30 Java interview questions with answers

- **What is JVM?**

JVM stands for Java Virtual Machine.
It converts bytecode into machine code for execution.
It provides platform independence and memory management.

- **Difference between JDK, JRE, and JVM?**

JDK: Java Development Kit – includes tools and compiler.
JRE: Java Runtime Environment – runs Java apps, no compiler.
JVM: Part of JRE that executes bytecode.

- **What is OOP?**

Object-Oriented Programming is a model based on objects.
It includes principles: inheritance, encapsulation, abstraction, and polymorphism.
Java follows OOP principles.

- **What is inheritance?**

Inheritance allows one class to acquire properties of another.
It promotes code reuse.
`extends` keyword is used for inheritance.

- **What is encapsulation?**

Encapsulation means wrapping data and code in a single unit.
It uses private variables and public getters/setters.
It improves security and control.

- **What is abstraction?**

Abstraction hides implementation details from the user.
It is achieved using abstract classes and interfaces.
Only essential features are exposed.

- **What is polymorphism?**

Polymorphism means one name, many forms.
It allows method overloading and overriding.
It increases code flexibility.

- **What is the difference between method overloading and overriding?**

Overloading: same method name, different parameters, same class.
Overriding: same method name/signature in child class.
Overloading is compile-time; overriding is runtime.

- **What is a constructor?**

Constructor is used to initialize objects.
It has the same name as the class.
It doesn’t return anything.

- **What is the difference between ‘==’ and ‘.equals()’?**

== compares references (memory address).

`.equals()` compares values or content.
Use `.equals()` for string comparison.

- **What is an interface?**

Interface is a blueprint with abstract methods.
Used to achieve abstraction and multiple inheritance.
Implemented using the `implements` keyword.

- **What is the difference between abstract class and interface?**

Abstract class can have concrete methods; interface can’t (till Java 7).
A class can extend one abstract class but implement multiple interfaces.
Abstract class uses `extends`; interface uses `implements`.

- **What is final keyword?**

`final` is used to declare constants, prevent method overriding, or class inheritance.
Final variable = constant, final method = cannot override.
Final class = cannot be subclassed.

- **What is static keyword?**

`static` means belonging to the class, not instances.
Static variables and methods can be accessed without object.
Useful for utility and shared data.

- **What is this keyword?**

`this` refers to the current object instance.
Used to avoid naming conflicts in constructors/methods.
It can also invoke current class methods or constructors.

- **What is super keyword?**

`super` refers to the parent class object.
Used to access parent class methods, constructors, or variables.
It helps in method overriding.

- **What is exception handling?**

Exception handling deals with runtime errors.
It uses `try`, `catch`, `throw`, `throws`, and `finally`.
Prevents program crash and ensures smooth execution.

- **Difference between checked and unchecked exceptions?**

Checked: compile-time (e.g., IOException).
Unchecked: runtime (e.g., NullPointerException).
Checked must be handled using try-catch or throws.

- **What is the use of finally block?**

It is always executed after try-catch.
Used to close resources like files or DB connections.
Even runs after `return` statement.

- **What is a package?**

Package is a group of related classes and interfaces.
Used for organizing code and avoiding name conflicts.
Declared using `package` keyword.

- **What is the difference between Array and ArrayList?**

Array is fixed in size and type.
ArrayList is dynamic and part of `java.util` package.
ArrayList provides more flexibility.

- **What is garbage collection in Java?**

Garbage collector removes unused objects from memory.
It improves memory management automatically.
Runs in background using JVM.

- **What is the use of 'instanceof' keyword?**

It checks if an object is an instance of a class or subclass.
Returns true/false.
Used in type-checking.

- **What is multithreading?**

Multithreading allows multiple tasks to run concurrently.
Improves CPU utilization.
Implemented using `Thread` class or `Runnable` interface.

- **What is synchronization?**

Synchronization prevents thread interference.
It ensures one thread accesses a block at a time.
Used in multi-threaded environments.

- **What is String immutability?**

String in Java is immutable.
Once created, its value cannot be changed.
Modifying it creates a new object.

- **What is the difference between String, StringBuilder, and StringBuffer?**

String is immutable.
StringBuilder is mutable and not thread-safe.
StringBuffer is mutable and thread-safe.

- **What are wrapper classes?**

Wrapper classes convert primitive types into objects.
Examples: Integer, Double, Boolean.
Used in collections and object-based operations.

- **What is autoboxing and unboxing?**

Autoboxing: converting primitive to wrapper (int → Integer).
Unboxing: converting wrapper to primitive (Integer → int).
Happens automatically in Java.

Spring Boot Interview Questions & Answers

1. **What is Spring Boot?**
   Spring Boot is a framework for building stand-alone, production-ready Spring applications.
   It eliminates boilerplate configuration.
   It comes with embedded servers like Tomcat.
2. **What are the advantages of Spring Boot?**
   Less configuration, embedded server, faster development.
   Supports microservices architecture easily.
   Auto-configuration and production-ready features included.
3. **What is @SpringBootApplication?**
   It's a combination of `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`.
   Used to mark the main class of a Spring Boot app.
   It bootstraps the entire application.
4. **What is auto-configuration in Spring Boot?**
   Auto-configuration configures Spring beans automatically.
   It checks the classpath and creates beans based on dependencies.
   It reduces manual configuration.
5. **What is the starter dependency?**
   Starter is a set of convenient dependency descriptors.
   For example: `spring-boot-starter-web` for web apps.
   It helps to include related libraries quickly.
6. **What is application.properties?**
   It's a file to configure Spring Boot application settings.
   Used for DB configs, port numbers, logging, etc.
   Can also use `application.yml`.
7. **How to change default port in Spring Boot?**
   Set `server.port=8081` in `application.properties`.
   It changes the embedded server port.
   Default is 8080.
8. **What is the use of @RestController?**
   `@RestController` = `@Controller` + `@ResponseBody`.
   Used to create RESTful web services.
   Sends JSON or XML response.
9. **What is the difference between @Component, @Service, and @Repository?**
   All are stereotypes and create beans.
   `@Service` for business logic, `@Repository` for DAO, `@Component` is generic.
   Each has specific use cases and benefits.
10. **What is dependency injection?**
    It's a design pattern to inject objects instead of creating them.
    Spring uses constructor or setter injection.
    It promotes loose coupling.
11. **What is @Autowired?**
    Used for automatic dependency injection.
    It can be used on constructor, setter, or fields.
    Spring resolves and injects the bean.
12. **How to handle exceptions in Spring Boot?**
    Use `@ControllerAdvice` and `@ExceptionHandler`.

Helps in global exception handling.
Returns proper error response.

13. **What is actuator in Spring Boot?**
    Actuator provides production-ready features.
    It shows app health, metrics, beans, env, etc.
    Add `spring-boot-starter-actuator` dependency.
14. **What is Spring DevTools?**
    It helps in live reload and hot swapping.
    Improves development experience.
    Add `spring-boot-devtools` dependency.
15. **How to connect database in Spring Boot?**
    Use JDBC or JPA dependencies.
    Configure DB URL, username, password in `application.properties`.
    Spring Boot auto-configures DataSource.
16. **What is Spring Data JPA?**
    It simplifies data access using JPA.
    Provides `JpaRepository` and built-in query methods.
    Supports custom queries using JPQL.
17. **What is @Entity?**
    It maps a Java class to a database table.
    Used in JPA/Hibernate for ORM.
    Must have `@Id` as primary key.
18. **What is @Repository?**
    It marks a DAO class.
    It helps Spring handle persistence exceptions.
    Used with Spring Data JPA interfaces.
19. **What is @Transactional?**
    Used for transaction management.
    Ensures operations are executed in a transaction.
    Rolls back on failure.
20. **How to expose REST APIs in Spring Boot?**
    Use `@RestController` and `@RequestMapping`.
    Define methods with HTTP verbs like `@GetMapping`, `@PostMapping`.
    Return data as JSON.
21. **How to validate data in Spring Boot?**
    Use `@Valid` or `@Validated` annotations.
    Add validation annotations like `@NotNull`, `@Size`.
    Handle errors using `BindingResult`.
22. **How to enable CORS in Spring Boot?**
    Use `@CrossOrigin` on controller or method.
    Can also configure globally using `WebMvcConfigurer`.
    Helps in calling APIs from different domains.
23. **What is Spring Boot CLI?**
    Command-line tool for running and testing Spring Boot apps.
    Supports Groovy-based scripts.
    Helps in quick prototyping.
24. **What is CommandLineRunner?**
    Interface used to run code after Spring Boot starts.
    Override `run()` method.
    Used for DB seeding or initial setup.

25. **What is the use of @Value?**
    Injects values from `application.properties`.
    Used for configuration values.
    Example: `@Value("${server.port}")`.
26. **What is profile in Spring Boot?**
    Used for environment-specific configuration.
    Set using `spring.profiles.active=dev`.
    Supports `application-dev.properties`.
27. **What is logging default in Spring Boot?**
    Uses Spring Boot’s built-in logging with Logback.
    Logs are printed to console by default.
    Can customize in `application.properties`.
28. **What is YAML in Spring Boot?**
    YAML is a configuration format like `.properties`.
    More readable with indentation.
    File is named `application.yml`.
29. **How to schedule tasks in Spring Boot?**
    Use `@EnableScheduling` and `@Scheduled`.
    Used for periodic tasks like cron jobs.
    Example: `@Scheduled(fixedRate = 5000)`.
30. **What is the difference between Spring and Spring Boot?**
    Spring requires manual config, server setup.
    Spring Boot provides auto-config, embedded server.
    Spring Boot simplifies Spring app development.

Microservices Interview Questions & Answers

1. **What are Microservices?**
   Microservices are small, independent services.
   Each service performs a specific business function.
   They communicate via APIs (mostly REST).
2. **What are the benefits of Microservices?**
   Scalability, flexibility, independent deployment.
   Faster development and fault isolation.
   Technology agnostic (each service can use different tech).
3. **Difference between Monolithic and Microservices?**
   Monolithic = single large application.
   Microservices = multiple small services.
   Microservices are easier to scale and maintain.
4. **What is Service Discovery?**
   It helps services find each other dynamically.
   Used when service locations change frequently.
   Eureka is a popular service discovery tool.
5. **What is an API Gateway?**
   Single entry point for all microservices.
   Handles routing, security, and load balancing.
   Examples: Zuul, Spring Cloud Gateway.

6. **What is Eureka Server?**
   A service registry by Netflix for service discovery.
   Services register themselves to Eureka.
   Clients fetch registry to call other services.
7. **What is Feign Client?**
   A declarative REST client in Spring Cloud.
   Used to call other microservices easily.
   Reduces boilerplate REST client code.
8. **What is Circuit Breaker?**
   It prevents repeated failures by breaking the call.
   Used when a service is down.
   Resilience4j and Hystrix are common tools.
9. **What is Resilience4j?**
   A lightweight fault tolerance library.
   Supports circuit breaker, retry, rate limiter.
   Preferred over Hystrix now.
10. **How do services communicate in Microservices?**
    Via REST APIs (HTTP) or messaging (Kafka, RabbitMQ).
    Synchronous or asynchronous.
    JSON is commonly used for data exchange.
11. **What is Load Balancing?**
    Distributes traffic across multiple service instances.
    Improves reliability and performance.
    Can be client-side or server-side.
12. **What is Spring Cloud?**
    Provides tools for building cloud-native microservices.
    Includes Eureka, Config Server, Gateway, etc.
    Simplifies microservices setup with Spring Boot.
13. **What is Config Server?**
    Centralized configuration for microservices.
    Fetches properties from Git or file system.
    Supports environment-specific configs.
14. **What is centralized logging?**
    Collect logs from all services in one place.
    Helps in debugging and monitoring.
    ELK Stack (Elasticsearch, Logstash, Kibana) is commonly used.
15. **What is distributed tracing?**
    Tracks requests across multiple services.
    Helps find failures or bottlenecks.
    Tools: Zipkin, Sleuth, Jaeger.
16. **What is Sleuth?**
    Spring Cloud tool for tracing and logging.
    Adds trace ID and span ID to logs.
    Works with Zipkin for distributed tracing.
17. **What is Zipkin?**
    A distributed tracing system.
    Collects and visualizes trace data.
    Works with Sleuth to track request flow.
18. **What is Docker?**
    Tool to create, run, and manage containers.

Helps package applications with dependencies.
Used for consistent deployment.

19. **What is a Container?**
    Lightweight, isolated environment to run apps.
    Uses system resources efficiently.
    Docker is the most common container tool.
20. **What is Kubernetes?**
    An orchestration tool for containers.
    Manages deployment, scaling, and networking.
    Works well with Docker containers.
21. **What is Service Registry?**
    Stores info about all service instances.
    Used for service discovery.
    Example: Eureka Server.
22. **What is rate limiting?**
    Limits number of requests to a service.
    Prevents overloading and abuse.
    Implemented via Gateway or Resilience4j.
23. **What is OAuth2 in Microservices?**
    Authorization protocol for secure APIs.
    Used for token-based authentication.
    Ensures secure communication between services.
24. **What is JWT?**
    JSON Web Token used for authentication.
    Stores user data in a signed token.
    Sent with each API request.
25. **How do you secure Microservices?**
    Use JWT, OAuth2, HTTPS, and API Gateway.
    Secure communication and data exchange.
    Validate tokens on each request.
26. **What is a sidecar pattern?**
    Helper service runs alongside main service.
    Handles logging, monitoring, or proxying.
    Used in service meshes like Istio.
27. **What is Saga pattern?**
    Manages distributed transactions across services.
    Uses sequence of local transactions with compensation.
    Ensures data consistency.
28. **What is eventual consistency?**
    All services may not be updated instantly.
    Data becomes consistent over time.
    Used in distributed systems.
29. **What is a bounded context?**
    Defines clear boundaries around a microservice.
    Each context handles a specific domain area.
    Reduces tight coupling.
30. **How to test Microservices?**
    Use unit tests, integration tests, and contract tests.
    Mock dependencies and test APIs.
    Postman and JUnit are commonly used tools.