

1. What is the internal working of a hashmap
2. What is concurrent hashmap
3. What is the difference between final, finally and finalize
 - a. final is a modifier applicable for method, variable and class
 - b. finally is a block always associated with try catch to maintain cleaner code
 - c. finalize method is invoked by the garbage collector just before destroying an object to perform clean up tasks
4. What is the difference between notify and notifyAll
5. What is Polymorphism
 - a. It is an ability of an object to take many forms
 - i. Compile time polymorphism - Method overloading
 1. Based on the parameter the execution of the method is decided during the compile time itself
 - ii. Runtime polymorphism - Method overriding
 1. The method that is going to be executed will be decided only during the run time or during the execution, When a subclass provides implementation of a method declared in the super class
6. What is encapsulation
7. How to achieve polymorphism with inheritance
8. What are the methods of an Object class
9. How to create a thread using Class and interface
10. What is a marker interface
11. Does java allow multiple inheritance
12. What is String and StringBuffer
 - a. String is immutable and StringBuffer is mutable
 - b. Immutable means object which cannot be modified
 - c. In case of String I use the code `String s = "Demo"` and if I modify the concat a new String with it it will add the data and store it in a new reference variable if the result is not stored then it will be eligible for garbage collection.
13. What is the difference between == and equals
 - a. == is used for reference or address comparison whereas equals is used for content comparison
 - b. Example

```
String s1 = "Lingtan";
String s2 = "Lingtan";
```

in case of `s1==s2` it is false because both objects point to different addresses

incase of `s1.equals(s2)` it is true because both the content are same

14. StringBuffer vs StringBuilder

- a. Both are used when mutability is required and both are similar incase of requirement
- b. StringBuffer is synchronized so only one method is executed in a time by one thread other thread has to wait
- c. StringBuilder is not synchronized
- d. When performance is important and thread safety is not required then we can use StringBuilder

15. When to choose interface, abstract and concrete class

- a. If i don't have any idea of implementation and i only have the requirement specification i go for interface - 100% abstraction
- b. If a class has partially implemented methods then go for abstraction
- c. If complete implantation is provided then go for concrete

16. Access specifiers and access modifiers

- a. there is no term called specifiers in java all are called modifiers only

17. `C c1 = new C();` vs `P p1 = new P();`

- a. In this case if the return type of the data is not known then we can go for second approach because parent can hold any kind of child data
- b. But with second approach we can call only the parent object methods but in 1st we can call any parent or child methods


18. Exception and Error

- a. Throwable class acts as a root for exception hierarchy
- b. Throwable class contains 2 child classes Exception and Error
- c. Exceptions are caused by our code
- d. Errors are caused by lack of system resources example out of memory error

19. Checked and Unchecked exceptions

- a. There are some exceptions that are checked by the compiler that is called CheckedExceptions - `FileNotFoundException`
- b. Those which are not checked by the compiler is called unchecked

20. Modifiers in java

Modifier	Usage	Area of Usage
public	- Accessible from any other class.	Classes, Methods, Variables, Constructors
protected	- Accessible within the same package and by subclasses.	Methods, Variables, Constructors
default	- Accessible only within the same package.	Classes, Methods, Variables, Constructors
private	- Accessible only within the same class.	Methods, Variables, Constructors
static	- Belongs to the class rather than an instance.	Methods, Variables, Nested Classes
final	- Value cannot be changed or class/method cannot be overridden.	Classes, Methods, Variables
abstract	- Must be implemented by subclasses, cannot instantiate.	Classes, Methods
synchronized	- Used to control access to a method or block by multiple threads. 	Methods, Blocks

volatile	- Value of a variable is always read from the main memory.	Variables
transient	- Excluded from serialization.	Variables
native	- Indicates that a method is implemented in native code using JNI.	Methods
strictfp	- Restricts floating-point calculations to ensure portability.	Classes, Methods

Abstract class vs Interface

- Object cannot be created for an Abstract class
- Abstract classes can have both concrete and non concrete methods
- Abstract classes can be extended

- Best example is one developer creates an interface with 100 methods and another developer is asked to write the implementation for the 100 methods but he failed to write the 100 methods and he ended up writing only 70 methods, So now the class becomes useless until 100 methods are written so now we have converted the class into an abstract class so that no one can create an object of the class and use it now, another developer is called and asked him to extend this abstract class and write the implementation for the remaining methods, why creating a new class is because this developer can come and continue the work at anytime so we created a new file
- Interfaces contains only non concrete methods and default methods
- The purpose of default method is to avoid the concept of forced implementation because when i add a new method in the interface file the classes implementing the interface is forced to implement the unimplemented method even though i don't want the method
- To resolve this issue the default method is created so that the compiler won't force and if the user wants to use this method he has to override the method manually

Static variables

When a variable is declared static then only a single memory is used to store the static variable even though multiple object of the classes are created

Example im creating 2 variables in a class, 1st is static and 2nd is non static and im creating default constructor and incrementing the 2 variables everytime the object is created, now whenever the object is created the 2 variables gets incremented but when printing the variable the static variable count gets incremented and non static is not incremented, this is because the static variable is always stored in the same memory but the non static is not

<https://www.freecodecamp.org/news/static-variables-in-java/#:~:text=A%20static%20method%20means%20it,%2C%20performance%2C%20and%20code%20organization.>

```
StaticTest s1 = new StaticTest();
StaticTest s2 = new StaticTest();
StaticTest s3 = new StaticTest();
StaticTest s4 = new StaticTest();

System.out.println(s1.count1);
```

```
System.out.println(s2.count1);
System.out.println(s3.count1);
System.out.println(s4.count1);
```

```
System.out.println(s1.count2);
System.out.println(s2.count2);
System.out.println(s3.count2);
System.out.println(s4.count2);
```

What is a static class in java ?

- A static class is a class that is created within a class
- When a class is required for only a particular class and it should not be used by other class then the class can be declared as a static class and this static class is not visible to the outer world
- Example

```
class StaticTest {
    public static int count1 = 0;
    public int count2 = 0;

    Static Test() {
        count1++;
        count2++;
    }

    static class StaticDemo{

        public void test(StaticTest stat) {
            System.out.println("Demo" + stat.count2);
        }
    }
}
```

```
StaticTest.StaticDemo s = new StaticTest.StaticDemo();
s.test(new StaticTest());
```

Here is a curated list of **100 tricky interview questions** that might be asked in **Zoho** for a **Java developer position**, along with their respective **answers**:

Core Java

1. **What is the difference between `==` and `equals()` in Java?**

- `==` compares references, while `equals()` compares the actual content of objects.

2. **What is the difference between a `HashMap` and a `Hashtable`?**

- `HashMap` is non-synchronized and allows one null key and multiple null values. `Hashtable` is synchronized and doesn't allow null keys or values.

3. **Explain the difference between an `ArrayList` and a `LinkedList`.**

- `ArrayList` uses a dynamic array, allowing fast random access but slow insertions and deletions. `LinkedList` uses a doubly linked list, allowing faster insertions and deletions but slower access.

4. **How does the `HashMap` work internally in Java?**

- It uses an array of buckets. The key's hash code determines the bucket index, and a linked list is used for collision handling.

5. **What are `static` blocks and `static` methods in Java?**

- `static` blocks are executed when the class is loaded. `static` methods belong to the class rather than instances of the class and can be called without creating an instance.

6. **How does the Java `Garbage Collector` work?**

- The garbage collector automatically reclaims memory by removing objects that are no longer referenced, using algorithms like Mark and Sweep or Generational GC.

7. **What is the difference between `throw` and `throws`?**

- `throw` is used to explicitly throw an exception, while `throws` declares an exception that can be thrown by a method.

8. **Can we override a private method in Java?**

- No, private methods are not visible to subclasses and cannot be overridden.

9. **What is the significance of the `final` keyword in Java?**

- `final` can be applied to variables (constant), methods (cannot be overridden), and classes (cannot be subclassed).

10. ****How are objects passed in Java: by reference or by value?****

- Java passes objects by value, but the value is the reference to the object, making it seem like pass-by-reference.

OOP Concepts

11. ****What is the difference between `abstraction` and `encapsulation`?****

- Abstraction hides complexity by showing only relevant information. Encapsulation hides data implementation by restricting access to it using access modifiers.

12. ****Explain the four pillars of OOP (Object-Oriented Programming).****

- Encapsulation, Inheritance, Polymorphism, and Abstraction.

13. ****What is the use of the `super` keyword in Java?****

- `super` is used to refer to the immediate parent class object, methods, and constructor.

14. ****What is `method overloading` and `method overriding` in Java?****

- Overloading allows methods with the same name but different parameters. Overriding allows a subclass to provide a specific implementation of a method already defined in its parent class.

15. ****Can we inherit constructors in Java?****

- No, constructors are not inherited in Java, but you can call a superclass constructor using `super()`.

16. ****What is an `interface` in Java?****

- An interface is a blueprint of a class containing static constants and abstract methods.

17. ****What is `multiple inheritance`? How is it achieved in Java?****

- Multiple inheritance is when a class inherits from more than one class. In Java, this is achieved through interfaces.

18. ****Can we instantiate an abstract class in Java?****

- No, abstract classes cannot be instantiated.

19. ****What is the `diamond problem`? How does Java handle it?****

- The diamond problem occurs when a class inherits from two classes with the same method signature. Java handles this using interfaces, not allowing multiple inheritance with classes.

20. ****Explain the difference between `IS-A` and `HAS-A` relationships.****
- `IS-A` represents inheritance, while `HAS-A` represents composition.

Advanced Java

21. ****What is the difference between `Callable` and `Runnable`?****
- `Callable` returns a value and can throw checked exceptions, whereas `Runnable` does not return a value.
22. ****What is the `volatile` keyword in Java?****
- `volatile` ensures visibility of changes to variables across threads.
23. ****What are `synchronized` blocks in Java?****
- Synchronized blocks control access to critical sections of code by multiple threads, ensuring only one thread can access a block at a time.
24. ****What is a `ThreadLocal` in Java?****
- `ThreadLocal` provides a separate copy of a variable for each thread accessing it.
25. ****How does the `ExecutorService` work in Java?****
- `ExecutorService` manages a pool of threads to execute tasks asynchronously.
26. ****What is the `Fork/Join` framework in Java?****
- It is a framework that helps parallelize tasks by breaking them into smaller subtasks (fork) and then combining the results (join).
27. ****What is the `reflection` API in Java?****
- It allows inspection and manipulation of classes, methods, and fields at runtime.
28. ****What is `serialization` and `deserialization` in Java?****
- Serialization is the process of converting an object into a byte stream. Deserialization is the reverse process.
29. ****What are `functional interfaces` in Java?****
- Functional interfaces have exactly one abstract method and are often used with lambda expressions (e.g., `Runnable`, `Callable`).
30. ****Explain `Lambda Expressions` in Java.****
- Lambda expressions provide a clear and concise way to implement functional interfaces using an expression rather than anonymous classes.

Collections

31. **What is the difference between `ArrayList` and `Vector`?**
- `ArrayList` is non-synchronized, while `Vector` is synchronized. `ArrayList` grows by 50% when needed, and `Vector` doubles its size.
32. **What is the purpose of the `Comparator` interface in Java?**
- `Comparator` is used to define multiple sorting sequences for objects.
33. **What is the difference between `Iterator` and `ListIterator`?**
- `Iterator` traverses in one direction (forward), while `ListIterator` can traverse in both directions and modify elements.
34. **Explain the internal working of a `TreeMap` in Java.**
- `TreeMap` stores entries in a Red-Black Tree, ensuring that keys are sorted in natural or custom order.
35. **What is the difference between `fail-fast` and `fail-safe` iterators?**
- `fail-fast` iterators throw `ConcurrentModificationException` if the collection is modified during iteration. `fail-safe` iterators do not throw such exceptions as they operate on a copy of the collection.
36. **How does `ConcurrentHashMap` work?**
- `ConcurrentHashMap` allows concurrent read and write operations by dividing the map into segments to reduce lock contention.
37. **What is the difference between `HashSet` and `TreeSet`?**
- `HashSet` is unordered and based on hashing, while `TreeSet` is ordered and based on a Red-Black Tree.
38. **What is the load factor in `HashMap`?**
- The load factor determines when the map's capacity should be increased to maintain a balance between space and time complexity.
39. **How do you sort a list of custom objects in Java?**
- By implementing `Comparable` or using a `Comparator`.
40. **What is the difference between `Arrays.asList()` and `Collections.singletonList()`?**
- `Arrays.asList()` returns a fixed-size list, while `Collections.singletonList()` returns an immutable list with only one element.

Exception Handling

41. ****What is a `checked` exception in Java?****

- Checked exceptions are exceptions that are checked at compile-time.

42. ****Can you catch multiple exceptions in a single `catch` block in Java?****

- Yes, starting from Java 7, multiple exceptions can be handled in a single `catch` block using the pipe (`|`) symbol.

43. ****What happens if a `finally` block contains a return statement?****

- If a `finally` block contains a return statement, it will override any previous return statements in the `try` or `catch` block.

44. ****What is `try-with-resources` in Java?****

- `try-with-resources` is used to automatically close resources like files or database connections after use.

45. ****What are custom exceptions in Java?****

- Custom exceptions are user-defined exceptions created by extending `Exception` or `RuntimeException`.

46. ****What is the difference between `throw` and `throwable`?**

- `throw` is used to explicitly throw an exception. `Throwable` is the superclass of all exceptions and errors in Java.

47. ****What is `stack trace` in Java?**

- A stack trace is a report that provides details of method calls that were active at a certain point in the program's execution, often during an exception.

48. ****What happens if the exception is not handled in Java?**

- If an exception is not caught, it will propagate up the method call stack and eventually be handled by the JVM, causing the program to terminate.

49. ****Can you handle errors in Java?**

- Errors, which are critical conditions (like `OutOfMemory`

Here's a continuation of the list from question 50 onwards to complete the 100 tricky Java interview questions for Zoho, along with answers:

Exception Handling (continued)

50. ****Can you handle errors in Java?**

- Errors, such as `OutOfMemoryError`, represent serious issues that are typically beyond recovery. However, you can catch them, but handling errors is not usually recommended.

51. **What is the difference between `final`, `finally`, and `finalize` in Java?**

- `final` is used for constants, methods, and classes that cannot be changed. `finally` is a block that is executed after a `try-catch` block. `finalize()` is a method used by the garbage collector before an object is destroyed.

52. **What is a suppressed exception in Java?**

- In a `try-with-resources` statement, if an exception is thrown in the `try` block and another exception is thrown when closing the resource, the second exception is suppressed and can be accessed via the `getSuppressed()` method.

Threads & Concurrency

53. **What is the difference between a process and a thread?**

- A process is an independent executing program, while a thread is a smaller execution unit within a process.

54. **How can we create a thread in Java?**

- Threads can be created either by extending the `Thread` class or by implementing the `Runnable` interface.

55. **What is synchronization in Java?**

- Synchronization is the process that allows multiple threads to access shared resources without conflicting, using the `synchronized` keyword.

56. **What is a deadlock in Java?**

- Deadlock occurs when two or more threads are blocked forever, each waiting on the other to release a resource.

57. **What is a race condition in Java?**

- A race condition occurs when two or more threads access shared data and try to change it simultaneously, leading to unpredictable results.

58. **What is the difference between `wait()` and `sleep()` in Java?**

- `wait()` is used for thread communication and releases the lock, while `sleep()` pauses the thread for a specified time without releasing the lock.

59. **What is `Thread.join()` in Java?**

- `Thread.join()` ensures that the thread on which `join` is called completes its execution before the calling thread resumes.

60. **What is `Thread.yield()` in Java?**

- `yield()` temporarily pauses the currently executing thread, giving other threads a chance to execute.

61. **What is `Thread Priority` in Java?**

- Threads can have priorities ranging from 1 (`MIN_PRIORITY`) to 10 (`MAX_PRIORITY`), indicating which threads should be preferred for execution when multiple threads are in contention.

62. **What are `daemon threads` in Java?**

- Daemon threads are background threads that do not prevent the JVM from exiting when the program finishes execution.

63. **What is `Thread Pool` in Java?**

- A thread pool reuses a fixed number of threads to execute multiple tasks, improving performance by avoiding thread creation overhead.

64. **What is the difference between synchronized and concurrent collections?**

- Synchronized collections are generally slower and block the entire collection for operations, while concurrent collections like `ConcurrentHashMap` use finer-grained locking mechanisms for better performance.

65. **What is `Executor Framework` in Java?**

- The `Executor` framework provides a mechanism to manage a pool of threads to execute tasks asynchronously, handling thread lifecycle and task execution.

66. **What is `Callable` and `Future` in Java?**

- `Callable` is similar to `Runnable` but can return a result and throw a checked exception. `Future` represents the result of an asynchronous computation, providing methods to check if the computation is complete or to cancel it.

67. **What is `ReentrantLock` in Java?**

- `ReentrantLock` is a lock class in `java.util.concurrent` that provides explicit locking, allowing for more flexible thread synchronization compared to the `synchronized` keyword.

68. **What is `CountDownLatch` in Java?**

- `CountDownLatch` is used to block one or more threads until a set of operations being performed by other threads is completed.

69. **What is `CyclicBarrier` in Java?**

- `CyclicBarrier` is used to synchronize multiple threads at a common point, allowing them to wait until all threads reach that point.

70. **What is `Semaphore` in Java?**

- `Semaphore` is used to control access to a resource by multiple threads, limiting the number of threads that can access the resource simultaneously.

Java 8 Features

71. **What is a `default method` in an interface?**

- A default method in an interface is a method with a body, which allows developers to add new functionality to interfaces without breaking existing implementations.

72. **What is a `functional interface`?**

- A functional interface is an interface with exactly one abstract method, used as the basis for lambda expressions and method references.

73. **What are `streams` in Java 8?**

- Streams are a sequence of objects that support functional-style operations for processing collections and arrays, such as filtering, mapping, and reducing.

74. **What is the `difference between Stream and Collection`?**

- A `Collection` is a data structure that holds data, while a `Stream` is a pipeline that processes data from a source.

75. **What are `Optional` in Java 8?**

- `Optional` is a container object that may or may not contain a non-null value, used to avoid `NullPointerException`.

76. **What is `method reference` in Java?**

- Method reference is a shorthand syntax for calling a method using the `::` operator, typically used with lambda expressions.

77. **Explain `java.time` API in Java 8.**

- `java.time` API introduces new classes like `LocalDate`, `LocalTime`, and `ZonedDateTime` for more robust date and time handling, replacing the old `Date` and `Calendar` classes.

78. **What is `Predicate` in Java 8?**

- `Predicate` is a functional interface representing a boolean-valued function, used for filtering in streams.

79. **What is the use of `forEach()` method in Java 8?**

- `forEach()` is used to iterate over elements in a collection or stream and perform an action on each element.

80. **What is the difference between `map()` and `flatMap()` in Java Streams?**

- `map()` is used for transforming elements, while `flatMap()` is used for flattening nested structures into a single stream.

Memory Management & Garbage Collection

81. **What is the `heap memory` and `stack memory` in Java?**

- Heap memory is used for dynamic allocation of objects, while stack memory stores method calls and local variables.

82. **What is a `memory leak` in Java?**

- A memory leak occurs when an application holds references to objects that are no longer needed, preventing them from being garbage collected.

83. **What are the different `garbage collection algorithms` in Java?**

- The most common algorithms are Mark and Sweep, Generational Garbage Collection, and G1 Garbage Collector.

84. **What is the difference between `strong`, `weak`, `soft`, and `phantom references` in Java?**

- Strong references prevent garbage collection, weak references allow GC when the object is weakly referenced, soft references allow GC but try to hold the object longer, and phantom references are used to track when an object has been finalized and ready for GC.

85. **What is the `permgen` and `metaspace` in Java?**

- `PermGen` (Permanent Generation) was used to store class metadata before Java 8. In Java 8, `PermGen` was replaced by `Metaspace`, which grows dynamically.

86. **Explain the difference between Minor GC and Major GC?**

- Minor GC cleans the young generation of memory (Eden and Survivor spaces), while Major GC cleans the entire heap, including the old generation.

87. **What is `OutOfMemoryError` in Java?**

- `OutOfMemoryError` occurs when the JVM cannot allocate more memory for objects or classes due to insufficient memory.

88. **How to manually trigger garbage collection in Java?**

- Garbage collection can be suggested using `System.gc()`, but the JVM is free to ignore this call.

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

- JVM (Java Virtual Machine) executes Java bytecode. JRE (Java Runtime Environment) provides the libraries and JVM. JDK (Java Development Kit) includes the JRE and development tools.

Design Patterns & Best Practices

90. **What is the singleton pattern in Java?**

- Singleton ensures that a class has only one instance and provides a global access point to it.

91. **What is the factory pattern in Java?**

- Factory pattern provides an interface for creating objects but allows subclasses to alter the type of objects that will be created.

92. **Explain the observer pattern in Java.**

- The observer pattern defines a one-to-many relationship between objects where one object (the subject) notifies multiple observers of any changes.

93. **What is the builder pattern in Java?**

- Builder pattern is used to construct complex objects step by step, making the code more readable and flexible.

Sure, here's the continuation of the list:

94. **What is the builder pattern in Java?**

- The builder pattern is used to create complex objects step by step, making the construction process clear and more flexible. It helps in constructing immutable objects and handling multiple optional parameters.

95. **Explain the proxy pattern in Java.**

- The proxy pattern provides a placeholder or surrogate for another object to control access to it. Proxies can be used for lazy initialization, access control, logging, and more.

96. **What is the `strategy pattern` in Java?**

- The strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. It allows a method to choose an algorithm at runtime.

97. **What is the `decorator pattern` in Java?**

- The decorator pattern allows behavior to be added to individual objects dynamically, without affecting other objects of the same class, typically using a wrapper.

98. **Explain the `adapter pattern` in Java.**

- The adapter pattern allows two incompatible interfaces to work together. It acts as a bridge between two objects.

99. **What is the `command pattern` in Java?**

- The command pattern encapsulates a request as an object, thereby allowing for parameterization of clients with different requests, queuing of requests, and logging the request operations.

100. **What is the `observer pattern` in Java?**

- The observer pattern defines a one-to-many relationship between objects where one object (the subject) notifies multiple observers of any changes in its state.

This completes the list of 100 tricky Java interview questions along with their answers that can help you prepare for a Java developer interview at **Zoho**. If you want more specific questions or deeper explanations on any of the topics, feel free to ask!