**Core java Interview Questions**

1. [How to create a immutable object in Java? Count all benefits?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques1)
   * **Immutable objects** are instances whose state doesn’t change after it has been initialized.

For example, String is an immutable class and once instantiated its value never changes.

* + Benefits:
    1. It is thread safe
    2. An immutable class is good for caching purposes because you don’t have to worry about the value changes
  + Process to create immutable object:
    1. Declare class as Final [ so it cannot be extended]
    2. Make all field as private [ to restrict direct access]
    3. Remove setter methods if any [ as state of immutable object cannot get changed]
    4. Make all mutable fields final [ so value can be assigned only once]
    5. Initialize all fields with deep copy [ i.e. instead of copying reference use deep copy]
    6. In getter method return copy of object rather than directly returning object

[ if we return object directly, the value can be overridden since reference would have been pointing to same memory location]

1. [What is the use of the finally block? Is finally block in Java guaranteed to be called? When finally block is NOT called?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques3)
   * Finally block is used for operations like closing resources, closing files, freeing up threads etc. since it is called in all the cases.
   * Only in case of some scenarios the finally block cannot be called\_
     1. If JVM crashes as it runs out of memory, and complete java process get killed from system
     2. If System.exit(int) get called before finally block get called.
2. [Why main() in java is declared as public static void main?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques6)
   * Public: to make it accessible to external world
   * Static: as main method is an entry point, it can not be instance method (since instance will not be available)
   * void: will not return any value
   * main: name of the method (can be anything)
3. [What is the difference between creating String as new() and literal?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques7)
   * **When we create a**String**object using the**new()**operator, it always creates a new object in heap memory.**
   * **On the other hand, if we create an object using**String**literal syntax e.g. “Baeldung”, it may return an existing object from the String pool, if it already exists.** Otherwise, it will create a new String object and put in the string pool for future re-use.
   * At a high level, both are the String objects, but the main difference comes from the point that new() operator always creates a new String object. Also, when we create a String using literal – it is interned.
4. [How does subString() inside String works?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques8)
   * subString is an overloaded method present in String class
   * subString(int startIndex); or subString(int startIndex,int endIndex); can throw IndexOutOfBound exception
   * from Java7 onwords, it uses character Sequence to create new String object in heap or String pool and can be referred in by new String
5. [Difference between interfaces and abstract classes?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques10)

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) as Abstract class is a class multiple inheritance is not allowed | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables** |
| 4) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 7) An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| 8) A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |

1. [When do you override hashCode and equals()?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-1/#ques11)
   * When we want to change the logic of equals we can override equals method of Object class.
   * The default behavior of equals method is it return true if hashCode of two objects are equal, hence as per contract between equals and hashCode method, we must override hashCode method whenever we override equals method.
2. [Explain abstraction and encapsulation? How are they related?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#abstraction-encalsulation)
   * Abstraction and Encapsulation are pillars of Object oriented programming
   * Where Abstraction means showing only essential details where as Encapsulation means binding data and related properties together.
   * For example; in a class Car
     1. There can be methods like Car can goForward,gobackword,stop,horn,takeLeft etc.
     2. But for taking turns, internally we are changing value of direction,
     3. So as a user, goForword(int value) method will get called but internally it is changing the variable direction.
     4. Since as end user the essential method is goForward() will be called, but since since because of encapsulation the value of direction variable getting changed
3. [How StringBuffer save the memory?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#stringbuffer-save-memory)
   * String buffer is a mutable version of String class
   * As String class is immutable whenever we create any new object using String, new Object reference is added in the memory. Which increase memory consumption.
   * On the other hand, if we use StringBuffer it is immutable and thread safe, also there are methods available in StringBuffer class to modify the String values , so instead of creating new object it works on same object hence it saves memory.
4. [Explain transient and volatile keywords in java?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#transient-volatile)
   * A volatile keyword is used in a multithreading environment where two threads reading and writing the same variable simultaneously.
   * The volatile keyword flushes the changes directly to the main memory instead of the CPU cache.
   * On the other hand, the transient keyword is used during serialization.
   * Fields that are marked as transient cannot be part of the serialization and deserialization.
   * If we don't want to save the value of any variable then we use transient keyword with that variable.

| **Sr. No.** | **Key** | **Volatile** | **Transient** |
| --- | --- | --- | --- |
| 1 | Basic | Volatile keyword is used to flush changes directly to the main memory | The transient keyword is used to exclude variable during serialization |
| 2. | Default value | Volatile are not initialized with a default value. | During deserialization, transient variables are initialized with a default value |
| 3 | Static | Volatile can be used with a static variable. | Transient cannot be used with the static keyword |
| 4 | Final | Volatile can be used with the final keyword | Transient can not be used with the final keyword |

1. [Difference between Iterator and ListIterator?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#iterator-vs-listiterator)
   * Iterator can traverse only in forward direction whereas ListIterator traverses both in forward and backward directions.
     1. Iterator has only next() method
     2. whereas listIterator has next() and previous() methods
   * ListIterator can help to replace an element whereas Iterator cannot.
     1. Set() method present in ListIterator is used to set value while iterating
   * We can use iterator with Map, List and Set, whereas listIterator can only be used with List
   * We can find index using nextIndex() and previousIndex() methods in ListIterator, these methods are not available in iterator
   * Methods in Iterator interface : next(), remove() and hasNext() etc.
   * Methods in ListIterator interface : next(), previous(), hasNext(), hasPrevious(), add(E e) etc.
2. [Why finalize() method should be avoided?](https://howtodoinjava.com/interview-questions/core-java-interview-questions-series-part-2/#why-avoid-finalize)
   * Finalize method is called by the garbage collector on an object when it determines that there are no more references to the object.
   * A subclass overrides the finalize method to dispose of system resources or to perform other cleanup.
   * But execution of finalize() method is not guaranteed.
   * Also, exception thrown by finalize() methods will not be caught by GC
   * Finalize method doesn’t support method chaining, i.e. if you want to use finalize() method, you must call finalize method of parent class in child class finalize() method.

Deep copy and shallow copy?

Explain all access modifiers?

What is garbage collection? Can we enforce it?

What is native keyword?

What if the difference between && and &??

[How to create an instance of any class without using new keyword](https://howtodoinjava.com/puzzles/how-to-create-an-instance-of-any-class-without-using-new-keyword/)

Is String thread-safe in Java

How do you count the number of occurrences of each character in a string?

25. Write a java program to reverse a string?

Difference between String, StringBuffer and StringBuilder?

Memory leak issue in String class

Why are strings immutable?

3. What is String constant pool?

What is an exception in Java?

How does exception handling work in Java?

What are exception handling keywords in Java?

What is the purpose of the throw and throws keywords?

How can you handle an exception?

Explain the Java exception hierarchy.

How can you catch multiple exceptions?

What is the difference between checked and unchecked exceptions in Java?

What is the difference between throw and throws keyword in Java?

What is the difference between an exception and error?

What is the OutOfMemoryError in Java?

What are chained exceptions in Java?

How do you write a custom exception in Java?

What is the difference between final, finally, and finalize in Java?

What happens when an exception is thrown by the main method?

What is a try-with-resources statement?

What is a stacktrace and how does it relate to an exception?

What are the advantages of Java exceptions?

Can you throw any exception inside a lambda expression’s body?

What are the rules that we need to follow when overriding a method that throws an exception?

What are some of the exception handling best practices?