Java 8 Interview Questions and Answers

These **Java 8 interview questions and answers** have been designed to get you acquainted with the nature of questions you may encounter during your technical interview for Java programming. Let's continue with knowing about Java 8.

What is Java 8?

Java 8 is a major feature release of Java programming language development. Its initial version was released on 18 March 2014. With the Java 8 release, Java provided support for functional programming, a new JavaScript engine, new APIs for date time manipulation, new streaming API, etc.

Java 8 Interview Questions and Answers

These **interview questions on Java 8** cover advanced topics that have been added in the new version (i.e., Java 8). Go through these questions to learn and enhance your knowledge of the topics that have been added in Java 8 such as Lambda expression, method reference, default methods, stream API, and many more.

Prerequisite: 200+ Java Interview Questions and Answers

Java 8 Interview Questions - Basic Level

Some of the basic level Java 8 interview questions along with their answers are as follows:

1. What are the new features introduced in JAVA 8?

There are dozens of features added to Java 8, the most significant ones are mentioned below –

- **Lambda expression** – Adds functional processing capability to Java.
- **Method references** – Referencing functions by their names instead of invoking them directly. Using functions as parameter.
- **Default method** – Interface to have default method implementation.
- **New tools** – New compiler tools and utilities are added like 'jdeps' to figure out dependencies.
- **Stream API** – New stream API to facilitate pipeline processing.

- **Date Time API** – Improved date time API.
- **Optional** – Emphasis on best practices to handle null values properly.
- **Nashorn, JavaScript Engine** – A Java-based engine to execute JavaScript code.

Along with these new featuers, lots of feature enhancements are done under-the-hood, at both compiler and JVM level.

2. How will you sort a list of string using Java 8 lambda expression?

Following code sorts a list of string using Java 8 lambda expression:

```
//sort using java 8
private void sortUsingJava8(List<String> names) {
  Collections.sort(names, (s1, s2) -> s1.compareTo(s2));
}
```

3. What are the characteristics of a Java 8 lambda expression?

A lambda expression is characterized by the following syntax -

```
parameter -> expression body
```

Following are the important characteristics of a lambda expression –

- **Optional type declaration** – No need to declare the type of a parameter. The compiler can inference the same from the value of the parameter.
- **Optional parenthesis around parameter** – No need to declare a single parameter in parenthesis. For multiple parameters, parentheses are required.
- **Optional curly braces** – No need to use curly braces in expression body if the body contains a single statement.
- **Optional return keyword** – The compiler automatically returns the value if the body has a single expression to return the value. Curly braces are required to indicate that expression returns a value.

4. Why lambda expression is to be used?

Lambda expressions are used primarily to define inline implementation of a functional interface, i.e., an interface with a single method only. In the above example, we've used various types of lambda expressions to define the operation method of MathOperation interface. Then we have defined the implementation of sayMessage of GreetingService.

Lambda expression eliminates the need of anonymous class and gives a very simple yet powerful functional programming capability to Java.

5. What kind of variable you can access in an lambda expression??

Using lambda expression, you can refer to final variable or effectively final variable (which is assigned only once). Lambda expression throws a compilation error, if a variable is assigned a value the second time.

6. What are method references?

Method references help to point to methods by their names. A method reference is described using :: (double colon) symbol. A method reference can be used to point the following types of methods –

- Static methods
- Instance methods
- Constructors using new operator (TreeSet::new)

7. Explain the System.out::println expression.

System.out::println method is a static method reference to println method of out object of System class.

8. What are functional interfaces?

Functional interfaces have a single functionality to exhibit. For example, a Comparable interface with a single method 'compareTo' is used for comparison purpose. Java 8 has defined a lot of functional interfaces to be used extensively in lambda expressions.

9. What is the purpose of BiConsumer<T,U> functional interface?

It represents an operation that accepts two input arguments, and returns no result.

10. What is the purpose of BiFunction<T,U,R> functional interface?

It represents a function that accepts two arguments and produces a result.

11. What is the purpose of BinaryOperator<T> functional interface?

It represents an operation upon two operands of the same type, producing a result of the same type as the operands.

12. What is the purpose of BiPredicate<T,U> functional interface?

It represents a predicate (Boolean-valued function) of two arguments.

13. What is the purpose of BooleanSupplier functional interface?

It represents a supplier of Boolean-valued results.

14. What is the purpose of Consumer<T> functional interface?

It represents an operation that accepts a single input argument and returns no result.

15. What is the purpose of DoubleBinaryOperator functional interface?

It represents an operation upon two double-valued operands and producing a double-valued result.

16. What is the purpose of DoubleConsumer functional interface?

It represents an operation that accepts a single double-valued argument and returns no result.

17. What is the purpose of DoubleFunction<R> functional interface?

It represents a function that accepts a double-valued argument and produces a result.

18. What is the purpose of DoublePredicate functional interface?

It represents a predicate (Boolean-valued function) of one double-valued argument.

19. What is the purpose of DoubleSupplier functional interface?

It represents a supplier of double-valued results.

20. What is the purpose of DoubleToIntFunction functional interface?

It represents a function that accepts a double-valued argument and produces an int-valued result.

21. What is the purpose of DoubleToLongFunction functional interface?

It represents a function that accepts a double-valued argument and produces a long-valued result.

22. What is the purpose of DoubleUnaryOperator functional interface?

It represents an operation on a single double-valued operand that produces a double-valued result.

23. What is the purpose of Function<T,R> functional interface?

It represents a function that accepts one argument and produces a result.

24. What is the purpose of IntBinaryOperator functional interface?

It represents an operation upon two int-valued operands and produces an int-valued result.

25. What is the purpose of IntConsumer functional interface?

It represents an operation that accepts a single int-valued argument and returns no result.

26. What is the purpose of IntFunction<R> functional interface?

It represents a function that accepts an int-valued argument and produces a result.

27. What is the purpose of IntPredicate functional interface?

It represents a predicate (Boolean-valued function) of one int-valued argument.

28. What is the purpose of IntSupplier functional interface?

It represents a supplier of int-valued results.

29. What is the purpose of IntToDoubleFunction functional interface?

It represents a function that accepts an int-valued argument and produces a double-valued result.

30. What is the purpose of IntToLongFunction functional interface?

It represents a function that accepts an int-valued argument and produces a long-valued result.

31. What is the purpose of IntUnaryOperator functional interface?

It represents an operation on a single int-valued operand that produces an int-valued result.

32. What is the purpose of LongBinaryOperator functional interface?

It represents an operation upon two long-valued operands and produces a long-valued result.

33. What is the purpose of LongConsumer functional interface?

It represents an operation that accepts a single long-valued argument and returns no result.

34. What is the purpose of LongFunction<R> functional interface?

It represents a function that accepts a long-valued argument and produces a result.

35. What is the purpose of LongPredicate functional interface?

It represents a predicate (Boolean-valued function) of one long-valued argument.

36. What is the purpose of LongSupplier functional interface?

It represents a supplier of long-valued results.

37. What is the purpose of LongToDoubleFunction functional interface?

It represents a function that accepts a long-valued argument and produces a double-valued result.

38. What is the purpose of LongToIntFunction functional interface?

It represents a function that accepts a long-valued argument and produces an int-valued result.

39. What is the purpose of LongUnaryOperator functional interface?

It represents an operation on a single long-valued operand that produces a long-valued result.

40. What is the purpose of ObjDoubleConsumer<T> functional interface?

It represents an operation that accepts an object-valued and a double-valued argument, and returns no result.

41. What is the purpose of ObjIntConsumer<T> functional interface?

It represents an operation that accepts an object-valued and an int-valued argument, and returns no result.

42. What is the purpose of ObjLongConsumer<T> functional interface?

It represents an operation that accepts an object-valued and a long-valued argument, and returns no result.

43. What is the purpose of Predicate<T> functional interface?

It represents a predicate (Boolean-valued function) of one argument.

44. What is the purpose of Supplier<T> functional interface?

It represents a supplier of results.

45. What is the purpose of ToDoubleBiFunction<T,U> functional interface?

It represents a function that accepts two arguments and produces a double-valued result.

46. What is the purpose of ToDoubleFunction<T> functional interface?

It represents a function that produces a double-valued result.

47. What is the purpose of ToIntBiFunction<T,U> functional interface?

It represents a function that accepts two arguments and produces an int-valued result.

48. What is the purpose of ToIntFunction<T> functional interface?

It represents a function that produces an int-valued result.

49. What is the purpose of ToLongBiFunction<T,U> functional interface?

It represents a function that accepts two arguments and produces a long-valued result.

50. What is the purpose of ToLongFunction<T> functional interface?

It represents a function that produces a long-valued result.

Java 8 Interview Questions - Experienced Level

Some of the advance (experienced) level of Java 8 interview questions along with their answers are as follow:

51. What is the purpose of UnaryOperator<T> functional interface?

It represents an operation on a single operand that produces a result of the same type as its operand.

52. What are default methods?

With java 8, an interface can have default implementation of a function in interfaces.

53. What are static default methods?

An interface can also have static helper methods from Java 8 onwards.

```
public interface vehicle {
   default void print() {
      System.out.println("I am a vehicle!");
   }

   static void blowHorn() {
      System.out.println("Blowing horn!!!");
   }
}
```

54. How will you call a default method of an interface in a class?

Using super keyword along with interface name.

```
interface Vehicle {
   default void print() {
      System.out.println("I am a vehicle!");
   }
}
class Car implements Vehicle {
   public void print() {
      Vehicle.super.print();
   }
}
```

55. How will you call a static method of an interface in a class?

Using name of the interface.

```
interface Vehicle {
   static void blowHorn() {
      System.out.println("Blowing horn!!!");
   }
}
class Car implements Vehicle {
   public void print() {
      Vehicle.blowHorn();
   }
}
```

56. What is streams in Java 8?

Stream represents a sequence of objects from a source, which supports aggregate operations.

57. What is stream pipelining in Java 8?

Most of the stream operations return stream itself so that their result can be pipelined. These operations are called intermediate operations and their function is to take input, process them, and return output to the target. collect() method is a terminal operation which is normally present at the end of the pipelining operation to mark the end of the stream.

58. What is the difference between Collections and Stream in Java8 ?

Stream operations do the iterations internally over the source elements provided, in contrast to Collections where explicit iteration is required.

59. What is the purpose of forEach method of stream in java 8?

Stream has provided a new method 'forEach' to iterate each element of the stream.

60. How will you print 10 random numbers using forEach of java 8?

The following code segment shows how to print 10 random numbers using forEach.

```
Random random = new Random();
random.ints().limit(10).forEach(System.out::println);
```

61. What is the purpose of map method of stream in java 8?

The 'map' method is used to map each element to its corresponding result.

62. How will you print unique squares of numbers in java 8?

The following code segment prints unique squares of numbers using map.

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
//get list of unique squares
List<Integer> squaresList = numbers.stream().map( i -> i*i).distinct().coll
```

63. What is the purpose of filter method of stream in java 8?

The 'filter' method is used to eliminate elements based on a criteria.

64. How will you print count of empty strings in java 8?

The following code segment prints a count of empty strings using filter.

```
List<String>strings = Arrays.asList("abc", "", "bc", "efg", "abcd","", "jkl
//get count of empty string
int count = strings.stream().filter(string -> string.isEmpty()).count();
```

65. What is the purpose of limit method of stream in java 8?

The 'limit' method is used to reduce the size of the stream.

66. How will you print 10 random numbers in java 8?

The following code segment shows how to print 10 random numbers.

```
Random random = new Random();
random.ints().limit(10).forEach(System.out::println);
```

67. What is the purpose of sorted method of stream in java 8?

The 'sorted' method is used to sort the stream.

68. How will you print 10 random numbers in a sorted order in java 8?

The following code segment shows how to print 10 random numbers in a sorted order.

```
Random random = new Random();
random.ints().limit(10).sorted().forEach(System.out::println);
```

69. What is Parallel Processing in Java 8?

parallelStream is the alternative of stream for parallel processing. Take a look at the following code segment that prints a count of empty strings using parallelStream.

```
List<String> strings = Arrays.asList("abc", "", "bc", "efg", "abcd","", "jk
//get count of empty string
int count = strings.parallelStream().filter(string -> string.isEmpty()).cou
//It is very easy to switch between sequential and parallel streams.
```

70. What are collectors in Java 8?

Collectors are used to combine the result of processing on the elements of a stream. Collectors can be used to return a list or a string.

```
List<String>strings = Arrays.asList("abc", "", "bc", "efg", "abcd","", "jkl
List<String> filtered = strings.stream().filter(string -> !string.isEmpty()
System.out.println("Filtered List: " + filtered);
String mergedString = strings.stream().filter(string -> !string.isEmpty()).
System.out.println("Merged String: " + mergedString);
```

71. What are Statistics collectors in Java 8?

With Java 8, statistics collectors are introduced to calculate all statistics when stream processing is being done.

72. How will you get the highest number present in a list using Java 8?

Following code will print the highest number present in a list.

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
IntSummaryStatistics stats = integers.stream().mapToInt((x) -> x).summarySt
System.out.println("Highest number in List : " + stats.getMax());
```

73. How will you get the lowest number present in a list using Java 8?

Following code will print the highest number present in a list.

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
IntSummaryStatistics stats = integers.stream().mapToInt((x) -> x).summarySt
System.out.println("Lowest number in List : " + stats.getMin());
```

74. How will you get the sum of all numbers present in a list using Java 8?

Following code will print the sum of all numbers present in a list.

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
IntSummaryStatistics stats = integers.stream().mapToInt((x) -> x).summarySt
System.out.println("Sum of all numbers : " + stats.getSum());
```

75. How will you get the average of all numbers present in a list using Java 8?

Following code will print the average of all numbers present in a list.

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
IntSummaryStatistics stats = integers.stream().mapToInt((x) -> x).summarySt
```

```
System.out.println("Average of all numbers : " + stats.getAverage());
```

76. What is Optional in Java8?

Optional is a container object which is used to contain not-null objects. Optional object is used to represent null with absent value. This class has various utility methods to facilitate code to handle values as 'available' or 'not available' instead of checking null values. It is introduced in Java 8 and is similar to what Optional is in Guava.

77. What is Nashorn in Java8?

With Java 8, Nashorn, a much improved javascript engine is introduced, to replace the existing Rhino. Nashorn provides 2 to 10 times better performance, as it directly compiles the code in memory and passes the bytecode to JVM. Nashorn uses invokedynamics feature, introduced in Java 7 to improve performance.

78. What is jjs in JAVA8?

For Nashorn engine, JAVA 8 introduces a new command line tool, jjs, to execute javascript codes at console.

79. Can you execute javascript code from java 8 code base?

Yes! Using ScriptEngineManager, JavaScript code can be called and interpreted in Java.

80. What is local datetime API in JAVA8?

Local − Simplified date-time API with no complexity of timezone handling.

81. What is zoned datetime API in JAVA8?

Zoned − Specialized date-time API to deal with various timezones.

82. What is chromounits in java8?

java.time.temporal.ChronoUnit enum is added in Java 8 to replace the integer values used in old API to represent day, month, etc.

83. How will you get the current date using local datetime api of java8?

Following code gets the current date using local datetime api −

```
//Get the current date
LocalDate today = LocalDate.now();
System.out.println("Current date: " + today);
```

84. How will you add 1 week to current date using local datetime api of java8?

Following code adds 1 week to current date using local datetime api –

```
//add 1 week to the current date
LocalDate today = LocalDate.now();
LocalDate nextWeek = today.plus(1, ChronoUnit.WEEKS);
System.out.println("Next week: " + nextWeek);
```

85. How will you add 1 month to current date using local datetime api of java8?

Following code adds 1 month to current date using local datetime api:

```
//add 1 month to the current date
LocalDate today = LocalDate.now();
LocalDate nextMonth = today.plus(1, ChronoUnit.MONTHS);
System.out.println("Next month: " + nextMonth);
```

86. How will you add 1 year to current date using local datetime api of java8?

Following code adds 1 year to current date using local datetime api –

```
//add 1 year to the current date
LocalDate today = LocalDate.now();
LocalDate nextYear = today.plus(1, ChronoUnit.YEARS);
System.out.println("Next year: " + nextYear);
```

87. How will you add 10 years to current date using local datetime api of java8?

Following code adds 10 years to current date using local datetime api –

```
//add 10 years to the current date
LocalDate today = LocalDate.now();
LocalDate nextDecade = today.plus(1, ChronoUnit.DECADES);
System.out.println("Date after ten year: " + nextDecade);
```

88. How will you get next tuesday using java8?

Following code gets next tuesday using java8 −

```
//get the next tuesday
LocalDate today = LocalDate.now();
LocalDate nextTuesday = today.with(TemporalAdjusters.next(DayOfWeek.TUESDAY
System.out.println("Next Tuesday on : " + nextTuesday);
```

89. How will you get second saturday of next month using java8?

Following code gets second saturday of next month using java8 −

```
//get the second saturday of next month
LocalDate firstInYear = LocalDate.of(date1.getYear(),date1.getMonth(), 1);
LocalDate secondSaturday = firstInYear.with(TemporalAdjusters.nextOrSame(Da
System.out.println("Second Saturday on : " + secondSaturday);
```

90. How will you get the instant of current date in terms of milliseconds using java8?

Following code gets the instant of current date in terms of milliseconds −

```
//Get the instant of current date in terms of milliseconds
Instant now = currentDate.toInstant();
```

91. How will you get the instant of local date time using time in of milliseconds using java8?

Following code gets the instant of local date time using time in of milliseconds −

```
Instant now = currentDate.toInstant();
ZoneId currentZone = ZoneId.systemDefault();
```

```
LocalDateTime localDateTime = LocalDateTime.ofInstant(now, currentZone);
System.out.println("Local date: " + localDateTime);
```

92. How will you get the instant of zoned date time using time in of milliseconds using java8?

Following code gets the instant of zoned date time using time in of milliseconds –

```
Instant now = currentDate.toInstant();
ZoneId currentZone = ZoneId.systemDefault();
ZonedDateTime zonedDateTime = ZonedDateTime.ofInstant(now, currentZone);
System.out.println("Zoned date: " + zonedDateTime);
```

93. Which class implements a decoder for decoding byte data using the Base64 encoding scheme in Java8?

static class Base64.Decoder – This class implements a decoder for decoding byte data using the Base64 encoding scheme as specified in RFC 4648 and RFC 2045.

94. Which class implements an encoder for encoding byte data using the Base64 encoding scheme in Java8?

static class Base64.Encoder – This class implements an encoder for encoding byte data using the Base64 encoding scheme as specified in RFC 4648 and RFC 2045.

95. How will you create a Base64 decoder?

getDecoder() method of Base64 class returns a Base64.Decoder that decodes using the Basic type base64 encoding scheme.

96. How will you create a Base64 encoder?

getEncoder() method of Base64 class returns a Base64.Encoder that encodes using the Basic type base64 encoding scheme.

97. How will you create a Base64 decoder that decodes using the MIME type base64 encoding scheme?

getMimeDecoder() method of Base64 class returns a Base64.Decoder that decodes using the MIME type base64 decoding scheme.

98. How will you create a Base64 encoder that encodes using the MIME type base64 encoding scheme?

getMimeEncoder() method of Base64 class returns a Base64.Encoder that encodes using the MIME type base64 encoding scheme.

99. How will you create a Base64 decoder that decodes using the URL and Filename safe type base64 encoding scheme?

getUrlDecoder() method of Base64 class returns a Base64.Decoder that decodes using the URL and Filename safe type base64 encoding scheme.

100. How will you create a Base64 encoder that encodes using the URL and Filename safe type base64 encoding scheme?

getUrlEncoder() method of Base64 class returns a Base64.Encoder that encodes using the URL and Filename safe type base64 encoding scheme.

What is Next?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)