Java 8 Interview Questions|Most Asked Questions For Interview

Java 8 Interview Questions Set 1


Java
8
(x) -> x + x

Java 8 Interview Questions|Most Asked Questions For Interview

Q.1 What New Features Were Added in Java 8?

Java 8 ships with several new features but the most significant are the following:

- **Lambda Expressions** – a new language feature allowing treating actions as objects
- **Method References** – enable defining Lambda Expressions by referring to methods directly using their names
- **Optional** – special wrapper class used for expressing optionality
- **Functional Interface** – an interface with maximum one abstract method, implementation can be provided using a Lambda Expression
- **Default methods** – give us the ability to add full implementations in interfaces besides abstract methods
- **Nashorn, JavaScript Engine** – Java-based engine for executing and evaluating JavaScript code
- **Stream API** – a special iterator class that allows processing collections of objects in a functional manner
- **Date API** – an improved, immutable JodaTime-inspired Date API

Q. 2 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.
- **The 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 the** 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.

Q.3 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 the MathOperation interface. Then we have defined the implementation of sayMessage of GreetingService.
- Lambda expression eliminates the need of an anonymous class and gives a very simple yet powerful functional programming capability to Java.

Q.4 What Is a Method Reference?

A method reference is a Java 8 construct that can be used for referencing a method without invoking it. It is used for treating methods as Lambda Expressions. They only work as syntactic sugar to reduce the verbosity of some lambdas. This way, the following code:

```
(o) -> o.toString();
```

can become:

```
Object::toString();
```

A method reference can be identified by a double colon separating a class or object name and the name of the method. It has different variations such as constructor reference:

```
String::new;
```

- Static method reference:

```
String::valueOf;
```

- Bound instance method reference:

```
str::toString;
```

- Unbound instance method reference:

```
String::toString;
```

Q5 What Is Nashorn in Java8?

- Nashorn is the new Javascript processing engine for the Java platform that shipped with Java 8. Until JDK 7, the Java platform used Mozilla Rhino for the same purpose. as a Javascript processing engine.
- Nashorn provides better compliance with the ECMA normalized JavaScript specification and better runtime performance than its predecessor.
- **Q. 6 What Is JJS?**
- In Java 8, jjs is the new executable or command-line tool used to execute Javascript code at the console.

Q.7 What Is a Functional Interface? What Are the Rules of Defining a Functional Interface?

- A functional interface is an interface with no more, no less but one single abstract method (*default* methods do not count).
- Where an instance of such an interface is required, a Lambda Expression can be used instead. More formally put: *Functional interfaces* provide target types for lambda expressions and method references.
- The arguments and return type of such expression directly match those of the single abstract method.
- For instance, the *Runnable* interface is a functional interface, so instead of:

```
Thread thread = new Thread(new Runnable() {
    public void run() {
        System.out.println("Hello World!");
    }
});
```

- you could simply do:

```
Thread thread = new Thread(() -> System.out.println("Hello World!"));
```

- Functional interfaces are usually annotated with the *@FunctionalInterface* annotation – which is informative and does not affect the semantics.

- **8 What Is Stream Pipelining in Java 8?**

- Stream pipelining is the concept of chaining operations together. This is done by splitting the operations that can happen on a stream into two categories: intermediate operations and terminal operations.
- Each intermediate operation returns an instance of Stream itself when it runs, an arbitrary number of intermediate operations can, therefore, be set up to process data forming a processing pipeline.
- There must then be a terminal operation that returns a final value and terminates the pipeline.

- **9 Will the Following Code Compile?**

```
@FunctionalInterface
public interface Function2<T, U, V> {
    public V apply(T t, U u);

    default void count() {
        // increment counter
    }
}
```

- Yes. The code will compile because it follows the functional interface specification of defining only a single abstract method. The second method, *count*, is a default method that does not increase the abstract method count.
- **#10 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.

Q. 12 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);

Q.13How 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);
```

- **14 How will you print count of empty strings in java 8?**

- The following code segment prints a count of empty strings using filters.
- List<String>strings = Arrays.asList(“abc”, “”, “bc”, “efg”, “abcd”,””, “jkl”);
- //get count of empty string
- int count = strings.stream().filter(string −> string.isEmpty()).count();

- **15 How will you get the lowest number present in a list using Java 8?**
- The 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).summaryStatistics();
- System.out.println("Lowest number in List : " + stats.getMin());

- **16 How will you get the sum of all numbers present in a list using Java 8?**

- The 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).summaryStatistics();
- System.out.println("Sum of all numbers : " + stats.getSum());

Q.17 How will you get the average of all numbers present in a list using Java 8?

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

Q.18 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);
```

- **Q.19 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.

Q.20 Advantages of Java SE 8 New Features?

We can get the following benefits from Java SE 8 New Features:

- More Concise and Readable code
- More Reusable code
- More Testable and Maintainable Code
- Highly Concurrent and Highly Scalable Code
- Write Parallel Code
- Write Database Like Operations
- Better Performance Applications
- More Productive code