GENERAL

interview questions on core java with answers

Published: September 3, 2026 | Index ID: #-

Mastering Core Java Interview Questions: A Complete Guide with Answers

Landing a Java developer position often hinges on how well you can articulate your knowledge of core Java concepts. Recruiters and hiring managers typically start with a set of fundamental questions that probe your understanding of object oriented design, language syntax, and runtime behavior. Whether you're a seasoned developer or a fresh graduate, this guide offers a curated list of the most common interview questions on core Java, complete with concise, high quality answers that showcase your expertise.

Why Core Java Knowledge Matters

Core Java forms the foundation for all Java applications, from small utilities to large enterprise systems. Mastery of its core features ensures that you can:

- Write clean, maintainable code that follows best practices.
- Diagnose performance bottlenecks and memory leaks.
- Understand how the JVM manages objects and threads.
- Leverage Java's rich standard library for rapid development.

Interviewers use core Java questions to gauge not only your technical competence but also your problem solving mindset and your ability to communicate complex ideas clearly.

Essential Core Java Topics Covered in Interviews

Below is a quick snapshot of the key areas that frequently surface in interviews:

1. Object Oriented Principles (Encapsulation, Inheritance, Polymorphism, Abstraction)
2. Java Language Syntax (Classes, Interfaces, Enums, Annotations)
3. Collections Framework
4. Concurrency and Threading
5. Exception Handling
6. Java Memory Model & Garbage Collection
7. Input/Output Streams
8. Java 8+ Features (Streams, Lambda, Optional)
9. Design Patterns in Java

Below we dive into specific questions for each of these categories.

Object Oriented Principles

1. What is the difference between an abstract class and an interface?

Answer: An abstract class can provide both abstract methods (without implementation) and concrete methods (with implementation). It may also hold state (instance variables). An interface, prior to Java 8, could only declare abstract methods; however, Java 8 introduced default and static methods, and Java 9 added private methods. Interfaces cannot maintain instance state (except static final fields). Use abstract classes when you need shared

code or state; use interfaces to define contracts that can be mixed into unrelated classes.

2. Explain encapsulation with an example.

Answer: Encapsulation is the bundling of data (fields) and the methods that operate on that data into a single unit—a class—and restricting access to the inner workings. For example:

```
public class BankAccount {
    private double balance; // private field

    public double getBalance() { return balance; }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }
}
```

Clients interact through public methods; the internal representation of balance remains hidden.

3. What is polymorphism and how is it achieved in Java?

Answer: Polymorphism allows objects of different classes to be treated as instances of a common superclass or interface. It is achieved via method overriding (runtime polymorphism) and method overloading (compile time polymorphism). For example:

```
public interface Shape {
    double area();
}

public class Circle implements Shape {
    private double radius;
    public double area() { return Math.PI * radius * radius; }
}

public class Rectangle implements Shape {
    private double width, height;
    public double area() { return width * height; }
}
```

Code that operates on Shape can use any concrete implementation.

Java Language Syntax

4. What are the key differences between a class and an interface?

Answer:A class defines a concrete implementation and can have fields, constructors, and instance methods. An

interface defines a contract; it can only contain abstract methods (until Java 8) and static or default methods. A class can extend one superclass but implement multiple interfaces.

5. Explain the use of the final keyword in Java.

Answer: The final keyword has three primary uses:

1. When applied to a class, it cannot be subclassed.
2. When applied to a method, it cannot be overridden.
3. When applied to a variable, it becomes a constant (its value cannot change).

6. What is the purpose of the super keyword?

Answer:super references the immediate superclass of the current class. It is used to invoke superclass constructors, call overridden methods, or access superclass fields that are hidden by subclass fields.

Collections Framework

7. Describe the difference between ArrayList and LinkedList.

Answer: ArrayList stores elements in a resizable array; it offers O(1) random access but O(n) insertion/deletion in the middle. LinkedList is a doubly linked list; it provides O(1) insertion/deletion once you have the node reference but O(n) random access. Choose ArrayList for frequent reads, LinkedList for frequent insertions/deletions.

8. How does a HashMap work internally?

Answer: HashMap uses an array of buckets. Each key is hashed to an index; collisions are resolved via chaining (linked lists) or, since Java 8, by converting chains into balanced trees when the bucket size exceeds a threshold. Retrieval is O(1) on average, but worst case O(n) if many keys hash to the same bucket.

9. What is the difference between HashSet and TreeSet?

Answer: HashSet uses a hash table; it offers O(1) insertion, deletion, and lookup, but does not maintain order. TreeSet is a red black tree; it stores elements in sorted order and offers O(log n) operations. Use HashSet for uniqueness without ordering; use TreeSet when sorted iteration is required.

Concurrency and Threading

10. Explain the difference between synchronized methods and ReentrantLock.

Answer:synchronized is a language feature that automatically acquires a monitor lock on the object or class. It is simple but less flexible. ReentrantLock, part of java.util.concurrent.locks, offers advanced features: try lock with timeout, fairness policies, and condition variables. ReentrantLock is preferable in complex concurrent scenarios.

11. What is a deadlock? Provide an example.

Answer: A deadlock occurs when two or more threads wait indefinitely for resources held by each other. Example:

Thread A holds lock1 and waits for lock2.
Thread B holds lock2 and waits for lock1.
Both threads block forever.

To avoid deadlocks, enforce a consistent lock ordering, use timeouts, or apply lock hierarchies.

12. How does the volatile keyword affect thread visibility?

Answer:volatile ensures that writes to a variable are immediately visible to other threads. It prevents caching of the variable in CPU registers or thread local memory. However, volatile does not provide atomicity for compound actions (e.g., incrementing a counter). For atomicity, use AtomicInteger or synchronized blocks.

Exception Handling

13. Distinguish between checked and unchecked exceptions.

Answer: Checked exceptions (e.g., IOException) must be declared in a method's throws clause or caught within the method. Unchecked exceptions (runtime exceptions like NullPointerException) do not require explicit handling; they propagate up the call stack automatically. Use checked exceptions for recoverable conditions; use unchecked for programming errors.

14. What is the purpose of the finally block?

Answer: The finally block executes regardless of whether an exception was thrown or caught. It's ideal for resource cleanup (closing files, releasing locks). Example:

```
try {
    // risky code
} catch (IOException e) {
    // handle
} finally {
    resource.close(); // always executed
}
```

15. Explain try-with-resources and its advantages.

Answer: Introduced in Java 7, try-with-resources automatically closes any AutoCloseable resources declared in the try header. It eliminates boilerplate finally blocks and reduces the risk of resource leaks. Example:

```
try (BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {
    return br.readLine();
}
```

Java Memory Model & Garbage Collection

16. What are the main memory areas in a JVM?

Answer: The JVM memory layout includes:

- Heap – runtime data area for objects and arrays.
- Stack – per thread storage for local variables and method frames.
- Method Area – stores class metadata (constant pool, field and method definitions).
- Native Method Stacks – for native code.
- Program Counter Register – per thread, points to the current instruction.

17. How does garbage collection work in Java?

Answer: The garbage collector (GC) automatically reclaims memory occupied by objects that are no longer

reachable from any live thread. Common algorithms include Mark Sweep, Copying, and Generational GC. The JVM categorizes objects into Young, Old, and Permanent generations, optimizing collection frequency and duration.

18. What is the difference between System.gc() and the GC running automatically?

Answer: System.gc() merely requests a GC run; the JVM may ignore it. Automatic GC decisions are based on memory thresholds, generational heuristics, and performance goals. Relying on System.gc() can degrade performance; instead, design memory efficient code.

Input/Output Streams

19. Explain the difference between InputStream and Reader.

Answer: InputStream reads raw bytes and is suitable for binary data. Reader reads characters and handles character encoding (UTF 8, ISO 8859 1, etc.). Use InputStream for images or network streams; use Reader for text files.

20. What is the purpose of BufferedReader and BufferedWriter?

Answer: Buffered streams wrap underlying streams to reduce the number of I/O operations by storing data in a buffer. This improves performance when reading or writing many small chunks.

21.

Baca Juga (Artikel Terkait)

- **advanced engineering electromagnetics solution manual edition hardcover by balanis constantine a published by wiley**
  URL: http://denniswildberger.com/advanced-engineering-electromagnetics-solution-manual-edition-hardcover-by-balanis-constantine-a-published-by-wiley.pdf
- **advanced engineering electromagnetics solutions manual**
  URL: http://denniswildberger.com/advanced-engineering-electromagnetics-solutions-manual.pdf
- **advanced engineering electromagnetics wiley 1989 grading**
  URL: http://denniswildberger.com/advanced-engineering-electromagnetics-wiley-1989-grading.pdf
- **advanced engineering fluid mechanics by biswas**
  URL: http://denniswildberger.com/advanced-engineering-fluid-mechanics-by-biswas.pdf

Tags: #interview #questions #core #java #with #answers