisions are handled using linked lists or trees within the buckets.
- The structure dynamically resizes and rehashes entries when needed.

Q. What's the difference between ArrayList and LinkedList?

L1 – Nice Actimize

Q. Do you know about immutable in Java? How to make one? Suppose I have 1000 custom variable and get method creating deep copy will degrade my system performance, what we can do in order to avoid this? - Use Immutable Collections or Builder Pattern or Caching Strategy or Use a Lazy Initialization Strategy

Q. Do you know how hashmap works? Do you know how hashset works? HashSet internally uses HashMap, if we can achieve HashSet functionality by hashmap then what is the need of HashSet? Why java introduces it? - In summary, while HashMap can be used to achieve HashSet functionality, HashSet provides a more efficient, type-safe, and simplified interface for working with sets,

making it a necessary part of the Java Collection Framework | straightforward interface for adding and removing elements.

Q. Suppose I have one temporary table having two columns with no primary key, no foreign reference, not unique, no constraint. Now, how will I map this Entity in JPA if we are not able to use @Id? How to handle such scenarios? – Using @Embedded annotation to create composite key or can use @IdClass [Class level annotation]

```
java
@Entity
@Table(name = "TEMP_TABLE")
public class TempTableEntity {

    @EmbeddedId
    private TempTableId id;

    // Other fields, if any

    // Getters and setters
}

@Embeddable
public class TempTableId implements Serializable {
    private String column1;
    private String column2;

    // Getters and setters, equals, hashCode
}
```

Q. You know Autowire, by default it using Singleton, If we want to create multiple objects of any bean in spring, how can we achieve it? -

@Scope("prototype") or @Lookup

Q. Were you faced with any difficulty while using Date or Calendar in Java 7, as they have introduced new APIs in Java 8? Any practical challenge faced? –

Thread Safety, Limited operations, simplified zoning date | Poor API design like getYear() starts from year 1900, issue while adding days to date.

L1 – Convera

Q. What's the difference between Microservices and Rest APIs? Are they both the same? - microservices are the building blocks of an application, while REST APIs are the communication mechanism between these microservices.

Q. Where and why we use API gateway? Have you implemented if so, which framework did you use? - API gateways are crucial components in microservices architecture, providing a single entry point for clients, simplifying communication, enhancing security, and optimizing performance.

Q. What is rate limiting? - Rate limiting is essential for managing network resources, ensuring the availability and performance of networks, and protecting against various types of attacks, including DoS, brute force, and API abuse. It can also help prevent resource exhaustion, reduce costs, and improve the user experience by reducing delays and improving responsiveness

Q. How to do inter-service communication in MS?

Q. Explain fault tolerance mechanisms in your MS.

Q. How do you connect to your DB in your MS?

- a. Configuration – Properties [spring.datasource , spring.jpa]
- b. Dependency management - spring-boot-starter-data-jpa
- c. DataSource Configuration - DataSource bean creation
- d. Repository layer
- e. Entity class

Q. How are you doing versioning of your application?

L1 – TCS

Q. What are the design pattern you have used in your MS? – Event Sourcing, Database per Service, Sidecar.

Q. How you are handling distributed transaction? – 1. Two Phase commit 2. Saga design pattern [Orchestrator – centralized controller | Choreographic – each microservice takes responsibility] 3. Event Sourcing.

Q. In which scenario we go for Saga design pattern? Do you know Orchestrator do in saga design pattern? - In the orchestration-based Saga pattern, an orchestrator is a central coordinator responsible for managing the sequence of steps involved in the Saga. The orchestrator directs the workflow, invoking each participating microservice to perform its part of the transaction and handling any compensations in case of failures.

Q. In monolithic application, how we handle transaction?

Q. How we enable transaction management in monolithic application? What configuration is required to enable it? How to enable in MS? -

@EnableTransactionManagement [at @Configuration] | @Transactional [at method level]

Q. What are the main advantages of MS? – Scalability, Flexibility in technology stack, Improved security, Fast time to market.

Q. Explain the latest PROD issue you faced and how you resolved it?

Q. Have you faced OutOfMemory error in PROD? Do you know why it comes?

How to fix it? Do you know the configuration to change? - java -Xms512m -Xmx4g -jar your-application.jar | Memory Leaks, Insufficient Heap Memory

Q. In what scenario we used Circuit Breaker? How to configure it? -

@CircuitBreaker

Q. Have you heard about SOLID principle? Can you tell me what is liskov-substitution principle?

Q. Do you know Docker? Why it is required? What is the role of kubernetes?

📦 Docker simplifies the creation and management of individual containers.

📦 Kubernetes takes container management to the next level by orchestrating and managing large numbers of containers across clusters of machines.

Q. Can you give some example of functional interfaces?

Q. how to get second highest value from list of integer?

Q. Suppose I have list of string contain 1 in the string, how to get only those string?

Q. How to remove duplicates from list of integer?

Q. What is the spring boot version you are using currently?

Q. Can you tell me the difference between @Autowire and @Qualifier annotation?

Q. What are the best practices you follow while designing REST APIs? – filter, pagination and sorting | Appropriate use of HTTP verbs | Rate Limiting

Q. Do you know any Java 11 features?

Q. Have you worked on security part in your application? What is the structure of JWT token? What is the expiration time for your application token?

Q. How we create an immutable object? How you create immutable class?

Q. How to find any code vulnerability or security issue in your application?

Q. What's the Junit version you are using? – 4.11 or Junit 4

L2 – Epam

Q. Do you have any experience working in AWS? – Understand the basic questions and answer

Q. What are the new feature released in Java 8? – Method reference, Optional class, default and static methods in interface.

Q. Can you please explain completableFuture? - A `CompletableFuture` in Java is a class that extends `Future` and implements `CompletionStage`. It represents a future result of an asynchronous computation and provides a powerful and flexible way to write asynchronous, non-blocking code.

Q. How does GC work in heap memory?

- **Heap Memory:** Divided into Young Generation (Eden, Survivor spaces), Old Generation, and Metaspace.
- **GC Types:** Serial, Parallel, CMS, G1, ZGC, Shenandoah.
- **GC Phases:** Marking, Sweeping, Compacting.
- **Process:** Minor GC for Young Generation, Major GC for Old Generation.

Q. What's the difference between Runnable and Callable interface?

Category	Runnable	Callable
Availability	Available since JDK 1.0	Introduced in Java 5 (JDK 1.5)
Return Type	<code>void</code>	Returns a result using <code>Future</code>
Exceptions	Cannot throw checked exceptions	Can throw checked exceptions
Methods	<code>run()</code>	<code>call()</code>
Usage	Suitable for fire-and-forget tasks	Suitable for tasks that need to return a result
Thread Execution	Can be executed by <code>Thread</code>	Can be executed by <code>ExecutorService</code>
Future	No	Returns a <code>Future</code> object

Q. What's the difference between Synchronized and ReentrantLock?

Synchronised Keyword	ReentrantLock
The synchronised keyword does not support fairness. Once released, any thread can acquire lock; no preference can be stated.	When constructing a ReentrantLock instance, you can specify the fairness property to make it fair. In the event of a conflict, the Fairness attribute gives the longest waiting thread a lock.
There is no solution to reduce thread blocking while using lock-in Java apps.	ReentrantLock has a handy tryLock() method that only acquires the lock if it is available and not held by another thread.
A thread can be blocked for an endless period waiting for a lock, and there was no method to regulate it.	ReentrantLock has a lockInterruptibly() method that can be used to interrupt a thread that is waiting for a lock.
No way to get a list of all threads waiting for the lock.	ReentrantLock additionally has a method for getting a list of all threads waiting for a lock.

Q. What's the difference between HashMap and ConcurrentHashMap?

Q. Difference between ArrayList and LinkedList?

Q. Can you let me know the use case of ThreadLocal? - ThreadLocal is a special class in Java that provides thread-local variables. These variables are each thread's private version of a variable that can be modified without affecting other threads.

Scenarios: User Session Information, Database Connections, Transaction Management, Logging Context

Q. Can you let me know what Spring AOP is? - Spring AOP (Aspect-Oriented Programming) is a module of the Spring Framework that provides a robust and elegant solution to address the challenges posed by cross-cutting concerns. It allows you to modularize these concerns, thus enhancing code maintainability and reducing redundancy

Q. What is spring inversion of control? - Spring Inversion of Control (IoC) is a design principle that describes the process of inverting the flow of control in a system. It is a way to decouple the execution of a certain task from its implementation. This means that the control of creating and instantiating objects is given to the Spring IoC container, rather than the developer.

Q. Can you let me know how to create ThreadPool in Java? What's the difference between corePoolSize and maxPoolSize?

```

int corePoolSize = 2;
int maximumPoolSize = 4;
long keepAliveTime = 10;
TimeUnit unit = TimeUnit.SECONDS;
BlockingQueue<Runnable> workQueue = new LinkedBlockingQueue<>(10);

ThreadPoolExecutor threadPool = new ThreadPoolExecutor(
    corePoolSize,
    maximumPoolSize,
    keepAliveTime,
    unit,
    workQueue
);

```

Difference between corePoolSize and maxPoolSize

- **corePoolSize:** This is the minimum number of threads that the pool will maintain. If fewer than this number of threads are running, the pool will create new threads to handle the tasks. If more than this number of threads are running, the pool will not create new threads until some of the existing threads are idle.
- **maxPoolSize:** This is the maximum number of threads that the pool will maintain. If the number of threads exceeds this value, the pool will not create new threads. If the number of threads falls below this value, the pool will create new threads to handle the tasks.

Here are some key differences between `corePoolSize` and `maxPoolSize`:

- **corePoolSize** is the minimum number of threads that the pool will maintain, while `maxPoolSize` is the maximum number of threads that the pool will maintain.
- **corePoolSize** is used to ensure that there are always a certain number of threads available to handle tasks, while `maxPoolSize` is used to prevent the pool from growing too large.
- **corePoolSize** is used to ensure that the pool has enough threads to handle a certain number of tasks, while `maxPoolSize` is used to prevent the pool from growing too large and consuming too many resources.

Q. Do you know how to use the swagger? Can you let me know how the RESTful API looks like?

A RESTful API typically consists of the following key elements: ²

- **Resources:** The entities or data that the API exposes, such as users, products, or orders. Each resource has a unique URL that clients can access.
- **Operations:** The HTTP methods (GET, POST, PUT, DELETE) that clients can use to interact with the resources. For example, a GET request to `/users`` would retrieve a list of users.
- **Parameters:** The data that clients need to provide when making requests, such as query parameters, request bodies, or path parameters.
- **Responses:** The data that the API returns in response to client requests, including status codes and response bodies.

Q. What design pattern you are familiar with? How did you use observer design pattern in your project?

I am familiar with the observer design pattern. The observer pattern is a behavioral design pattern that defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. This pattern is commonly used to implement distributed event handling systems.

In my projects, I have used the observer pattern to implement event-driven systems where multiple objects need to react to changes in another object. For example, in a system where multiple UI components need to update their state based on changes in a data model, the observer pattern can be used to notify each component of the changes and update their state accordingly.

Q. Can you briefly explain what SOLID principle is?

Q. What's the benefits of MS?

Q. How to handle distributed transaction?

- **Two-Phase Commit:** Ensures strong consistency but can be slow and has coordinator failure issues.
- **Three-Phase Commit:** Reduces blocking but is complex and rarely used.
- **Saga Pattern:** Provides eventual consistency with better scalability and fault tolerance.
- **Transactional Messaging:** Ensures ordered and reliable event processing.
- **Two-Phase Locking:** Ensures serializability but can lead to deadlocks.

Q. What's the difference between 2-Phase and 3-Phase design pattern?

Feature	2-Phase Design Pattern	3-Phase Design Pattern
Concept	Ensures that all participants in a transaction agree to commit or abort the transaction.	Extends the 2-Phase Commit to handle additional coordination and fault tolerance.
Phases	Two phases: Prepare and Commit/Abort	Three phases: Prepare, Pre-Commit, and Commit/Abort
Phases Explained	Prepare Phase: Participants prepare to commit and notify the coordinator. Commit/Abort Phase: Coordinator decides and notifies all participants to either commit or abort.	Prepare Phase: Participants prepare to commit and notify the coordinator. Pre-Commit Phase: Coordinator sends a pre-commit request to all participants. Commit/Abort Phase: Coordinator sends the final commit or abort decision to all participants.
Coordinator's Role	Central coordinator collects votes and makes the final decision.	Central coordinator collects votes, sends pre-commit, and then final commit/abort.
Participants' Role	Participants prepare resources for a potential commit and follow the coordinator's final decision.	Participants prepare resources, then lock resources upon pre-commit, and follow the final decision.
Fault Tolerance	Limited fault tolerance; if the coordinator fails after preparing, participants may be left in an uncertain state.	Improved fault tolerance; the pre-commit phase reduces the window where participants are in an uncertain state.
Usage	Commonly used in distributed systems where atomicity is required.	Used in systems requiring higher reliability and fault tolerance, such as distributed databases.
Complexity	Simpler and less overhead compared to 3-Phase.	More complex with additional overhead due to the extra phase.
Communication Overhead	Lower due to only two rounds of communication.	Higher due to the additional pre-commit round.
Performance Impact	Generally better performance due to fewer phases and less communication.	Potentially slower due to the additional phase and communication steps.
Example Use Case	Distributed transaction management in simple distributed systems.	Distributed databases and high-reliability systems where reducing the risk of uncertain states is crucial.



Q. Could you please elaborate on Saga design pattern?

Q. How can we deal with distributed tracing? Can you let me know what to use Zipkin and configuration?

Dependencies:

<dependencies>

<!-- Spring Cloud Sleuth -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-sleuth</artifactId>

</dependency>

<!-- Spring Cloud Sleuth Zipkin -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-sleuth-zipkin</artifactId>

</dependency>

</dependencies>

Properties:

spring.zipkin.base-url=http://localhost:9411

spring.sleuth.sampler.probability=1.0

View Traces in Zipkin:

Access the Zipkin UI at <http://localhost:9411>. You can search for traces, view trace details, and analyze the latency and dependencies of your microservices.

Q. are you aware of ELK?

Q. Do you know how to use Jenkins? Do you edit the pipeline of the script?

Q. Do you use Docker? Do you edit Docker file? Do you know what is Docker compose?

- **Docker:** A platform for containerizing applications.
- **Dockerfile:** A script for building Docker images.
- **Docker Compose:** A tool for managing multi-container Docker applications.

By using Docker and Docker Compose, you can simplify the process of managing and deploying your applications in a consistent environment.

Q. What's the difference between deployment and Service?

Feature	Deployment	Service
Purpose	Manages the deployment and scaling of a set of pods.	Exposes a set of pods as a network service.
Main Function	Ensures the desired state of application replicas.	Provides stable networking to reach pods.
Pod Management	Creates and manages pods through ReplicaSets.	Does not create pods; selects existing pods.
Rolling Updates	Supports rolling updates and rollbacks.	Not applicable.
Scaling	Allows scaling up/down the number of pod replicas.	Does not handle scaling.
Selectors	Uses selectors to manage which pods it should manage.	Uses selectors to determine which pods it routes traffic to.
Networking	Not responsible for exposing pods to the network.	Provides stable DNS name and load balancing.
Types	Only one type: Deployment.	Several types: ClusterIP, NodePort, LoadBalancer, ExternalName.
Lifecycle Management	Handles the complete lifecycle of pods, including updates and rollbacks.	Focuses on maintaining network access and load balancing.
Example Use Case	Ensuring a web application runs with multiple replicas, updating versions without downtime.	Exposing the web application to other services or external traffic.

- **Deployments:** Focus on managing and scaling pod replicas, ensuring the application runs as desired, and handling updates and rollbacks.
- **Services:** Focus on providing network access to pods, offering stable IP addresses and load balancing.

Q. What is Ingress of kubernetes?

In Kubernetes, an Ingress is a resource that manages external access to services within a cluster, typically HTTP and HTTPS. It provides a way to route traffic from outside the cluster to services inside the cluster based on rules you define. Ingress can perform functions such as load balancing, SSL termination, and name-based virtual hosting.

Ingress vs. Services

Feature	Ingress	Service
Purpose	Manages external access to services based on rules.	Exposes a set of pods within the cluster.
Types of Traffic	HTTP and HTTPS	Any TCP/UDP
Load Balancing	Yes	Yes
TLS/SSL	Yes	No
Path-Based Routing	Yes	No
Host-Based Routing	Yes	No



- **Ingress:** Manages external access to Kubernetes services, handling HTTP and HTTPS routing, load balancing, SSL termination, and more.
- **Ingress Controller:** Implements the Ingress resource, fulfilling the rules defined in the Ingress resource.

Q. What is the configMap of K8s?

In Kubernetes, a ConfigMap is a resource used to store configuration data in key-value pairs. It allows you to decouple configuration artifacts from image content, making applications more portable. ConfigMaps are designed to provide a way to manage environment-specific configuration variables without embedding them in your application code.

- **ConfigMap:** A Kubernetes resource used to store configuration data in key-value pairs.
- **Purpose:** Decouple configuration from application code, making applications more portable and easier to manage.
- **Usage:** ConfigMaps can be used as environment variables or mounted as files in pods.
- **Management:** ConfigMaps can be created, viewed, updated, and deleted using Kubernetes commands.

Q. What’s the difference between Mock and Spy in Mockito?

Feature	Mock	Spy
Definition	Simulated object that mimics behavior of real objects	Wraps an existing instance of a class to call real methods while allowing stubbing
Usage	To create objects that simulate behavior of real dependencies	To verify behavior of real objects but allow stubbing/mockng of certain methods
Default Behavior	Returns default values (null, 0, false) for all methods unless stubbed	Calls real methods unless explicitly stubbed
Stubbing	Requires explicit stubbing of methods to define behavior	Allows partial stubbing; real methods are called by default
Creation	Created from scratch with no real implementation	Wraps an existing real instance
Typical Use Case	When you need a dummy object with no real implementation	When you need a real object but want to control some of its behavior
Example	<code>`List<String> mockedList = Mockito.mock(List.class);`</code>	<code>`List<String> spyList = Mockito.spy(new ArrayList<>());`</code>
Example Stubbing	<code>`when(mockedList.get(0)).thenReturn("first element");`</code>	<code>`doReturn("second element").when(spyList).get(1);`</code>
Behavior Verification	Verifies interactions with a mock object 	Verifies interactions with real methods, allowing partial stubbing 

Q. Are you familiar with Cucumber? - Cucumber is a tool for running automated acceptance tests written in a behavior-driven development (BDD) style.

Q. What's the messaging queue you are using? Can you let me know what is Topic and Partition of kafka?

- **Topic:** A named stream of records, used to organize data.
- **Partition:** A division of a topic, enabling scalability, parallelism, and fault tolerance.

Kafka's architecture, with topics and partitions, allows it to handle large-scale, real-time data feeds efficiently and reliably. If you have any more questions or need further details on Kafka or messaging

Q. In what scenario we can use kafka and how to guarantee the message delivery in Kafka?

Kafka is suitable for a variety of use cases that require high-throughput, real-time data processing and reliable message delivery. By configuring replication, acknowledgment levels, idempotent producers, and proper offset management, Kafka can provide robust guarantees on message delivery, ensuring data integrity and consistency across distributed systems.

Q. Do you know how to implement Kafka for message ordering?

By carefully designing your Kafka topic and producer to use appropriate partition keys, you can maintain message ordering while leveraging Kafka's distributed architecture.

Q. in RDBMS, how many ways we can optimize the query? – Normalization and de-normalization whenever needed, use EXIST instead of IN.

Q. What's the difference between Index and Composite Index?

Q. What's the execution plan of DB?

Understanding and interpreting execution plans is essential for database performance tuning. By analyzing execution plans, you can identify inefficient operations, optimize queries, and ensure that the database is using the most efficient path to retrieve data.

Q. What's are all the annotation provided by Spring JPA? - @Transient, @ManyToMany, @Procedure, @EnableJpaRepostiry, @PersistentContext, @PersistentUnit, @NotNull, @Size, @Email, @Pattern.

Q. What's the benefits of having NoSQL DB against SQL DB?

In summary, NoSQL databases offer advantages in scalability, schema flexibility, performance, data model support, availability, fault tolerance, cost-effectiveness, and suitability for specific use cases such as big data analytics and real-time applications. However, they also require careful evaluation and consideration to ensure they align with the application's requirements and data management needs.

Q. Are you familiar with Redis?

Overall, Redis is a powerful and versatile tool used in a wide range of applications, including web applications, real-time analytics, messaging systems, caching layers, session stores, and more. Its simplicity, speed, and rich feature set make it a popular choice for developers and organizations looking to improve performance and scalability in their applications.

Q. Can you let me know what spring boot actuator is?

Q. Can you let me know what spring boot starter is?

Overall, Spring Boot starters simplify the development process by providing a streamlined way to add dependencies, configure applications, and follow best practices within the Spring ecosystem. They are an essential part of Spring Boot's convention-based approach to building production-ready applications with minimal effort.

Q. Can you let me know how to use conditional autowire? Do you know @ConditionalOn?

Q. Can you let me know the ceremony of agile methodology?

1. Sprint Planning
2. Stand up meeting
3. Sprint Review
4. Sprint Retrospective

Q. What's the recent challenge you faced and how you overcome it?

Q. Can you explain the architecture of your current project?

Q. What are the advantages of using domain driven design in MS?

Here are the major five advantages of using Domain-Driven Design (DDD) in Microservices (MS) architecture:

- Clear bounded contexts
- Enhanced alignment with business needs
- Decoupling
- Independent scaling
- Improved maintainability and evolution

Q. How can you monitor the health of microservices?

Certainly! Here are 5 major bullet points on how to monitor the health of microservices, including AppDynamics:

- Implement health check endpoints (`/health`) in each microservice to report status.
- Use metrics collection tools like Prometheus and Grafana for performance monitoring.
- Utilize centralized logging using ELK Stack or Fluentd for aggregation and analysis.
- Employ distributed tracing tools such as Jaeger or Zipkin to trace requests across microservices.
- Leverage APM tools like AppDynamics for application performance monitoring and deep dive analysis.

Q. Have you heard about CQRS pattern? Can you tell me what all is about?

CQRS Pattern Overview

- **Command Query Responsibility Segregation (CQRS)** is a design pattern that separates the read and write operations of an application's data model into separate models.
- **Core Principle:** Instead of using a single data model to handle both commands (write operations that change state) and queries (read operations that fetch data), CQRS advocates using separate models for these operations.
- **Key Concepts:**
 - **Commands:** Represent actions that change the state of the system (e.g., creating an entity, updating data).
 - **Queries:** Represent requests for information from the system (e.g., fetching data for display).
 - **Separate Models:** CQRS suggests having separate models to handle commands and queries, optimized for their respective tasks.

Example Use Cases

- **E-commerce:** Separate models for order processing (commands) and product listings (queries).
- **Banking:** Commands for transactions and queries for account balances.
- **Social Media:** Commands for posting content and queries for retrieving user feeds.

CQRS is particularly beneficial in scenarios where the write and read operations have distinct requirements for performance, scalability, or optimization, allowing developers to tailor solutions more precisely to these needs.

Q. Have you heard of zero-downtime deployment in MS? What are all the strategy you are following up in your current project?

Certainly! Here are deployment strategies for achieving zero-downtime deployment in microservices, each with a brief description:

- **Blue-Green Deployment:** Maintains two identical production environments (blue and green) to switch traffic seamlessly between old and new versions.
- **Canary Releases:** Deploys updates to a subset of users or servers to validate new versions before a full rollout.
- **Rolling Updates:** Deploys new versions in small increments across the production environment while monitoring and validating at each step.
- **Feature Toggles (Feature Flags):** Controls feature visibility through runtime configuration, allowing gradual rollout and quick rollback of new features.
- **Immutable Infrastructure:** Replaces entire instances or containers with new, immutable versions to ensure consistency and reliability during updates.

Q. What are all the scrum ceremony you followed in your current project?

Q. What is the branching strategy you are following in your project? Related to git/bitbucket

Branching Strategy Overview

- **Main Branches:**
 - **Master (or Main):** Represents the production-ready codebase. This branch is stable and should ideally reflect the latest deployed version in production.
 - **Develop (or Integration):** Acts as the integration branch where all completed features are merged before they are released. It serves as the base for testing and pre-production activities.
- **Supporting Branches:**
 - **Feature Branches:** Created from ``develop`` and used for developing new features or enhancements. Once complete, they are merged back into ``develop``.
 - **Release Branches:** Branched off ``develop`` when preparing for a release. Used for final testing, bug fixes, and versioning before merging into ``master`` for deployment.
 - **Hotfix Branches:** Emergency branches branched off ``master`` to fix critical bugs found in production. Once fixed, they are merged into both ``master`` and ``develop``.

Q. Which version of Junit you are using? Can you tell me how can you test an exception block?

Testing Exception Handling with JUnit 5

JUnit 5 provides assertions and annotations that make it straightforward to test methods that are expected to throw exceptions:

1. Using Assertions:

- JUnit 5 provides `assertThrows()` method from `org.junit.jupiter.api.Assertions` to verify that a specific exception is thrown during the execution of a piece of code.

```
java Copy code  
  
import static org.junit.jupiter.api.Assertions.*;  
  
@Test  
void testExceptionHandling() {  
    assertThrows(Exception.class, () -> {  
        // Code that should throw an exception  
        throw new Exception("This is an exception message");  
    });  
}
```

- In the example above:
 - `assertThrows()` expects the first argument to be the expected exception type (`Exception.class`).
 - The second argument is a lambda expression that contains the code that should throw the exception.

Q. What is this idempotency concept in REST?

Key Points of Idempotency in REST

- **Safe Repeats:** Idempotent operations can be repeated multiple times with the same effect as making the call once.
- **HTTP Methods:**
 - **GET:** Always idempotent, as it should only retrieve data without altering the server state.
 - **PUT:** Idempotent because multiple identical requests should update a resource to the same state.
 - **DELETE:** Idempotent since deleting a resource that does not exist should still result in the same state (resource not present).
 - **POST:** Generally not idempotent, as it often creates new resources and multiple identical requests can lead to multiple resources being created.
 - **PATCH:** Not guaranteed to be idempotent, as it can make partial updates which might have cumulative effects.

Q. What is composite primary key?

Q. How can we implement composite primary key in JPA?

Summary

- `@IdClass` is useful when you want to keep the primary key fields directly in the entity class.
- `@EmbeddedId` is more object-oriented, encapsulating the composite key into a separate class.

Q. How can we achieve multiple databases sources in MS?

Q. [SQL Question] Write SQL query in which you need to print customer ID, first and last name who has placed more than one order. [Customer is one entity | Order is one entity | foreign key: customer Id]

L1 – TechMojo

Q. What is your understanding of heap and stack in Java memory model?

Q. Why is try catch expensive? What it makes it expensive?

Q. Do you know what the daemon threads are?

Q. How to ensure that number of partition connected to a particular instance in Apache Kafka? How does load balancing will look like? What is the default behavior of Kafka? What type of relationship occurs between consumer in consumer group and partition of topic? One to one or many to many or one to many or many to one?

Q. How will we ensure that consumer group always get the data from only one partition? Ensure message ordering.

L1 – Victoria Secret

Q. Leetcode medium question – Rearrange Array elements by sign

Q. Design a system in which you are displaying top 20 product based on the clickbait the system received in the last 24 hours and data displayed should be recent. How many Micro-services will be there, DB, pattern you can use?

L1 – Epam

Q. Write program for binary search.

Q. Where you use Encapsulation in your project? How to achieve it?

Q. What classes in java uses Immutability? Can you write custom immutable class?

Q. How can we achieve Abstraction in java?

Q. Why we are calling it Runtime polymorphisms?

Q. Can you brief me about advantages of abstract class in java 8?

1. Code Reusability

Abstract classes allow you to provide a common base of functionality that can be shared among multiple derived classes. This is useful when you have common behavior that should be inherited by all subclasses.

2. Fields and Constructors

Abstract classes can have instance variables (fields) and constructors, which interfaces cannot have. This allows abstract classes to maintain state and initialize objects.

3. Multiple Methods with Implementation

While interfaces can have default methods with implementation, abstract classes allow multiple methods with full implementation, which can be more flexible and powerful in some cases.

4. Access Modifiers

Abstract classes can define methods with different access modifiers (public, protected, and private), while interface methods (default and static) must be public.

5. Extending Only One Class

Java supports single inheritance, meaning a class can extend only one other class. This makes abstract classes suitable for creating a well-defined inheritance hierarchy, ensuring that subclasses derive from a single abstract base class.

6. Tight Coupling

Abstract classes can provide tightly coupled functionality where subclasses need to share a lot of common code. This is useful when you want to ensure that subclasses inherit a certain structure and behavior.

7. Better for Evolving APIs

Abstract classes are better suited for evolving APIs, where you might want to add new methods in the future without breaking existing implementations. Adding new methods to an abstract class with default implementations won't affect subclasses that don't need to override them.

Q. Difference between Comparator and Comparable interface?

Q. what is the time complexity of HashSet?

Q. When will you use List and when will you use Set?

Q. What is the internal working of concurrent hashmap and how it helps to resolve the ConcurrentModificationException?

Q. What is the difference between Java 7 and Java 8 memory management?

Q. In Exceptional handling we are following one design pattern. Are you aware of it?

The Chain of Responsibility pattern is a powerful design pattern used in exception handling to decouple the main application logic from the logic for handling different types of exceptions. This pattern promotes cleaner code, separation of concerns, and scalability, making it easier to manage and extend exception handling logic.

Q. Difference between Callable and Runnable interface?

Q. Are you aware of the CompletableFuture Object?

Q. Can you explain the use case of flatMap in stream API?

Q. Can you name some functional interface which stream API uses?

L1 – Arise

Q. Internal working of HashMap? Why equals method used?

Q. Internal working of LinkedHashMap?

The `LinkedHashMap` maintains a doubly-linked list in addition to the `HashMap`'s structure, enabling it to provide predictable iteration order based on either insertion or access. This involves additional pointers in each entry to manage the list and ensures that operations like insertion, access, and removal update both the hash table and the linked list accordingly. This combination allows `LinkedHashMap` to offer both efficient key-based access and predictable iteration order.

Q. What is treeMap use for?

Q. Why are interfaces and abstract class useful?

Q. What is SOLID design principle says? What are the benefits of it?

Q. What are functional interfaces? What are the benefits?

Q. Can we use HashMap in multithreading environment? What benefits do concurrent hashmap provides? What are LinkedHashMap and TreeMap alternative in scenarios of multi-threading? What Collections.synchronizedMap() does internally?

Q. Instead of synchronized keyword, what we can use in order to achieve the same functionality?

L1 – Globant

Q. Where we can use LinkedList and where we can use ArrayList? Give me a practical scenario.

ArrayList

Use Case:

When you need fast random access to elements and your list is primarily used for read operations.

Example Scenario:

Suppose you are developing a contact management system where you frequently access contacts by their index to display them quickly.

LinkedList

Use Case:

When you have a list with frequent insertions and deletions, especially at the beginning or middle of the list.

Example Scenario:

Imagine you are implementing a playlist for a music player where songs can be added or removed frequently while maintaining their order.

Q. What are all the things we need to take of while creating HashMap with custom key and value? Can you please write the implementation of hashCode and equals?

Q. Which design pattern you have used? What is the scenario, why we went with Singleton? Spring boot provides Singleton object but is it truly singleton? Is it really one object created in the memory? Please write custom Singleton class code. Instead of synchronized block, is there any other mechanisms to let work singleton class in multi-threaded environment?

Instead of using a synchronized block, you can use other mechanisms to ensure thread safety in a Singleton class:

1. Bill Pugh Singleton Design:

This approach leverages the Java memory model's guarantees about class initialization. The inner static helper class is not loaded into memory until the `getInstance` method is called.

```
java Copy code

public class Singleton {
    private Singleton() {
    }

    private static class SingletonHelper {
        private static final Singleton INSTANCE = new Singleton();
    }

    public static Singleton getInstance() {
        return SingletonHelper.INSTANCE;
    }
}
```


2. Enum Singleton:

This is the simplest way to create a thread-safe Singleton. The JVM ensures that the enum value is instantiated only once.

```
java Copy code

public enum Singleton {
    INSTANCE;

    // Add your methods here
    public void someMethod() {
        // implementation here
    }
}
```

3. Static Block Initialization:

This approach initializes the Singleton instance in a static block. This is useful if the initialization might throw an exception.

```
java Copy code

public class Singleton {
    private static final Singleton INSTANCE;

    static {
        try {
            INSTANCE = new Singleton();
        } catch (Exception e) {
            throw new RuntimeException("Exception occurred in creating singleton instance");
        }
    }

    private Singleton() {
    }

    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```

Q. Why String is immutable in Java? Create custom immutable class. Which cloning is used and why?

- `String` Immutability: Ensures security, thread-safety, caching, and consistent hash codes.
- Custom Immutable Class: Should be `final`, fields should be `private` and `final`, and no setters or mutable fields.
- Cloning:
 - Shallow Cloning: Copies the structure but not referenced objects.
 - Deep Cloning: Creates a complete independent copy including all referenced objects.

Choosing between shallow and deep cloning depends on whether you need to clone only the object's structure or the entire object graph.

Q. Suppose you are using one micro-service which calls other micro-service and other micro-service got down, then what will happen? What happens to the requests which are coming in?

Q. What is the time complexity of `Arrays.sort()`? Which sorting has $O(n^2)$ time complexity?

- `Arrays.sort()`:
 - Primitive Arrays: $O(n \log n)$ using Dual-Pivot Quicksort.
 - Object Arrays: $O(n \log n)$ using TimSort.
- Sorting Algorithms with $O(n^2)$ Time Complexity:
 - Bubble Sort
 - Selection Sort
 - Insertion Sort

These $O(n^2)$ algorithms are generally less efficient than $O(n \log n)$ algorithms for larger datasets, making them suitable primarily for educational purposes or small arrays.

Q. Which design pattern you have worked on? Do you have any idea about facade design pattern?

- **Facade Design Pattern:** Provides a unified, simplified interface to a complex subsystem.
- **Purpose:** To make the subsystem easier to use, encapsulate complexity, and reduce coupling.
- **Structure:** Includes a Facade class and one or more subsystem classes.
- **Advantages:** Simplifies usage, reduces dependencies, and improves maintenance.
- **Disadvantages:** Overuse can lead to complexity and potential performance issues.

Q. Can you let me know what is Liskov substitution principle?

- **Liskov Substitution Principle (LSP):** Subtypes must be substitutable for their base types without altering the correctness of the program.
- **Principle:** Subclasses should extend the base class's behavior without changing its intended behavior or contracts.
- **Benefits:** Promotes code reusability, maintainability, and reduces errors by ensuring subclasses adhere to the expectations of their base classes.

Q. What are class-loader in Java?

Q. Difference between class level lock and method level lock?

Q. What is Heap data-structure?

- **Heap:** A complete binary tree satisfying the heap property (min-heap or max-heap).
- **Operations:** Includes insertion, deletion, and peek, with time complexities of $O(\log n)$ for insertion and deletion, and $O(1)$ for peek.
- **Use Cases:** Implementing priority queues, heapsort, and efficient algorithms for scheduling and resource management.

Q. Have you worked with Queue? How priority queues works internally?

- **Queue:** A linear data structure following FIFO order.
- **Priority Queue:** A queue where each element has a priority, and elements are served based on priority.
- **Internal Structure:** Typically implemented using a heap (min-heap or max-heap).
- **Heap Operations:** Insertion and removal involve adjusting the heap to maintain the heap property.

Priority queues are useful in scenarios where elements need to be processed based on their priority rather than their order of arrival, such as in scheduling tasks or managing resources.

Q. Difference between HashMap and LinkedHashMap? How to make linkedHashMap synchronized?

Q. Enhancement in interface from Java 8? What are access modifier we can use in default and static method?

- **Default Methods:** Provide method implementations in interfaces; can be `public` or package-private (default) but not `protected` or `private`.
- **Static Methods:** Methods that belong to the interface itself; can only be `public`.
- **Functional Interfaces:** Interfaces with a single abstract method, useful for lambda expressions and method references.

Q. What is reduce method of Stream API? Example.

Q. Is it possible to re-initiate a terminal stream in java 8? -No

Q. Have you worked on spring core?

Spring Core provides fundamental features like dependency injection, bean lifecycle management, configuration options, and aspect-oriented programming. It simplifies application development by managing object creation, dependencies, and cross-cutting concerns efficiently.

Q. Why REST APIs are called stateless?

Q. How did you integrate swagger in your application from scratch? Any annotation of swagger that you used?

1. Add Swagger dependencies to your project.
 2. Configure Swagger with a `SwaggerConfig` class.
 3. Use Swagger annotations to document your API.
 4. Access Swagger UI to view and test your API.
 5. Customize documentation as needed using `Docket` and `ApiInfo`.
- `@Api`: Describes a resource or a group of endpoints.
 - `@ApiOperation`: Describes an API operation or endpoint.
 - `@ApiParam`: Describes a parameter for an API operation.
 - `@ApiResponse`: Describes a response for an API operation.
 - `@ApiModel`: Describes a model used in API documentation.
 - `@ApiModelProperty`: Describes a property in a model.

Q. Can you differentiate PUT and PATCH operation? How do you apply PATCH operation? Which HTTPs methods are Idempotent?

Q. Any idea on DRY, YAGNI, KISS?

- DRY (Don't Repeat Yourself): Focus on reducing code duplication by abstracting common logic.
- YAGNI (You Aren't Gonna Need It): Avoid implementing features or complexity before they are actually required.
- KISS (Keep It Simple, Stupid): Aim for simplicity in design and implementation to make code more understandable and maintainable.

Q. Scenario based: How do you go ahead with authentication and authorization in Microservice architecture from scratch?

1. Define authentication and authorization requirements.
2. Choose and configure an Identity Provider.
3. Implement authentication and token validation in microservices.
4. Implement authorization using RBAC or ABAC.
5. Integrate with an API Gateway for centralized management.
6. Ensure secure communication and monitor authentication/authorization processes.
7. Test thoroughly for security and functionality.

Q. Design pattern of Microservices that you worked on? CQRS design pattern?

- **Microservices Design Pattern:** Involves service decomposition, API gateway, service discovery, load balancing, centralized logging, resilience, database per service, event-driven architecture, and security.
- **CQRS Design Pattern:** Separates commands and queries to optimize for read and write operations, often used with event sourcing for better scalability and performance.

Q. How inter-service communication happens in Microservices?

Q. How kafka server starts? Can you talk about partition in kafka?

- **Kafka Server Startup:** Involves configuring properties, starting Zookeeper, and then launching the Kafka broker which connects to Zookeeper, initializes log directories, and starts accepting client requests.
- **Partitions in Kafka:** Allow for parallel processing, scalability, and fault tolerance by splitting topic data across multiple brokers, with each partition managing a sequence of messages. Partitions are essential for distributing data, load balancing, and handling large volumes of messages efficiently.

Q. Can you differentiate a FUNCTION and stored PROCEDURE in SQL?

Summary

- **Functions:** Return values, can be used in SQL queries, and are typically used for calculations and transformations.
- **Stored Procedures:** Execute commands, manage transactions, and handle complex logic with potential side effects. They are not used directly in SQL expressions.

Q. Can you talk about Normalization in SQL?

Benefits of Normalization

1. **Reduces Data Redundancy:** Minimizes duplication of data, which can lead to inconsistencies.
2. **Improves Data Integrity:** Ensures that data is stored in a consistent and accurate manner.
3. **Enhances Query Efficiency:** Makes querying more efficient by organizing data logically.
4. **Simplifies Database Maintenance:** Makes it easier to update data without anomalies or redundant records.

Q. Do you have any idea about CASE expression in SQL?

- **`ELSE` Clause:** Optional. Specifies a default result if no conditions match.
- **Data Types:** All **`THEN`** and **`ELSE`** results must be of the same data type or implicitly convertible to the same data type.
- **Use Cases:** Useful for conditional logic in **`SELECT`** statements, **`ORDER BY`**, and **`UPDATE`** statements.

Q. Can you differentiate between where and having clause in SQL?

Aspect	`WHERE` Clause	`HAVING` Clause
Purpose	Filters rows before any groupings or aggregations.	Filters groups or aggregated results after grouping.
Usage	Used to specify conditions for individual rows.	Used to specify conditions for groups of rows.
Applicable To	Non-aggregated columns.	Aggregated columns (e.g., results of `COUNT`, `SUM`, etc.).
Execution Order	Applied before the `GROUP BY` clause.	Applied after the `GROUP BY` clause.
Can Be Used With	`SELECT`, `UPDATE`, `DELETE` statements.	`SELECT` statements with `GROUP BY`.
Example	<code>`SELECT * FROM employees WHERE salary > 50000;`</code>	<code>`SELECT department, COUNT(*) FROM employees GROUP BY department HAVING COUNT(*) > 10;`</code>
Supports	Comparisons, logical operators, and other conditions.	Aggregation functions (e.g., `COUNT`, `SUM`, `AVG`).

Q. Bean scope available in spring boot? What is happens if I inject prototype type scoped bean into singleton scoped bean?

- **Singleton Scope:** One instance per application context.
- **Prototype Scope:** New instance per request.
- **Request, Session, Application:** Specific to web applications and different contexts.

Injecting a prototype bean into a singleton bean may result in the prototype bean being shared across multiple uses unless you use mechanisms to explicitly retrieve new instances. Using ``ObjectFactory``, ``Provider``, or manually fetching the bean from the context can help manage this behavior.

2. Prototype Bean Injection:

- When a prototype-scoped bean is injected into a singleton-scoped bean, Spring will inject the same instance of the prototype bean into the singleton bean.
- The prototype bean will not be recreated with each request to the singleton bean. Instead, the singleton bean will hold a reference to a single instance of the prototype bean.

Q. How do you implement transaction management in spring boot from scratch?

1. **Add Dependencies:** Ensure you have the required Spring Boot and JPA dependencies.
2. **Configure DataSource:** Set up your database connection and JPA properties.
3. **Enable Transaction Management:** Use `@EnableTransactionManagement` if needed.
4. **Define Transactions:** Use `@Transactional` to manage transactions on service methods.
5. **Programmatic Transactions:** Use `TransactionTemplate` or `PlatformTransactionManager` if needed.
6. **Exception Handling:** Handle rollback behavior for exceptions.

Q. How to do custom exception in spring boot? What is the difference between `@RestControllerAdvice` and `@ControllerAdvice`?

Key Differences

Aspect	<code>@ControllerAdvice</code>	<code>@RestControllerAdvice</code>
Response Type	Can return views or JSON/XML responses	Returns JSON/XML responses by default
Used With	<code>@Controller</code>	<code>@RestController</code>
Default Behavior	Uses <code>@ResponseBody</code> to write response body	Combines <code>@ControllerAdvice</code> and <code>@ResponseBody</code>

- `@ControllerAdvice` is used for broader exception handling in MVC applications, and it can handle both JSON/XML responses and views.
- `@RestControllerAdvice` is tailored for RESTful APIs, automatically providing JSON/XML responses and is preferred for handling exceptions in RESTful services.

Q. Is it possible that I can provide my own configuration in spring boot? - Yes

Q. Can you differentiate between `@Qualifier` and `@Primary`?

Aspect	<code>@Qualifier</code>	<code>@Primary</code>
Purpose	Used to specify which bean to inject when multiple candidates are available.	Designates a bean as the default choice when multiple candidates are available.
Usage	Applied to fields or methods to resolve ambiguity between beans of the same type.	Applied to a bean definition to indicate it should be the default.
Typical Use	When you need to choose a specific bean by name or type.	When you want to set a default bean for injection.
Example	<code>@Qualifier("specificBeanName")</code>	<code>@Primary</code> on the bean class or method definition.
Conflict Resolution	Resolves conflicts by specifying a bean explicitly.	Resolves conflicts by marking one bean as the preferred choice.
Scope	Can be used with <code>@Autowired</code> , <code>@Inject</code> , etc., to disambiguate bean selection.	Used to set a default bean among multiple candidates.

Q. Which software methodology you have followed? – Agile? Can you mentioned some roles involved in Agile methodology?

- **Product Owner:** Defines and prioritizes the product backlog to maximize value.
- **Scrum Master:** Facilitates Scrum processes and removes impediments.
- **Development Team:** Creates and delivers product increments.
- **Stakeholders:** Provide feedback and set expectations.
- **Agile Coach:** Guides Agile practices and fosters continuous improvement.
- **Business Analyst:** Translates business needs into requirements.
- **UX/UI Designer:** Designs user interfaces and experiences.

Q. In software development, what are Non-functional requirements?

Non-functional requirements (NFRs) define the quality attributes, system performance, and constraints that a software system must meet. Unlike functional requirements, which specify what the system should do (e.g., user login functionality), non-functional requirements specify how the system should perform these functions.

