Top 50 'Difference Between' Interview Questions for Java Developers

1. JDK vs JRE vs JVM

JDK: Development kit including compiler.

JRE: Runtime environment to execute Java programs.

JVM: Machine that runs Java bytecode.

2. Abstract Class vs Interface

Abstract Class: Can have both abstract and concrete methods.

Interface: Only abstract methods (before Java 8).

3. ArrayList vs LinkedList

ArrayList: Fast random access, slow insert/delete.

LinkedList: Slow random access, fast insert/delete.

4. HashMap vs Hashtable

HashMap: Not synchronized, allows one null key.

Hashtable: Synchronized, does not allow null keys.

5. == vs .equals()

==: Compares references (memory location).

.equals(): Compares values (can be overridden).

6. String vs StringBuilder vs StringBuffer

String: Immutable.

StringBuffer: Mutable, synchronized.

StringBuilder: Mutable, not synchronized.

7. Overloading vs Overriding

Overloading: Same method, different parameters.

Overriding: Same method, same parameters, different class.

8. Static vs Final

Static: Belongs to class, shared among objects.

Final: Prevents modification of class, method, or variable.

9. Checked vs Unchecked Exceptions

Checked: Must be handled (IOException, SQLException).

Unchecked: Runtime errors (NullPointerException).

10. Process vs Thread

Process: Independent execution, separate memory.

Thread: Lightweight, shared memory within a process.

11. Deep Copy vs Shallow Copy

Shallow Copy: Copies references, changes reflect in both objects.

Deep Copy: Creates independent copies of objects.

12. Heap Memory vs Stack Memory

Heap: Stores objects, shared among threads.

Stack: Stores method calls, local variables.

13. Synchronized vs Concurrent Collections

Synchronized: Locks whole collection.

Concurrent: Uses java.util.concurrent for better performance.

14. Volatile vs Synchronized

Volatile: Ensures visibility across threads.

Synchronized: Provides atomicity by locking sections.

15. Callable vs Runnable

Runnable: No return value.

Callable: Returns a result (Future<T>), supports exceptions.

16. HashSet vs TreeSet

HashSet: Unordered, uses hashing (O(1) operations).

TreeSet: Ordered, uses Red-Black Tree (O(log n) operations).

17. Comparable vs Comparator

Comparable: Defines natural ordering inside class.

Comparator: Custom ordering outside class.

18. Serialization vs Deserialization

Serialization: Converts object to byte stream.

Deserialization: Converts byte stream back to object.

19. ForkJoinPool vs ThreadPoolExecutor

ForkJoinPool: Uses work-stealing for parallelism.

ThreadPoolExecutor: Fixed number of worker threads.

20. Mutable vs Immutable Objects

Immutable: Cannot be modified after creation.

Mutable: Can be changed after creation.

21. Spring Boot vs Spring MVC

Spring Boot: Standalone, microservices-friendly.

Spring MVC: Web framework for Java apps.

22. REST API vs SOAP

REST: Lightweight, JSON/XML, stateless.

SOAP: XML-based, more secure, used in enterprise apps.

23. SQL vs NoSQL

SQL: Structured, relational database.

NoSQL: Unstructured, schema-less, scalable.

24. Microservices vs Monolithic Architecture

Microservices: Independent services, scalable.

Monolithic: Single large application, harder to scale.

25. Lambda Expressions vs Anonymous Classes

Lambda: Shorter syntax, introduced in Java 8.

Anonymous Class: More verbose, used before Java 8.

26. RESTful Web Services vs GraphQL

REST: Predefined endpoints, fixed structure.

GraphQL: Flexible queries, client-driven API.

27. GET vs POST in HTTP

GET: Retrieves data, can be cached.

POST: Sends data, not cached, used for sensitive data.

28. String Pool vs Heap Memory

String Pool: Stores unique string literals.

Heap Memory: Stores all objects including strings.

29. Checked Exception vs Runtime Exception

Checked: Requires handling.

Runtime: Unchecked, occurs at runtime.

30. Primitive vs Wrapper Classes

Primitive: int, double, etc. No object features.

Wrapper: Integer, Double, etc. Provides additional methods.

31. Soft Reference vs Weak Reference

Soft: Garbage collected when memory is low.

Weak: Collected more aggressively by GC.

32. Iterator vs ListIterator

Iterator: Can only traverse forward.

ListIterator: Can traverse both directions.

33. BlockingQueue vs ConcurrentLinkedQueue

BlockingQueue: Thread-safe with waiting mechanisms.

ConcurrentLinkedQueue: Non-blocking queue with better performance.

34. HashMap vs ConcurrentHashMap

HashMap: Not thread-safe.

ConcurrentHashMap: Thread-safe, better concurrency.

35. Stream API vs Collections API

Stream API: Functional operations on data.

Collections API: Traditional data structure manipulations.

36. Garbage Collection vs Manual Memory Management

GC: Automated memory cleanup.

Manual: Programmer manages memory (C, C++).

37. WeakHashMap vs HashMap

WeakHashMap: Uses weak references, entries removed when no strong reference.

HashMap: Uses strong references, entries persist.

38. Static Binding vs Dynamic Binding

Static: Resolved at compile-time.

Dynamic: Resolved at runtime (method overriding).

39. ClassLoader vs Class.forName()

ClassLoader: Loads classes at runtime.

Class.forName(): Loads class dynamically by name.

40. Bitwise Operators vs Logical Operators

Bitwise: Operates at bit level.

Logical: Operates on boolean values.

41. Predicate vs Function in Java Streams

Predicate: Returns boolean.

Function: Returns transformed value.

42. TreeMap vs HashMap

TreeMap: Ordered (Red-Black Tree).

HashMap: Unordered (Hashing).

43. Deadlock vs Starvation in Threading

Deadlock: Two threads waiting on each other.

Starvation: A thread never gets CPU time.

44. IdentityHashMap vs HashMap

IdentityHashMap: Uses reference equality.

HashMap: Uses equals() for key comparison.

45. Singleton vs Prototype in Spring

Singleton: One instance per Spring container.

Prototype: New instance every time requested.

46. Fail-fast vs Fail-safe Iterators

Fail-fast: Throws ConcurrentModificationException.

Fail-safe: Works on a copy, avoids exceptions.

47. Public vs Private Access Modifier

Public: Accessible from anywhere.

Private: Accessible only within the same class.

48. Synchronous vs Asynchronous Processing

Synchronous: Tasks execute sequentially.

Asynchronous: Tasks run concurrently, improving performance.

49. Path vs Classpath in Java

Path: Defines where system binaries are located.

Classpath: Defines where Java classes and libraries are located.

50. Iterator vs Spliterator

Iterator: Used for sequential traversal.

Spliterator: Supports parallel processing with better performance.