# Top 100 Java Interview Questions

## Core Java (Basics & OOPs)

1. **What are the main features of Java?**
   Java is object-oriented, platform-independent, secure, robust, multithreaded, and supports automatic memory management (garbage collection).

2. **What is the difference between JDK, JRE, and JVM?**
   - **JVM**: Java Virtual Machine runs Java bytecode.
   - **JRE**: Java Runtime Environment contains JVM + libraries.
   - **JDK**: Java Development Kit contains JRE + development tools.

3. **What are access modifiers in Java?**
   They define access scope: `public`, `protected`, `default`, and `private`.

4. **What is the difference between `==` and `.equals()`?**
   `==` compares object references, `.equals()` compares content.

5. **What is a constructor?**
   A constructor initializes an object. It has no return type and matches the class name.

6. **Can a constructor be private?**
   Yes, to restrict instantiation (e.g., Singleton pattern).

7. **What is abstraction and encapsulation?**
   - **Abstraction**: Hides internal details and shows only functionality.
   - **Encapsulation**: Binds data and methods into a single unit.

8. **Explain inheritance.**
   One class inherits another's properties/methods using `extends`.

9. **What is method overloading?**
   Multiple methods with the same name but different parameters.

10. **What is method overriding?**
    A subclass provides a specific implementation of a method from the parent class.

---

## Data Types & Control Flow

11. **What are primitive data types?**
    byte, short, int, long, float, double, boolean, char.

12. **What is type casting?**
    Converting one data type into another. Two types: implicit and explicit.

13. **Difference between static and non-static variables?**
    Static belongs to class, non-static to instance.

14. **Difference: final, finally, finalize()?**
    - `final`: constant or non-overridable.
    - `finally`: block executed after try-catch.
    - `finalize()`: method before object is garbage collected.

15. **Use of `this` and `super`?**
    - `this`: refers to current class object.
    - `super`: refers to parent class.

---

## Strings

16. **Difference between String, StringBuffer, StringBuilder?**
    - String: immutable
    - StringBuffer: mutable and thread-safe
    - StringBuilder: mutable but not thread-safe

17. **Why are Strings immutable?**
    For security, synchronization, and caching.

18. **How does `intern()` work?**
    Returns canonical representation from string pool.

19. **equals() vs hashCode()?**
    `equals()` checks equality, `hashCode()` provides hash value for hashing structures.

20. **"abc" + 1 + 2 vs 1 + 2 + "abc"?**
    - "abc" + 1 + 2 → "abc12"
    - 1 + 2 + "abc" → "3abc"

---

## Collections

21. **Difference between Array and ArrayList?**
    Array is fixed-size, ArrayList is dynamic.

22. **What is the Collection framework?**
    A set of classes and interfaces for storing and manipulating groups of data.

23. **List vs Set vs Map?**
    - List: Ordered, allows duplicates.
    - Set: No duplicates.
    - Map: Key-value pairs.

24. **ArrayList vs LinkedList?**
    ArrayList is faster in access; LinkedList is better for insertion/deletion.

25. **HashSet vs TreeSet?**
    HashSet is unordered, TreeSet is sorted.

26. **How does HashMap work internally?**
    Uses `hashCode()` and `equals()` to store key-value pairs in buckets.

27. **Can HashMap store null keys?**
    Yes, one null key allowed.

28. **HashMap vs Hashtable?**
    HashMap is non-synchronized; Hashtable is synchronized.

29. **What is ConcurrentHashMap?**
    Thread-safe version of HashMap without locking entire map.

30. **Fail-fast vs fail-safe?**

    - Fail-fast throws `ConcurrentModificationException`.
    - Fail-safe does not (e.g., `CopyOnWriteArrayList`).

---

## Exception Handling

31. **Difference between checked and unchecked exceptions?**

    - Checked: Compile-time exceptions (e.g., IOException).
    - Unchecked: Runtime exceptions (e.g., NullPointerException).

32. **Explain try-catch-finally with example.**
    try-catch handles exceptions; finally always executes.

33. **Can we have multiple catch blocks?**
    Yes, to handle different exception types.

34. **Difference between throw and throws?**

    - `throw`: Used to throw an exception.
    - `throws`: Declares an exception in method signature.

35. **What happens if there's a return in try and finally block?**
    Finally block overrides the return in try block.

---

## Multithreading & Concurrency

36. **What is multithreading?**
    Ability of CPU to execute multiple threads simultaneously.

37. **Difference between process and thread?**

    - Process: Independent program execution.
    - Thread: Lightweight process within a process.

38. **How to create a thread in Java?**

    - Extending `Thread` class
    - Implementing `Runnable` interface

39. **Runnable vs Thread class?**
    Runnable is preferred as Java supports multiple interface inheritance.

40. **What is synchronization?**
    Prevents thread interference by locking shared resources.

41. **Difference between wait() and sleep()?**

    - wait(): Releases lock, used in synchronized context.
    - sleep(): Pauses thread, doesn't release lock.

42. **What is a deadlock?**
    Two or more threads waiting on each other indefinitely for resources.

43. **How to prevent deadlock?**
    Avoid nested locks, use lock ordering, timeout strategies.

44. **Synchronized method vs synchronized block?**

    - Method locks entire method
    - Block can lock only specific code/objects

45. **What is volatile keyword?**
    Ensures visibility of variable changes across threads.

---

## Java 8 Features

46. **What are lambda expressions?**
    Anonymous functions used to implement functional interfaces.

47. **What is a functional interface?**
    Interface with a single abstract method.

48. **What is Stream API?**
    Enables functional-style operations on data collections.

49. **Difference between map() and flatMap()?**
    - map(): transforms elements
    - flatMap(): flattens nested structures

50. **Default and static methods in interface?**
    Interfaces can have default and static methods from Java 8.

---

## Interfaces & Abstract Classes

51. **Difference between abstract class and interface?**
    Abstract classes can have state and constructors; interfaces can't (before Java 8).

52. **Can we create an object of an interface?**
    No.

53. **Can interfaces have constructors?**
    No.

54. **What happens if a class does not implement all interface methods?**
    It must be declared abstract.

55. **Can a class implement multiple interfaces?**
    Yes, Java allows multiple interface inheritance.

---

## Garbage Collection

56. **How does garbage collection work in Java?**
    JVM automatically deletes unreachable objects.

57. **Types of references in Java?**
    Strong, Soft, Weak, Phantom.

58. **Difference between System.gc() and Runtime.gc()?**
    Both request garbage collection, but JVM may ignore.

59. **What is a memory leak in Java?**
    Unused objects remain referenced and not garbage collected.

60. **Can we force garbage collection in Java?**
    No, we can only request it.

## Serialization & Deserialization

61. **What is serialization in Java?**
    Converting an object into a byte stream for storage or transmission.
62. **How to make a Java class serializable?**
    Implement the `Serializable` interface.
63. **What is `serialVersionUID`?**
    It's used to verify compatibility between sender and receiver during deserialization.
64. **What happens if a class has a non-serializable field?**
    Mark it as `transient` to skip during serialization.
65. **Difference between Serializable and Externalizable?**
    `Serializable` is a marker interface; `Externalizable` gives full control over serialization via `readExternal()` and `writeExternal()`.

---

## Inner Classes

66. **What is an inner class?**
    A class defined within another class.
67. **What are anonymous inner classes?**
    Classes without a name used for instantiating classes with certain methods, usually in a single statement.
68. **Can static classes be inner classes?**
    Yes, called static nested classes.

69. **Difference between static and non-static inner classes?**
    Static inner classes can't access outer class members directly; non-static can.
70. **Use cases for inner classes?**
    Event handling, grouping classes logically, increasing encapsulation.

---

## Java Memory Model & Performance

71. **What is the memory structure of JVM?**
    JVM memory is divided into Method Area, Heap, Stack, PC Registers, and Native Method Stack.
72. **What is PermGen and Metaspace?**
    PermGen (Java 7 and earlier) stored metadata; replaced by Metaspace in Java 8.
73. **What is heap and stack memory?**
    - Heap: Used for dynamic memory allocation.
    - Stack: Stores method calls and local variables.
74. **How to avoid memory leaks?**
    Remove unused object references, close resources, use profilers.
75. **Tools to analyze memory usage in Java?**
    VisualVM, JConsole, JProfiler, Eclipse MAT.

---

## Keywords & Modifiers

76. **What is the use of `transient` keyword?**
    Used to indicate that a field should not be serialized.
77. **Difference between static block and constructor?**
    Static block runs once when class is loaded; constructor runs on object creation.
78. **Can we override static methods?**
    No, static methods are hidden, not overridden.
79. **Can main() be overloaded?**
    Yes, but JVM always calls the standard `main(String[] args)`.
80. **Why is the main method static?**
    So JVM can invoke it without creating an object.

---

## Design Principles & Patterns

81. **What is SOLID in Java?**
    A set of design principles: Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.

82. **Explain the Singleton Design Pattern.**
    Ensures only one instance of a class exists using a private constructor and a static method.
83. **What is Factory Design Pattern?**
    Creates objects without exposing instantiation logic to the client.
84. **What is Observer Design Pattern?**
    Objects (observers) subscribe to events from another object (subject) and get notified when state changes.
85. **Best practices for Java programming?**
    Follow naming conventions, write clean code, use design patterns, handle exceptions properly, write unit tests.

---

## Coding Challenges (Popular)

86. **Reverse a string without using built-in methods.**
    Use a loop or `StringBuilder` with reverse logic.
87. **Check if a number is a palindrome.**
    Convert to string, reverse it, and compare.
88. **Find duplicate elements in an array.**
    Use HashSet or nested loops.
89. **Implement Fibonacci series using recursion.**
    `if(n<=1) return n; else return fib(n-1)+fib(n-2);`
90. **Find the first non-repeated character in a string.**
    Use LinkedHashMap to count frequencies.
91. **Sort an array without using sort().**
    Implement bubble sort, selection sort, or insertion sort.
92. **Check if two strings are anagrams.**
    Sort both strings and compare or use character counts.
93. **Custom implementation of HashMap?**
    Create an array of LinkedLists and use hashCode() to place entries.
94. **Producer-consumer problem using threads.**
    Use wait(), notify(), synchronized block or BlockingQueue.
95. **Count frequency of each word in a sentence.**
    Split string, use HashMap for counting.

---

## Advanced & Tricky Questions

96. **Compile-time vs runtime polymorphism?**
    - Compile-time: method overloading
    - Runtime: method overriding
97. **Can a class be both abstract and final?**
    No, abstract needs subclassing; final prevents it.
98. **What is method hiding?**
    Static method in subclass with same signature as in superclass.
99. **How does autoboxing/unboxing work?**
    Automatic conversion between primitives and wrapper classes.
100. **What is the diamond problem and how does Java handle it?**
    Occurs with multiple inheritance; Java avoids it by allowing multiple interfaces but not classes. Interfaces don't carry state, so no ambiguity.