Basics & Data Types

1. **What are the primitive data types in Java?**
   → byte, short, int, long, float, double, boolean, char.
2. **What is the default value of a boolean?**
   → `false`.
3. **What is the difference between `int` and `Integer`?**
   → `int` is a primitive type; `Integer` is an object wrapper class.
4. **What is autoboxing and unboxing?**
   → Autoboxing is converting a primitive to a wrapper (`int` to `Integer`). Unboxing is the reverse.
5. **What is the size of a `char` in Java?**
   → 16 bits (2 bytes); it uses Unicode.
6. **What is the range of `int` in Java?**
   → –2,147,483,648 to 2,147,483,647.
7. **What is the difference between `float` and `double`?**
   → `float` is 32-bit, `double` is 64-bit; double is more precise.
8. **What is a `String` in Java?**
   → A sequence of characters; it is immutable.
9. **How to compare two strings in Java?**
   → Use `.equals()` for content, `==` for reference comparison.
10. **What is `final` keyword in Java?**
    → It can make a variable constant, a method non-overridable, or a class non-inheritable.

OOP (Object-Oriented Programming)

11. **What are the pillars of OOP?**
    → Encapsulation, Inheritance, Polymorphism, Abstraction.
12. **What is encapsulation?**
    → Hiding internal data using private fields and public getters/setters.
13. **What is method overloading?**
    → Same method name, different parameters within the same class.
14. **What is method overriding?**
    → Redefining a method from the parent class in a subclass.
15. **Can we override static methods?**
    → No, static methods belong to the class, not the instance.
16. **What is a constructor?**
    → A special method used to initialize objects.
17. **Can constructors be overloaded?**
    → Yes, by changing the parameter list.
18. **What is the purpose of `this` keyword?**
    → Refers to the current object instance.
19. **What is the difference between `super` and `this`?**
    → `super` refers to the parent class; `this` refers to the current class.
20. **What is abstraction?**
    → Hiding implementation details and showing only the functionality.

Inheritance, Interfaces & Classes

21. **Can Java support multiple inheritance?**
    → Not with classes, but yes using interfaces.
22. **What is an interface?**
    → A contract that defines abstract methods; implemented by classes.
23. **What is the difference between abstract class and interface?**
    → Abstract class can have method bodies; interfaces (before Java 8) couldn't.
    Interfaces support multiple inheritance.
24. **Can an interface have default methods?**
    → Yes, from Java 8 onwards using `default` keyword.
25. **Can we instantiate an abstract class?**
    → No, it must be extended.
26. **What is the access modifier `protected`?**
    → Visible to the same package and subclasses.
27. **What is a nested class?**
    → A class defined inside another class (static or non-static).
28. **What is the difference between `public`, `private`, and `default` access?**
    → `public`: everywhere; `private`: same class only; `default`: same package.
29. **What is a static class in Java?**
    → Only nested classes can be static; top-level classes cannot.
30. **Can we have static methods in interfaces?**
    → Yes, from Java 8 onwards.

Memory, Exceptions, and Collections Basics

31. **What is the JVM?**
    → Java Virtual Machine – runs compiled Java bytecode.
32. **What is garbage collection?**
    → Automatic memory management to delete unused objects.
33. **What is the difference between stack and heap?**
    → Stack: stores method calls and local variables;
    Heap: stores objects and class instances.
34. **What are checked and unchecked exceptions?**
    → Checked: must be handled (e.g., `IOException`);
    Unchecked: runtime exceptions (e.g., `NullPointerException`).
35. **What is `finally` block used for?**
    → Code that runs no matter what – even if an exception is thrown.
36. **What is the use of `try-with-resources`?**
    → Automatically closes resources like files or DB connections.
37. **What is the difference between `Array` and `ArrayList`?**
    → Arrays have fixed size and store primitives or objects;
    `ArrayList` is resizable and part of the Collections framework.
38. **What is a `HashMap`?**
    → A key-value pair data structure that allows fast retrieval using keys.

39. **What is the difference between == and `.equals()` in objects?**
    → == compares references; `.equals()` compares values (if overridden).
40. **What is immutability in Java?**
    → Once an object is created, its state cannot change.
    → Example: `String`, `Integer`, `LocalDate`.

Advanced & Popular Java Interview Questions and Answers (3+ Years Experience)

✪ Advanced OOP & Java Concepts

1. What is the difference between composition and inheritance?

- Inheritance represents an "is-a" relationship. Composition represents a "has-a" relationship.
- Composition is more flexible and is generally preferred for better encapsulation and code reuse.

2. What is the `Object` class in Java?

- It is the root class of all Java classes. Common methods include `toString()`, `equals()`, `hashCode()`, `clone()`, and `getClass()`.

3. What is the `clone()` method?

- It creates and returns a copy of the object. It performs a shallow copy by default.

4. What is the difference between shallow copy and deep copy?

- Shallow copy copies object references; deep copy duplicates all objects.

5. What is the `transient` keyword?

- It marks a variable as non-serializable. The value is not saved during serialization.

6. What are enums in Java?

- Enums are special classes for constants. They can have fields, methods, and constructors.

✪ Exception Handling

7. What's the difference between `throw` and `throws`?

- `throw` is used to explicitly throw an exception. `throws` declares exceptions that a method might throw.

8. Can you catch multiple exceptions in one catch block?

- Yes, using the pipe `|` operator since Java 7: `catch (IOException | SQLException e)`

9. Best practice for creating custom exceptions?

- Extend `Exception` or `RuntimeException`. Include constructors and optionally a message/ cause.

✪ Collections & Data Structures

10. Difference between `HashMap`, `Hashtable`, `ConcurrentHashMap`?

- `HashMap`: not thread-safe, allows null.
- `Hashtable`: thread-safe (legacy), no null keys.
- `ConcurrentHashMap`: thread-safe, better performance in concurrent apps.

11. How does `HashMap` work internally?

- Uses `hashCode()` to find the bucket, then `equals()` for key matching. Uses array + linked list or red-black trees.

12. Difference between `List`, `Set`, `Map`?

- `List`: ordered, duplicates allowed.
- `Set`: unordered, no duplicates.
- `Map`: key-value pairs.

13. Why doesn't `Set` allow duplicates?

- It uses `equals()` and `hashCode()` to ensure uniqueness.

14. Difference between `ArrayList` and `Vector`?

- `ArrayList`: not synchronized.
- `Vector`: synchronized but slower.

15. What is fail-fast vs fail-safe iterator?

- Fail-fast: throws `ConcurrentModificationException`.
- Fail-safe: works on a cloned copy (e.g., `ConcurrentHashMap`).

✪ Concurrency & Multithreading

16. Ways to create a thread?

- Extend `Thread` class or implement `Runnable`/`Callable`.

17. Difference between `Runnable` and `Callable`?

- `Runnable`: returns no result, can't throw checked exceptions.
- `Callable`: returns result, can throw checked exceptions.

18. Purpose of `synchronized`?

- Ensures mutual exclusion and visibility of changes across threads.

19. Difference between `wait()`, `notify()`, `notifyAll()`?

- `wait()` pauses the thread, `notify()` wakes one thread, `notifyAll()` wakes all waiting threads.

20. Thread-safe collections in Java?

- `Vector`, `Hashtable`, `ConcurrentHashMap`, `CopyOnWriteArrayList`.

21. What is a deadlock? How to prevent it?

- When two threads wait for each other's resources. Prevent by locking resources in a consistent order or using `tryLock()`.

✪ Java 8+ Features

22. What are lambda expressions?

- Shorter syntax for functional interfaces: `(a, b) -> a + b`

23. What is the Stream API?

- A functional-style API for processing collections (e.g., map, filter, reduce).

24. Difference between `map()`, `filter()`, `reduce()`?

- `map()`: transforms elements.
- `filter()`: filters by condition.
- `reduce()`: aggregates values.

25. What are functional interfaces?

- Interfaces with one abstract method (e.g., `Runnable`, `Callable`, `Function`).

26. What is `Optional` used for?

- Avoid `NullPointerException`. It wraps a value that may be null.

✪ Memory & Performance

27. How does garbage collection work?

- JVM automatically deletes unreachable objects to free memory.

28. Can memory leaks happen in Java?

- Yes, if references are unintentionally held (e.g., static lists).

29. Strong vs Weak vs Soft references?

- Strong: GC never removes.
- Weak: removed at next GC.
- Soft: removed when memory is low.

30. What is PermGen/Metaspace?

- PermGen (Java 7): stores class metadata. Replaced by Metaspace in Java 8, which grows dynamically.

31. How do you profile Java performance?

- Use tools like VisualVM, JConsole, or profilers to monitor CPU/memory usage.

✪ File I/O & Serialization

32. What is serialization?

- Converting an object into a byte stream to save or transmit.

33. Superclass not serializable: what happens?

- Its fields are not serialized unless manually handled.

34. Difference between `FileReader` and `BufferedReader`?

- `BufferedReader` adds buffering for efficiency.

35. How to read/write files using NIO?

- Use `Files.readAllLines()`, `Paths`, and `BufferedWriter` from `java.nio.file`.

✪ Best Practices

36. What is clean code?

- Code that is readable, maintainable, and well-structured with meaningful names.

37. What design patterns have you used?

- Singleton, Factory, Builder, Observer, Strategy, etc.

38. What is test-driven development (TDD)?

- Writing tests before writing the code to fulfill those tests.

39. How do you handle exceptions and logging?

- Use try-catch, custom exceptions, and libraries like SLF4J or Log4j.

40. How do you make classes testable?

- Use dependency injection, avoid static methods, and separate concerns.