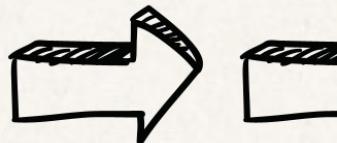
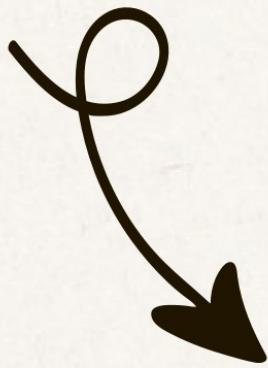


# **Ace your Java Interviews**

## **50 Must-Know Questions**

### **(0-3 Years)**

**Save this Post for Future Use**



# Java Interview Questions

## 1. What is Java and what are its key features?

Java is a platform-independent, object-oriented programming language.

Key features include:

- **Platform Independence:** "Write Once, Run Anywhere" through JVM
- **Object-Oriented:** Supports encapsulation, inheritance, abstraction and polymorphism
- **Memory Management:** Automatic garbage collection
- **Multithreading:** Built-in support for concurrent programming
- **Robustness:** Strong type checking and exception handling

## 2. Explain the difference between JDK, JRE, and JVM.

- **JVM (Java Virtual Machine):** Runtime environment that executes Java bytecode
- **JRE (Java Runtime Environment):** JVM + libraries needed to run Java applications
- **JDK (Java Development Kit):** JRE + development tools (compiler, debugger) needed to develop Java applications

## 3. What is the difference between primitive and reference data types?

- **Primitive types:** Store actual values directly in memory (int, char, boolean, double, etc.). Passed by value.
- **Reference types:** Store memory addresses pointing to objects (String, Arrays, Objects). Passed by reference to the object.

## 4. Explain method overloading vs method overriding.

- **Overloading:** Same method name with different parameters in the same class. Resolved at compiletime.
- **Overriding:** Subclass provides specific implementation of a method already defined in parent class. Resolved at runtime using dynamic method dispatch.

# Java Interview Questions

## 5. What is the difference between String, StringBuilder, and StringBuffer?

- **String:** Immutable, threadsafe. Creates new object for each modification.
- **StringBuilder:** Mutable, not threadsafe. Better performance for singlethreaded operations.
- **StringBuffer:** Mutable, threadsafe (synchronized methods). Slower than StringBuilder due to synchronization overhead.

## 6. Explain the four pillars of OOP.

- **Encapsulation:** Bundling data and methods together, hiding internal implementation
- **Inheritance:** Acquiring properties and behaviors from parent class
- **Polymorphism:** Same interface with different implementations (method overloading/overriding)
- **Abstraction:** Hiding complex implementation details, showing only essential features

## 7.What is the difference between abstract class and interface?

- **Abstract Class:** Can have both abstract and concrete methods, instance variables, constructors. Single inheritance only.
- **Interface:** Only abstract methods (until Java 8), constants only, no constructors. Supports multiple inheritance.

## 8.Explain access modifiers in Java.

- **private:** Accessible only within the same class
- **default (packageprivate):** Accessible within the same package
- **protected:** Accessible within package and subclasses
- **public:** Accessible from anywhere

Dheenadayalan Dhanapal

---

# Java Interview Questions

## 9. What is a constructor and what are its types?

Constructor initializes objects when created. Types:

- **Default Constructor:** No parameters, provided by compiler if none defined
- **Parameterized Constructor:** Takes parameters to initialize object with specific values
- **Copy Constructor:** Not directly supported in Java, but can be implemented manually

## 10. What is the super keyword used for?

- Access parent class variables: **super.variable**
- Call parent class methods: **super.method()**
- Call parent class constructor: **super()** (must be first statement)

## 11. Explain Java memory model (Heap vs Stack).

- **Stack Memory:** Stores method calls, local variables, partial results. Threadspecific, automatic cleanup.
- **Heap Memory:** Stores objects and instance variables. Shared among threads, managed by garbage collector.

## 12. What are the different types of references in Java?

- **Strong Reference:** Default reference, prevents garbage collection
- **Weak Reference:** Allows garbage collection when only weak references exist
- **Soft Reference:** Collected when memory is low
- **Phantom Reference:** Used for cleanup actions before object is garbage collected

# Java Interview Questions

## 13. What is garbage collection and how does it work?

Automatic memory management that reclaims memory used by objects no longer referenced. Process:

- Mark unreferenced objects
- Sweep (delete) marked objects
- Compact remaining objects to reduce fragmentation

## 14. Explain the exception hierarchy in Java.

1) Throwable (toplevel)

- 1.1) Exception (checked exceptions that must be handled)
  - 1.1.1) IOException, SQLException, ClassNotFoundException
- 1.2) RuntimeException (unchecked exceptions)
  - 1.2.1) NullPointerException, ArrayIndexOutOfBoundsException
- 1.3) Error (systemlevel errors)
  - 1.3.1) OutOfMemoryError, StackOverflowError

## 15. What is the difference between checked and unchecked exceptions?

- **Checked Exceptions:** Must be handled at compiletime using trycatch or throws declaration
- **Unchecked Exceptions:** Runtime exceptions that don't require explicit handling

## 16. Explain try, catch, finally block.

**try:** Contains code that might throw an exception

**catch:** Handles specific exceptions

**finally:** Always executes regardless of exception occurrence (used for cleanup)

Dheenadayalan Dhanapal

# Java Interview Questions

## 17. What is the difference between throw and throws?

- **throw:** Used to explicitly throw an exception within a method
- **throws:** Used in method signature to declare that method might throw specific exceptions

## 18. What is the Collections Framework?

Unified architecture for storing and manipulating groups of objects.

Includes:

- **Interfaces:** List, Set, Map, Queue
- **Implementations:** ArrayList, HashMap, HashSet, etc.
- **Algorithms:** Sorting, searching utilities

## 19. Explain the difference between List, Set, and Map.

- **List:** Ordered collection allowing duplicates (ArrayList, LinkedList)
- **Set:** Unordered collection with unique elements (HashSet, TreeSet)
- **Map:** Keyvalue pairs with unique keys (HashMap, TreeMap)

## 20. What is the difference between ArrayList and LinkedList?

- **ArrayList:** Dynamic array, fast random access  $O(1)$ , slow insertion/deletion in middle  $O(n)$
- **LinkedList:** Doubly linked list, slow random access  $O(n)$ , fast insertion/deletion  $O(1)$

## 21. What is the difference between HashMap and TreeMap?

- **HashMap:** Hash table based,  $O(1)$  average access, no ordering
- **TreeMap:** Redblack tree based,  $O(\log n)$  access, maintains sorted order

# Java Interview Questions

## 22. Explain HashMap internal working.

HashMap uses array of buckets with hash function to determine index. Process:

1. Calculate hash code of key
2. Apply hash function to find bucket index
3. Handle collisions using chaining (linked list/tree structure)
4. Resize when load factor exceeds threshold (0.75)

## 23. What is the difference between Iterator and ListIterator?

- **Iterator:** Forward only traversal, works with all collections
- **List Iterator:** Bidirectional traversal, works only with List, allows modification during iteration

## 24. What are Lambda expressions?

Anonymous functions that enable functional programming.

**Syntax:** (parameters) > expression

### // Before Java 8

```
Comparator<String> comp = new Comparator<String>() {  
    public int compare(String a, String b) { return a.compareTo(b); }  
};
```

### // Java 8 Lambda

```
Comparator<String> comp = (a, b) > a.compareTo(b);
```

## 25. What is the Stream API?

Functional approach to process collections declaratively. Supports:

- **Intermediate operations:** filter, map, sorted, etc.. (return Stream)
- **Terminal operations:** collect, forEach, reduce, etc.. (return result)

Dheenadayalan Dhanapal

---

# Java Interview Questions

## 26. Explain Optional class.

Container that may or may not contain a value, helps avoid NullPointerException.

### Sample Code:

```
Optional<String> optional = Optional.ofNullable(getString());  
optional.ifPresent(System.out::println);  
String result = optional.orElse("default");
```

## 27. What are functional interfaces?

Interfaces with exactly one abstract method. Can be used with lambda expressions:

- **Predicate<T>**: Boolean test
- **Function<T,R>**: Transform input to output
- **Consumer<T>**: Accept input, no return
- **Supplier<T>**: Supply value, no input

## 28. What are method references?

Shorthand for lambda expressions that call existing methods:

### // Lambda

```
list.forEach(x > System.out.println(x));
```

### // Method reference

```
list.forEach(System.out::println);
```

## 29. How do you create threads in Java?

Two ways:

1. **Extend Thread class:** Override run() method

2. **Implement Runnable interface:** Implement run() method, pass to Thread constructor

Dheenadayalan Dhanapal

---

# Java Interview Questions

## 30. What is multithreading and its benefits?

Concurrent execution of multiple threads within a program. Benefits:

- Better resource utilization
- Improved performance through parallelism
- Better user experience (responsive UI)
- Better system throughput

## 31. Explain thread lifecycle states.

- **NEW:** Thread created but not started
- **RUNNABLE:** Thread executing or ready to execute
- **BLOCKED:** Thread blocked waiting for monitor lock
- **WAITING:** Thread waiting indefinitely for another thread
- **TIMED\_WAITING:** Thread waiting for specified time period
- **TERMINATED:** Thread completed execution

## 32. What is synchronization and why is it needed?

Mechanism to control access to shared resources by multiple threads.

Prevents:

**Race conditions:** Unpredictable results due to timing

**Data inconsistency:** Corruption of shared data

## 33. Explain wait(), notify(), and notifyAll() methods.

- **wait():** Thread releases lock and waits until notified
- **notify():** Wakes up single waiting thread
- **notifyAll():** Wakes up all waiting threads
- **Must be called within synchronized context.**

# Java Interview Questions

## 34. What is the difference between synchronized method and synchronized block?

- **Synchronized method:** Entire method is synchronized, acquires lock on object/class
- **Synchronized block:** Only specific code block is synchronized, can specify custom lock object

## 35. What is serialization and deserialization?

- **Serialization:** Converting object to byte stream for storage/transmission
- **Deserialization:** Converting byte stream back to object
- Implement Serializable interface, use ObjectOutputStream / ObjectInputStream.

## 36. What is the transient keyword?

Marks instance variables to be excluded from serialization. Transient fields are not saved during serialization.

## 37. What is reflection in Java?

Runtime inspection and manipulation of classes, methods, and fields. Used by frameworks for:

- Dependency injection
- ORM mapping
- Testing frameworks
- Access through Class object: `Class.forName("className")`

# Java Interview Questions

## 38. What is the difference between == and equals()?

- == : Compares references (memory addresses) for objects, values for primitives
- equals() : Compares object content/state (if properly overridden)

## 39. Why should you override hashCode() when overriding equals()?

**Contract requirement:** If two objects are equal according to equals(), they must have the same hash code. Ensures proper behavior in hashbased collections (HashMap, HashSet).

## 40. What is the Comparable vs Comparator interface?

- **Comparable:** Defines natural ordering within class, implement compareTo() method
- **Comparator:** Defines external comparison logic, implement compare() method

## 41. What are the different ways to read a file in Java?

- **FileInputStream/FileReader:** Byte/character streams
- **BufferedReader:** Buffered reading for better performance
- **Scanner:** Convenient parsing of primitive types
- **Files class (Java 7+):** Utility methods like Files.readAllLines()

## 42. What is the difference between FileInputStream and FileReader?

- **FileInputStream:** Reads raw bytes, suitable for binary files
- **FileReader:** Reads characters with default encoding, suitable for text files

# Java Interview Questions

## 43. What is NIO (New I/O)?

Nonblocking I/O introduced in Java 1.4, featuring:

- **Channels:** Connections to files/sockets
- **Buffers:** Containers for data
- **Selectors:** Monitor multiple channels for events

Better performance for highconcurrency applications.

## 44. What is dependency injection?

Design pattern where objects receive dependencies from external source rather than creating them. Benefits:

- **Loose coupling:** Easier testing and maintenance
- **Flexibility:** Easy to swap implementations
- **Testability:** Easy to mock dependencies

## 45. What is Maven?

Build automation and project management tool. Features:

- **Dependency management:** Automatic downloading of required libraries
- **Standard project structure:** Consistent layout across projects
- **Build lifecycle:** Standardized build phases (compile, test, package)

## 46. What is JUnit?

Unit testing framework for Java. Features:

- **Annotations:** @Test, @Before, @After for test lifecycle
- **Assertions:** Methods to verify expected results
- **Test runners:** Execute and report test results

Dheenadayalan Dhanapal

---

# Java Interview Questions

## 47. What are some Java performance best practices?

- Use appropriate collection types for use case
- Avoid unnecessary object creation in loops
- Use StringBuilder for string concatenation
- Implement proper equals() and hashCode()
- Use connection pooling for database operations
- Profile applications to identify bottlenecks

## 48. What is the difference between deep copy and shallow copy?

- **Shallow copy:** Creates new object but references same nested objects
- **Deep copy:** Creates new object with copies of all nested objects  
Implement Cloneable interface or use serialization for deep copy.

## 49. Explain the Singleton design pattern.

Ensures only one instance of class exists. Implementation approaches:

- **Eager initialization:** Instance created at class loading
- **Lazy initialization:** Instance created when first requested
- **Threadsafe:** Using synchronization or enum approach

## 50. What are some common coding standards in Java?

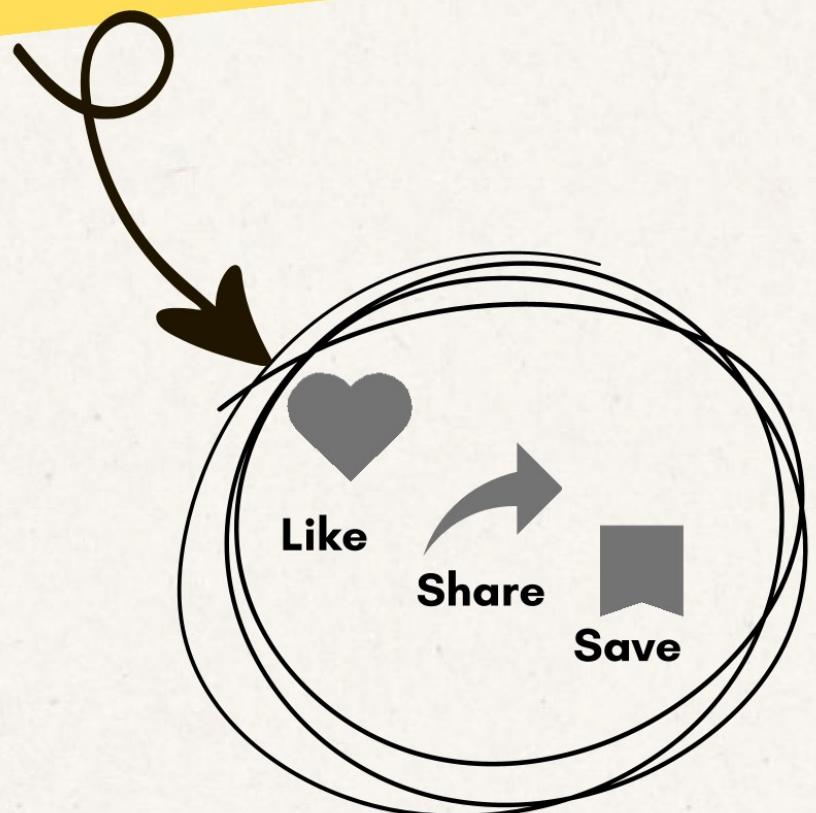
- **Naming conventions:** CamelCase for methods/variables, PascalCase for classes
- **Package naming:** Reverse domain name (com.company.project)
- **Code organization:** One public class per file
- **Documentation:** Use Javadoc for public APIs
- **Exception handling:** Don't catch generic Exception unless necessary
- **Constants:** Use static final variables with UPPER\_CASE naming

Dheenadayalan Dhanapal

---

# **THANK YOU !**

**If You Need More Posts Like This !**



Dheenadayalan Dhanapal