Most Asked Java , SpringBoot , JPA , Multithreading and Microservices Interview Questions 2025

1. What is JDK, JRE, and JVM?

JDK (Java Development Kit)

- JDK is a software development kit used to develop Java applications.
- It contains:
  - JRE
  - Compiler (javac)
  - Debugger
  - Development tools (javadoc, jdb, etc.)

📌 Used by developers

JRE (Java Runtime Environment)

- JRE provides the environment to run Java programs
- Contains:
  - JVM
  - Core Java libraries

📌 Used by users to run Java apps

JVM (Java Virtual Machine)

- JVM executes bytecode
- Responsible for:
    - Memory management
    - Garbage Collection
    - Security
    - Platform independence

Flow Example

.java → javac → .class (bytecode) → JVM → Output

🎯 Interview Tip:

JDK = development, JRE = runtime, JVM = execution engine

---

2. Explain Class Loader in Java with Example

What is Class Loader?

Class Loader is a JVM subsystem that loads .class files into memory dynamically at runtime.

Types of Class Loaders

1. Bootstrap ClassLoader
    - Loads core Java classes (String, Object)
2. Extension (Platform) ClassLoader
    - Loads classes from lib/ext
3. Application ClassLoader

    - Loads user-defined classes

Parent Delegation Model

- JVM asks parent class loader first
- Improves security and avoids duplicate loading

Example

```
public class Test {
   public static void main(String[] args) {
      System.out.println(Test.class.getClassLoader());
      System.out.println(String.class.getClassLoader());
   }
}
```

Output

```
AppClassLoader
null
```

🎯 Interview Tip:

Bootstrap ClassLoader returns null because it's native.

---

3. Explain System.out.println()

```
System.out.println("Hello");
```

Breakdown

- System → final class

- out → static PrintStream object
- println() → prints data + new line

Internal Flow

System → PrintStream → Console

🎯 Interview Tip:

println is a method of PrintStream, not System.

---

4. Explain public static void main(String[] args)

public

- JVM must access it from anywhere

static

- JVM does not create object to call main

void

- JVM does not expect return value

String[] args

- Command-line arguments

Example

java Test Hello

args[0] = "Hello"

🎯 Interview Tip:

main is static because JVM calls it without object creation.

---

5. Read Values Dynamically from Config Server Without Restart

Solution: Spring Cloud Config + @RefreshScope

How it Works

1. Config stored in Git
2. Config Server exposes it
3. Client fetches config
4. /actuator/refresh reloads values

Example

```
@RefreshScope
@RestController
public class TestController {
    @Value("${message}")
    private String message;
}
```

🎯 Interview Tip:

@RefreshScope enables runtime refresh without restart.

---

6. How to Fix Memory Leak Problems

Objects are no longer used but not garbage collected.

Causes

- Static references
- Unclosed resources
- Listener references

Fix

- Close resources
- Use weak references
- Analyze heap dump

Tools

- JVisualVM
- JProfiler
- MAT

🎯 Interview Tip:

Memory leaks cause OutOfMemoryError.

---

7. What is OOPS Concept?

Four Pillars

1. Encapsulation – data hiding
2. Inheritance – code reuse
3. Polymorphism – many forms
4. Abstraction – hide implementation

Example

```
class Car {
   private int speed;
}
```

🎯 Interview Tip:

OOPS improves maintainability and reusability.

---

8. Difference between throw and throws

throw	throws
Used inside method	Used in method signature
Throws one exception	Declares multiple exceptions
Used to create exception	Used to propagate

```
throw new IOException();
```

🎯 Interview Tip:

throw is action, throws is declaration.

---

9. Difference between final, finally, finalize

final	finally	finalize
Constant	Cleanup block	GC method

final	finally	finalize
Prevent inheritance	Always executes	Called by GC

🎯 Interview Tip:

finalize is deprecated in Java 9+

---

10. == vs equals()

- == → compares references
- equals() → compares content

```
new String("a") == new String("a") // false
```

---

11. Why String is Immutable

Reasons

- Security
- Caching
- Thread safety
- HashMap key stability

🎯 Interview Tip:

Immutability allows String pool.

---

12. How HashMap Works Internally

1. Hash key
2. Calculate index
3. Store in bucket
4. Collision → LinkedList / Tree

🎯 Interview Tip:

Java 8 uses Red-Black Tree after threshold.

---

13. HashMap vs ConcurrentHashMap

HashMap	ConcurrentHashMap
Not thread-safe	Thread-safe
Faster	Slightly slower
Allows null	No null

---

14. ArrayList vs LinkedList

ArrayList	LinkedList
Fast read	Fast insert/delete
Dynamic array	Doubly linked list

---

15. static keyword

Used for:

- Variables
- Methods
- Blocks
- Nested classes

🎯 Interview Tip:

Static belongs to class, not object.

---

16. Can we override static method?

❌ No

Static methods are compile-time binding, not runtime polymorphism.

---

17. Abstract class vs Interface

Abstract	Interface
Multiple methods	Pure abstraction
State allowed	No state

---

18. Marker Interface

- No methods
- Used for metadata

Examples:

- Serializable
- Cloneable

---

19. transient keyword

- Prevents serialization

```
transient int password;
```

---

20. Comparable vs Comparator

Comparable	Comparator
Natural order	Custom order
compareTo	compare

EXCEPTION HANDLING

---

21. Difference between Checked and Unchecked Exceptions

Checked Exceptions

- Checked at compile time
- Must be handled using try-catch or throws

Examples:

- IOException
- SQLException

Unchecked Exceptions

- Checked at runtime
- Extend RuntimeException

Examples:

- NullPointerException
- ArrayIndexOutOfBoundsException

🎯 Interview Tip:

Checked = recoverable, Unchecked = programming mistakes

---

22. What is Exception Propagation?

When an exception is not handled in a method, it is passed to the calling method.

Example

```
void m1() {
   int a = 10 / 0;
}
void m2() {
   m1();
```

}

Exception propagates from m1 → m2 → main.

🎯 Interview Tip:

Propagation works only for unchecked exceptions by default.

---

23. Can We Have Multiple Catch Blocks?

✅ Yes

Rules

- Order matters
- Child exception first
- Parent exception last

try {

} catch (ArithmeticException e) {

} catch (Exception e) {

}

🎯 Interview Tip:

Compiler error if parent catch comes before child.

---

24. What Happens if Exception Occurs in finally Block?

- Exception in finally overrides original exception

- Dangerous behavior

🎯 Interview Tip:

Avoid throwing exceptions in finally block.

---

25. Can Constructor Throw Exception?

✅ Yes

```
class Test {
    Test() throws IOException {
    }
}
```

🎯 Interview Tip:

Object creation fails if constructor throws exception.

---

26. How Do You Create a Custom Exception?

Steps

1. Extend Exception or RuntimeException
2. Create constructor

```
class BusinessException extends RuntimeException {
    BusinessException(String msg) {
        super(msg);
```

```
    }
}
```

🎯 Interview Tip:

Use RuntimeException for unchecked business errors.

---

27. What is try-with-resources?

Automatically closes resources.

```
try(FileReader fr = new FileReader("a.txt")) {
}
```

🎯 Interview Tip:

Introduced in Java 7.

---

28. Difference between Error and Exception

Error	Exception
JVM issue	Application issue
Not recoverable	Recoverable
Example: OOM	IOException

---

29. What is Rethrowing an Exception?

Catching and throwing again.

```
catch(Exception e) {
   throw e;
}
```

---

30. Best Practices for Exception Handling

- Never swallow exceptions
- Log properly
- Use custom exceptions
- Avoid generic Exception

🔹 MULTITHREADING & CONCURRENCY

---

31. What is Multithreading?

Multiple threads executing concurrently to improve performance.

🎯 Use cases

- Web servers
- Background tasks

---

32. Difference between Thread and Runnable

Thread	Runnable
Class	Interface
Single inheritance	Supports multiple inheritance

🎯 Interview Tip:

Runnable preferred.

---

33. What is Synchronization?

Ensures only one thread accesses shared resource at a time.

synchronized void print() {}

---

34. What is Deadlock? How to Avoid It?

Deadlock

Two threads waiting for each other's locks.

Avoid

- Lock ordering
- Avoid nested locks
- Use tryLock()

---

35. Difference between wait() and sleep()

wait()	sleep()
Releases lock	Holds lock
Object class	Thread class

---

36. What is volatile keyword?

- Ensures visibility across threads
- Prevents caching

🎯 Interview Tip:

volatile ≠ synchronization

---

37. synchronized Method vs Block

Method	Block
Whole method	Specific block
Less efficient	More control

---

38. What is Executor Framework?

Manages thread pool.

ExecutorService es = Executors.newFixedThreadPool(5);

🎯 Interview Tip:

Improves performance and resource management.

---

39. Callable vs Runnable

Callable	Runnable
Returns value	No return
Throws exception	Cannot

---

40. What is Thread-safe Collection?

Collections safe for concurrent access.

Examples:

- ConcurrentHashMap
- CopyOnWriteArrayList

🔹 MEMORY & PERFORMANCE

---

41. How Do You Identify Memory Leaks?

- Heap dump analysis
- Monitoring GC
- Profiling tools

---

42. Stack vs Heap Memory

Stack	Heap
Method calls	Objects
Fast	Slower
Thread-specific	Shared

---

43. Causes of OutOfMemoryError

- Memory leak
- Large objects
- Insufficient heap

---

44. What is Weak Reference?

- Garbage collected easily
- Used in caching

---

45. How to Analyze Heap Dump?

Tools:

- Eclipse MAT
- JVisualVM

---

46. What is Metaspace (Java 8)?

- Stores class metadata
- Replaced PermGen
- Grows dynamically

---

47. How Garbage Collector Works (Java 8+)

- Young Gen
- Old Gen
- Minor & Major GC

---

48. Minor GC vs Major GC

Minor	Major
Young gen	Old gen
Fast	Slow

- SPRING / SPRING BOOT

---

49. Spring vs Spring Boot

Spring	Spring Boot
Manual config	Auto config
External server	Embedded

---

50. What is Dependency Injection?

Objects provided by container instead of manual creation.

---

51. What is IoC Container?

Manages:

- Object creation
- Dependency injection
- Lifecycle

---

52. What is @SpringBootApplication?

Combination of:

- @Configuration
- @EnableAutoConfiguration
- @ComponentScan

---

53. @Component vs @Service vs @Repository

Same purpose, different layers.

🎯 Interview Tip:

@Repository handles DB exceptions.

---

54. Spring Boot Auto-Configuration

Uses:

- classpath
- conditions
- starters

---

55. Global Exception Handling

Using @ControllerAdvice.

---

56. @ControllerAdvice

Centralized exception handling.

---

57. @Controller vs @RestController

Controller	RestController
View	JSON
@ResponseBody needed	Auto

---

58. @Autowired – How It Works Internally

- Uses reflection
- By type → qualifier → name

---

59. What is @Transactional?

Manages DB transactions.

🔹 MICROSERVICES

---

60. Microservices Architecture

Small independent services communicating via REST.

---

61. Monolithic vs Microservices

Monolith	Microservices
Single deploy	Independent deploy

---

62. What is Config Server?

Centralized configuration management.

---

63. Read Config Without Restart ⭐

- @RefreshScope
- /actuator/refresh

---

64. What is @RefreshScope?

Reloads bean at runtime.

---

65. Eureka Server

Service registry and discovery.

---

66. API Gateway

Single entry point.

---

67. Circuit Breaker

Stops cascading failures.

---

68. Resilience4j

Fault tolerance library.

---

69. Feign Client

Declarative REST client.

---

70. Saga Pattern

Manages distributed transactions.

- DATABASE & JPA

---

71. What is JPA?

Specification for ORM.

---

72. JPA vs Hibernate

Hibernate implements JPA.

---

73. Lazy vs Eager Loading

Lazy loads on demand.

---

74. N+1 Problem

Multiple unnecessary DB calls.

---

75. Transactions in Spring Boot

Using @Transactional.

---

76. Optimistic vs Pessimistic Locking

Optimistic	Pessimistic
Version based	Lock based

🔥 BONUS – REAL INTERVIEW QUESTIONS

---

77. How Do You Fix Production Issues?

- Logs
- Metrics
- Rollback
- Hotfix

---

78. Handle High Traffic

- Load balancer
- Cache
- Scaling
- Async

---

79. Exception Handling for REST APIs

- Global handler
- Proper HTTP codes

---

80. Securing Microservices

- OAuth2

- JWT
- API Gateway

---

81. Distributed Transactions

- Saga
- Event-driven