Java Interview Questions

Ajay'sTech

Basic Interview Questions

Q1. What are the data types in Java? Java has two types of data types:

1. Primitive Data Types: These include byte, short, int, long, float, double, char, and boolean. They store simple values and are not objects.
2. Non-Primitive Data Types: These include String, Array, Class, and Interface. They are derived from primitive data types and provide more functionalities.

Q2. What are wrapper classes?

Wrapper classes provide an object representation of primitive data types, such as Integers, Doubles, and Booleans. These classes allow primitives to be used in collections and provide useful utility methods.

Q3. Are there dynamic arrays in Java?

Java arrays are fixed indynamically. Q4 size. However, ArrayList (from the Java.util package) provides a dynamic array implementation where elements can be added or removed dynamically.

Q4. What is JVM?

The Java Virtual Machine (JVM) is a part of the Java Runtime Environment (JRE). It is responsible for executing Java bytecode by converting it into machine code specific to the operating system.

Q5. Why is Java platform-independent?

Java achieves platform independence through bytecode. The Java compiler converts code into bytecode, which the JVM interprets for the underlying OS, making Java write-once, run-anywhere.

Q6. What are local and global variables?

- Local variables are declared inside methods or blocks and are accessible only within their scope.
- Global variables (also called instance variables) are declared within a class but outside any method and have a wider scope.

Q7. What is data encapsulation?

Encapsulation is an OOP principle where data (variables) and code (methods) are bundled into a single unit (class). It restricts direct access to data using access modifiers (private, protected).

Q8. What is function overloading?

Function overloading allows multiple methods to have the same name but different parameter lists. The compiler differentiates them based on the number or type of parameters.

Example:

```
public class Figure
{
     public int area(int a, int b)
     {
     int rectangleArea = a*b;
     return rectangleArea;
     }
     public  int area(int a)
     {
     int squareArea = a*a;
     return squareArea;
     }
public static void main(String[] args ){
     Figure f = new Figure();
     System.out.println("Area of square " + f.area(5));
     System.out.println("Area of Rectangle " + f.area(5,3));
     }
}
```

Q9. What is function overriding?

Overriding allows a subclass to provide a specific implementation of a method defined in its superclass. It enables dynamic method dispatch (runtime polymorphism).

Q10. Why is the main method static in Java?

The main method is static so that it can be called without creating an instance of the class, allowing the program to start execution without object instantiation.

Q11. What is the difference between the throw and throws keywords in Java?

Feature	throw	throws
Purpose	`Used to explicitly throw an exception`	`Declares that a method may throw an exception`
Usage	`throw new Exception("Error")`	`public void myMethod() throws IOException`
Number of Exceptions	`Can throw one exception at a time`	`Can declare multiple exceptions using commas`

Q12. What do you mean by singleton class?

A singleton class ensures that only one instance of the class exists throughout the application's lifecycle. It is implemented using a private constructor, a static instance variable, and a public static method that returns the single instance. The most common way to create a singleton class is using the lazy initialization or eager initialization approach.

Q13. Does every try block need a catch block?

No, a try block does not necessarily need a catch block. It can be followed by either a catch block, a final block, or both. A catch block handles exceptions that may arise in the try block, while a final block ensures that certain code (such as resource cleanup) is executed regardless of whether an exception occurs.

Q14. What is the usage of the super keyword in Java?

The super keyword in Java is used to refer to the parent class. It can be used to:

1. Call the constructor of the parent class.
2. Access the parent class's methods and variables when they are overridden in a subclass.
3. Differentiate between methods and attributes of the parent and child class when they have the same name.

Q15. What do you mean by the final keyword?

The final keyword is used to restrict modifications in Java. It can be applied in three contexts:

1. Final variable: Its value cannot be changed once assigned.
2. Final method: Prevents method overriding in subclasses.
3. Final class: Prevents inheritance by other classes.

Q16. How is an exception handled in Java?

Java handles exceptions using the try-catch-finally mechanism:

1. Try block: Contains the code that might generate an exception.
2. Catch block: Handles the exception and defines what should be done when an error occurs.
3. Finally block: Executes regardless of whether an exception occurs or not, often used for resource cleanup (e.g., closing files or database connections).

Q17. How can objects in a Java class be prevented from serialization?

Serialization converts an object into a byte stream for storage or transmission. To prevent serialization:

1. Declare fields as transient to exclude them from serialization.
2. Implement writeObject() and readObject() methods to control serialization.
3. Extend NotSerializableException to explicitly prevent serialization.

Q18. What is the difference between a constructor and a method in Java?

Constructor	Method
It has no return type.	It always has a return type. It has a return type void when not returning anything.
It always has the same name as the class name.	It can have any name of its choice.

Q19. Why is reflection used in Java?

Reflection in Java allows a running program to inspect and manipulate its methods, fields, and constructors at runtime. It is commonly used in frameworks, debugging tools, and JavaBeans to dynamically access class properties.

Q20. What are the different types of ClassLoaders in Java?

Java provides three main types of ClassLoaders:

1. Bootstrap ClassLoader: Loads core Java classes from rt.jar and other essential libraries. It is implemented in native code and does not have a Java class representation.
2. Extension ClassLoader: Loads classes from the JRE/lib/ext directory or any other specified extension directories. It is implemented as sun.misc.Launcher$ExtClassLoader.
3. System (Application) ClassLoader: Loads application classes from the classpath (defined by CLASSPATH, -cp, or -classpath options). It is a child of the Extension ClassLoader.

Q21. What is a copy constructor in Java?

A copy constructor creates a new object by copying the properties of an existing object. It takes an instance of the same class as an argument and initializes the new object with the same values.

Q22. What is object cloning in Java?

Object cloning is a way to create an exact copy of an object. Java provides the clone() method from the Cloneable interface to perform shallow copies. A shallow copy copies field values but does not duplicate referenced objects, while a deep copy creates new instances of referenced objects.

Q23. Is Java a purely object-oriented language?

No, Java is not purely object-oriented because it supports primitive data types like int, char, boolean, and double, which are not objects. A purely object-oriented language would require every entity to be an object.

Q24. What is a package in Java?

A package in Java is a collection of related classes and interfaces grouped to organize code and prevent naming conflicts.

- Built-in packages: java.lang, java.util, etc.
- User-defined packages: Created by developers for organizing custom classes.

Q25. What is coercion in Java?

Coercion in Java refers to the automatic or explicit conversion of one data type into another.

- Implicit coercion: Automatically converts smaller data types to larger ones (e.g., int to double).
- Explicit coercion (casting): Converts larger data types to smaller ones using type casting (e.g., (int) 3.14).

Q26. Can a private method be overridden in Java?

No, private methods cannot be overridden because they are not accessible outside their class. If a subclass defines a method with the same name, it is treated as a separate method rather than an override.

Q27. What are the phases in the lifecycle of a thread in Java?

A Java thread goes through the following states:

1. New: The thread is created but has not started executing.
2. Runnable: The thread is ready to run and waiting for CPU allocation.
3. Blocked: The thread is waiting for a resource or lock to be available.
4. Waiting: The thread is indefinitely waiting for another thread to notify it.
5. Timed Waiting: The thread waits for a specified time (e.g., using `Thread.sleep()`).
6. Terminated: The thread has completed execution or stopped due to an error.

Q28. What is a marker interface in Java?

A marker interface is an interface with no methods or fields, used to provide metadata to the JVM or compiler. Examples include `Serializable` and `Cloneable`. Modern Java prefers annotations over marker interfaces.

Q29. What is a memory leak in Java?

A memory leak occurs when objects that are no longer needed are not garbage collected because they are still referenced somewhere. This can cause excessive memory consumption and slow down the application.

Q30. What is the difference between `new` and `newInstance()` in Java?

- `New` is a keyword that creates a new object of a known class at compile time.
- `newInstance()` (from `Class`) creates an object dynamically at runtime, requiring reflection, and is slower because it involves additional security and access checks.

Q31. What is the difference between JDK, JRE, and JVM?

- **JDK (Java Development Kit)** – Contains **JRE + development tools** (compiler, debugger) for **developing and running** Java applications.
- **JRE (Java Runtime Environment)** – Includes **JVM + libraries** needed to **run** Java applications but lacks development tools.
- **JVM (Java Virtual Machine)** – Executes Java bytecode, providing **platform independence** and **memory management (GC).**

JDK > JRE > JVM – JDK includes JRE, and JRE includes JVM.
JDK is for developers, while **JRE is for users running Java applications**

Q32. What is the difference between abstraction and encapsulation?

- Abstraction hides implementation details and exposes only essential functionalities (e.g., using interfaces and abstract classes).
- Encapsulation bundles data and methods within a class and restricts direct access using access modifiers.

Q33. What is inheritance in Java?

Inheritance in Java is a mechanism where a **child class acquires properties and behaviors** of a **parent class**, promoting **code reusability** and **hierarchical relationships.**

- Achieved using the `extends` **keyword.**
- Supports **single and multilevel inheritance** (not multiple inheritance with classes).
- Allows method **overriding** for polymorphism.
- **The super (`super`) keyword** is used to access parent class members.

Q34. What are functional interfaces in Java 8?

Functional interfaces have exactly one abstract method and are used with lambda expressions.

Examples include Runnable, Callable, and Comparator.

Q35. What is polymorphism in Java?

Polymorphism allows the same method to behave differently based on the context.

- Compile-time polymorphism (Method Overloading): Methods with the same name but different parameters.
- Runtime polymorphism (Method Overriding): A subclass provides a specific implementation of a parent method.

Q36. What is the purpose of the default keyword in interfaces?

The default keyword allows methods in interfaces to have default implementations, enabling backward compatibility without forcing all implementing classes to override them.

Q37. What is an interface in Java?

An **interface** in Java is a **blueprint** for classes that defines a **contract** without implementation.

- Declared using the `interface` **keyword**.
- Contains **only abstract methods** (until Java 7).
- **Java 8+** allows **default and static methods** with implementations.
- Supports **multiple inheritance**.
- Implemented by classes using the `implements` **keyword**.

Q38. What is the difference between ArrayList and Vector?

- ArrayList is not synchronized (faster), while Vector is synchronized (thread-safe).
- ArrayList increases its size by 50% when full, while Vector doubles its size.

Q39. What is an abstract class?

An **abstract class** in Java is a class that **cannot be instantiated** and is meant to be **extended by subclasses**.

- Declared using the `abstract` **keyword**.
- Can have **both abstract (without implementation) and concrete methods.**
- Used for **partial implementation and code reusability**.
- **Must be extended** by a subclass that provides implementations for abstract methods.

Q40. What is the difference between `HashMap` and `ConcurrentHashMap`?

- HashMap is not thread-safe, while ConcurrentHashMap is thread-safe.
- ConcurrentHashMap locks only portions of the map, improving performance.
- HashMap allows one null key, but ConcurrentHashMap does not.

Q41. What is the difference between an abstract class and an interface?

- Abstract class: Can have both abstract and concrete methods.
- Interface: Contains only abstract methods (before Java 8) and supports multiple inheritance.

Q42. What is the Java Memory Model (JMM)?

The **Java Memory Model (JMM)** defines how **threads interact with memory** and ensures **visibility, ordering, and atomicity** of shared data in a **multi-threaded environment.**

- Controls how **variables are read/written across threads.**
- Ensures **happens-before relationships** to prevent race conditions.
- Uses **volatile, synchronized, and locks** for thread safety.

- Helps in **optimizing CPU caching and instruction reordering**.
- Ensures **safe and predictable concurrency behavior**.

Q43. What is this keyword in Java?

This refers to the current instance of a class, distinguishing between instance variables and parameters with the same name.

Q44. What are Java Generics?

Generics provide compile-time type safety by allowing a class, method, or interface to work with different types while avoiding runtime errors.

Q45. What are access modifiers in Java?

- Private: Accessible only within the same class.
- Default: Accessible within the same package.
- Protected: Accessible within the same package and subclasses.
- Public: Accessible from anywhere.

Q46. What is the purpose of the synchronized keyword?
Synchronized ensures that only one thread can execute a block of code or method at a time, preventing race conditions.

Q47. What is a static method in Java?

A **static method** in Java belongs to the **class**, not instances. It can be called using the **class name** without creating an object.

- Declared using the `static` **keyword.**
- Can **access only static variables and methods** directly.
- Cannot use `this` **or** `super`.
- Commonly used for **utility methods** (e.g., `Math.pow()`).

```
class Example {
  static void display() {
        System.out.println("Static Method");
    }
}
                    Example.display(); // Call without creating an object
```

Q48. What are Java 8 Streams?

Java 8 **Streams** provides a **functional programming** approach to processing data efficiently. They allow operations like **filtering, mapping, and reducing** collections in a **declarative and parallelizable** way.

- Supports **sequential (`stream()`) and parallel (`parallelStream()`) processing.**
- Uses **lazy evaluation** for optimized execution.
- Common methods: `filter()`, `map()`, `reduce()`, `collect()`, `forEach()`.

Q49. What is garbage collection in Java?

1. **Automatic Memory Management**: Java's Garbage Collector (GC) automatically reclaims memory by removing unused objects.
2. **Heap Memory Cleanup**: GC works in the heap, where objects are dynamically allocated.
3. **Identifies Unreachable Objects**: Objects with no active references are eligible for garbage collection.
4. **Prevents Memory Leaks**: Helps manage memory efficiently and avoids out-of-memory errors.
5. **No Manual Deallocation**: Unlike languages like C/C++, Java does not require explicit `free()` or `delete()`.
6. **Uses Mark and Sweep Algorithm**: Identifies live objects (mark) and removes dead ones (sweep).
7. **JVM Optimization for GC**: JVM parameters like `-Xms`, `-Xmx`, `-XX:+UseG1GC` help tune GC performance.

Q50. What is the difference between implements and extends?

- Implements are used for interfaces.
- Extends are used for class inheritance.

Intermediate Interview Questions:

Q51. What is the purpose of the "assert" statement in Java?

The assert statement in Java is used to validate assumptions during development and debugging. It checks whether a given expression evaluates to true. If the condition is false, an AssertionError is thrown.

Assertions are mainly used for debugging and testing purposes to detect logical errors early in the development process.

Q52. What is the difference between ArrayList and LinkedList in Java?

- ArrayList: A resizable array-based data structure that provides fast random access (O(1)) but slower insertions and deletions (O(n)) due to shifting elements. It is preferred when frequent access operations are needed.
- LinkedList: A doubly linked list implementation that allows efficient insertions and deletions (O(1)) but slower random access (O(n)). It is more suitable for scenarios with frequent modifications and dynamic resizing.

Q53. What is the purpose of the hashCode() method in Java?

The hashCode() method generates a unique integer value (hash code) that represents an object's contents. It is primarily used in hash-based collections like HashMap, HashSet, and Hashtable for efficient storage and retrieval. A properly implemented hashCode() ensures better performance in hashing-based operations.

Q54. What is the purpose of the toString() method in Java?

The toString() method returns a string representation of an object, typically including its class name and key attributes. It enhances debugging, logging, and object readability. When an object is printed or concatenated with a string, the toString() method is implicitly called. Developers often override it to provide meaningful output.

Q55. How is encapsulation achieved in Java?

Encapsulation in Java is achieved using access modifiers (private, protected, public) to restrict direct access to class members. It ensures data hiding and maintains control over how data is accessed and modified. Getters and setters are commonly used to enforce controlled access to private variables.

Q56. What are method references in Java?

Method references provide a shorthand way to refer to existing methods without executing them immediately. They improve code readability by replacing lambda expressions with direct references to class or instance methods. They are used with functional interfaces and written as Class::methodName.

Example: System.out::println instead of x -> System.out.println(x).

Q57. What are annotations in Java?

Annotations in Java are metadata tags used to provide additional information about code elements such as classes, methods, and variables.

They start with @ (e.g., @Override, @Deprecated) and are used for configuration, code documentation, validation, and runtime processing. Annotations enhance code organization and facilitate frameworks like Spring and Hibernate.

Q58. What is the BitSet class used for in Java?

The BitSet class represents a sequence of bits that can grow dynamically. It provides efficient bitwise operations such as setting, clearing, flipping, and checking bits. It is used for memory-efficient handling of binary data, such as flags, filters, and permission settings.

Q59. What is a CyclicBarrier in Java?

A CyclicBarrier is a synchronization mechanism that allows multiple threads to wait at a common barrier point until all threads reach it. Once all participating threads arrive, they proceed together. It is useful in parallel programming scenarios where tasks must be synchronized before continuing execution.

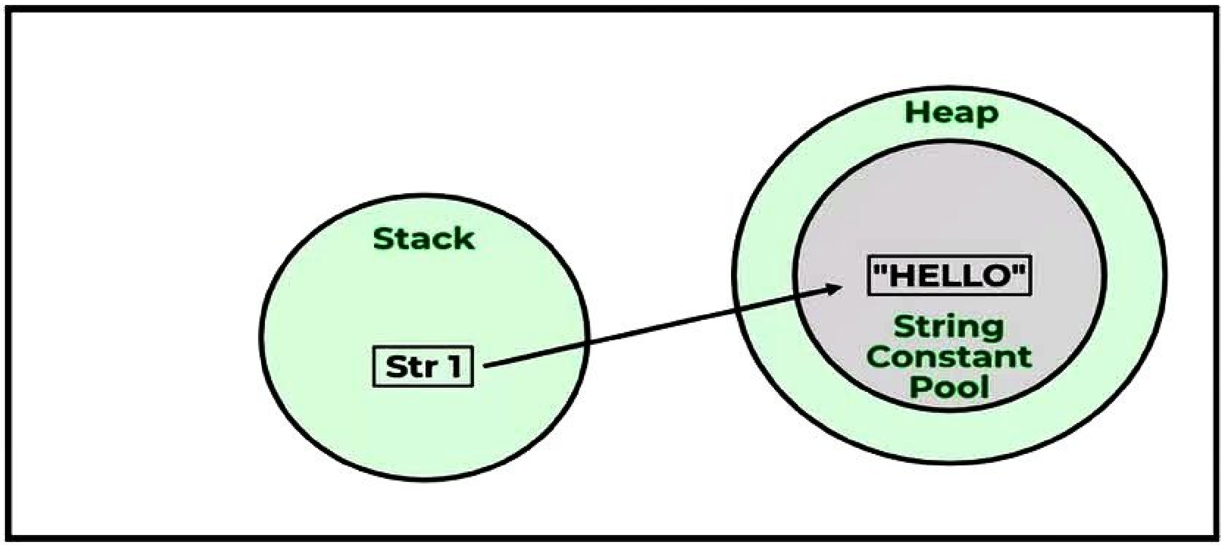
Q60. What are the types of JDBC statements in Java?

JDBC (Java Database Connectivity) provides three types of statements for database interaction:

- Statement: Executes simple SQL queries without parameters.
- PreparedStatement: Precompiled SQL statement with placeholders for parameters, improving performance and security.
- CallableStatement: Used for executing stored procedures in the database.

Q61. What is the Java String Pool?

The Java String Pool is a memory optimization technique where the JVM stores string literals in a common pool to avoid redundant allocations. When a new string literal is created, the JVM checks the pool first; if the string exists, it reuses the reference instead of creating a new object. This conserves memory and enhances performance.



Example:

```
String str1="Hello";
// "Hello" will be stored in String Pool
// str1 will be stored in stack memory
```

Q62. What is the difference between Path and Classpath?

1. Path specifies the location of **system executables** like `Java` and `javac`.

   Classpath specifies the location of **Java class files, JARs, and resources.**

2. Path is used by the **operating system**, while Classpath is used by the **JVM.**
3. The path is set via the `PATH` environment variable, whereas Classpath is set using `CLASSPATH` or `-cp` option.
4. Example: `set PATH=C:\Java\bin` (Path), `set CLASSPATH=C:\libs\myLib.jar` (Classpath).

Q63. What is the difference between Heap and Stack memory?

- Heap Memory: Stores objects and data that need to persist throughout the program. Managed by the JVM's garbage collector.
- Stack Memory: Holds method call frames and local variables. It manages method execution and automatically deallocates memory when a method exits.

Q64. Can we use String with a switch-case statement?

Yes, **String** can be used in a **switch-case** statement from **Java 7 onwards.** The switch works by **computing the hashcode** of the string and then comparing it.

```
String fruit = "Apple";
switch (fruit) {
   case "Apple": System.out.println("It's an Apple!"); break;
   case "Mango": System.out.println("It's a Mango!"); break;
   default: System.out.println("Unknown fruit");
}
```

Key Points:

- **Case labels are case-sensitive** (`"Apple" ≠ "apple"`).
- **Uses `String.hashCode()` for comparison** internally.
- **Less efficient than integer-based switches** due to hashing.

Q65. What are the different types of class loaders?

Java has three primary class loaders:

1. Bootstrap ClassLoader – Loads core Java classes (e.g., rt.jar).
2. Extension ClassLoader – Loads classes from the JDK's extensions directory ($JAVA_HOME/lib/ext).
3. System (Application) ClassLoader – Loads application classes from the classpath, configurable using -cp or -classpath.

Q66. What is the difference between fail-fast and fail-safe iterators?

- Fail-fast iterators: Immediately throw a ConcurrentModificationException if a collection is modified while iterating.
- Fail-safe iterators: Operate on a cloned copy of the collection, allowing modifications without exceptions.

Q67. What is a compile-time constant in Java?

A compile-time constant is a value assigned at the time of compilation and remains unchanged throughout execution. Example:

static final int MAX_VALUE = 100;

Q68. What is the difference between Map and Queue in Java?

- Map: Stores key-value pairs and allow fast retrieval based on keys (e.g., HashMap).
- Queue: Stores elements in a specific order (FIFO, Priority-based, etc.), designed for processing elements sequentially.

Q69. What is the difference between LinkedHashMap and PriorityQueue?

- LinkedHashMap: Maintains insertion order and maps keys to values.
- PriorityQueue: Orders elements based on priority (natural ordering or a custom comparator).

Q70. What is a memory-mapped buffer in Java?

A memory-mapped buffer allows a file to be directly mapped into memory, improving I/O efficiency by enabling direct access to file contents. Useful for handling large files.

Q71. What is the difference between notify() and notifyAll()?

- notify(): Wakes up one waiting thread.
- notifyAll(): Wakes up all waiting threads.

Q72. What are the types of exceptions in Java?

1. Checked Exceptions (e.g., IOException) – Must be handled at compile-time.
2. Unchecked Exceptions (e.g., NullPointerException) – Occurs at runtime and doesn't require explicit handling.

Q73. What is OutOfMemoryError?

- Error when **JVM runs out of memory**.
- Caused by **memory leaks, large objects, or infinite loops**.

Q74. What is the difference between == and equals()?

- == compares references (memory addresses).
- equals() compares object content (must be overridden in custom classes).

Q75. How can you concatenate multiple strings in Java?

@TECHTALKS_WITH_AJAY