1. What are some of the important features that are introduced in Java 8?

There are many features important for development using Java that were implemented in Java 8. Some of these features are as follows:

- Lambda expressions
- Default methods for interfaces
- Functional interface
- Stream API
- Date API

2. What is a lambda expression?

The lambda expression was added into Java 8 to provide users with an anonymous function that can be called without using the function name but has a functional set of parameters with a lambda entity.

The following is the structure of a lambda expression:

```
(Argument List) ->{expression;} or
(Argument List) ->{statements;}
```

3. What are the most important advantages of using Java 8?

The introduction of Java 8 has been proven to be very helpful for programmers in the following ways:

- Code is now highly readable
- More reusability of code
- Code size is now very compact
- Minimum boilerplate code
- Easier testability methods
- Parallel execution and operations

4. Describe the syntax of a lambda expression.

Lambda expressions can be divided into three parts as shown in the below syntax:

```
//Lambda expression
(int a, int b) -> { System.out.println(a+b); return a+b;}
```

1. **Arguments**: A lambda expression can have zero or more arguments at any point in time.
2. **Array token**: It is used to point to the body of the expression.
3. **Body**: The body consists of expressions and statements. Braces are not required if it has only a single statement.

5. What is the use of the @FunctionalInterface annotation?

This annotation is used in functional interfaces to add more readability to the code as an informative aspect. However, it does not affect the runtime or the semantics of the code.

6. What is the meaning of functional interfaces in Java 8?

Functional interfaces in Java 8 are interfaces having a single abstract method.

Following are the three types of methods that can be present:

- The static method
- The default method
- The overridden class method

- **7. What is the use of the String::ValueOf expression in Java 8?**
- String::ValueOf is a simple static method referencing the valueOf method, belonging to the class 'String.'
- Next up on this Java8 Interview Questions and answers post, we have to check out an important concept regarding the interface.
- **8. Is it possible to create a custom functional interface in Java 8?**
- Yes, Java 8 supports the creation of a custom functional interface.
- The following example denotes the creation of an interface called 'CustInterface' as shown:

```
package org.arpit.java2blog;
public interface CustInterface{
```

```
•
•  void print();
•  default void printColor()
•  {
•  System.out.println("Printing the color");
•  }
•  }
```

- The main class can be created as shown below:

```
•  package org.arpit.java2blog.constructor;
•  public class FunctionalIntefaceMain {
•  public static void main(String[] args)
•  {
•  FunctionalIntefaceMain pMain=new FunctionalIntefaceMain();
•  pMain.printForm(() -> System.out.println("Printing form"));
•  }
•  public void printForm(CustInterface c)
•  {
•  c.print();
•  }
•  }
```

- When the program runs, it prints out the 'printing form' message to the screen. It is very vital that you **learn Java** with the programming approach.

- **9. What is the meaning of method reference in Java 8?**
- Method references are used in Java 8 to refer to methods of functional interfaces. It can be considered as a short-code version of using a lambda expression.
- The following is the expression for a method reference:

```
•  Class::methodname
```

- **10. Differentiate between a predicate and a function in Java 8.**

Predicate	Function
Returns True or False	Returns an object
Used as an assignment target for lambda expressions	Can be used for both lambda expressions and method references

11. What are default methods in Java 8?

Default methods are the interfaces in Java 8 that make use of the 'default' keyword. Default methods are added onto Java 8 to give users the functionality and ability to use backward compatibility.

12. What are the core API classes for date and time in Java 8?

There are three main core API classes for date and time in Java 8 as given below:

- LocalDate
- LocalTime
- LocalDateTime

13. What is the easiest way to print the current date and time using the new APIs in Java 8?

The 'now' method, which is a part of LocalDate, can be used to get the current date as shown below:

```
LocalDate currentDate = LocalDate.now();
System.out.println(currentDate);
```

Similarly, it can also be used to get the current time:

```
LocalTime currentTime = LocalTime.now();
System.out.println(currentTime);
```

14. What were the issues that were fixed with the new Date and Time API of Java 8?

With the older versions of **Java**, java.util.The date was mutable. This means it has absolutely no thread safety.

Also, java.text.SimpleDateFormat was not thread-safe in the older versions. The older Date and Time API was difficult to understand for programmers in terms of readability too.

15. What are PermGen and Metaspace in Java 8?

The Java Virtual Machine has been using PermGen for class storage until Java 7. It is now superseded by Metaspace.

Metaspace has a huge advantage over PermGen that makes the former grow dynamically without any constraint, while PermGen has a fixed maximum size.

16. Differentiate between intermediate and terminal operations in Java 8.

Intermediate Operation	Terminal Operation
Used for the transition to a new state	Used to end the process under execution
Lazy execution of code, i.e., code is not executed as soon as it is encountered	Not lazy; code is immediately executed upon encounter

17. Can the following piece of code compile successfully?

```
@FunctionalInterface
public interface Function2<T, U, V> {
public V apply(T t, U u);
default void count() {
    // increment counter
}
}
```

Yes, the code can compile and execute without any errors. It uses functional interface specifications when the single abstract method is being defined.

18. What is Nashorn in Java 8?

Nashorn is a newly introduced JavaScript processing engine that came bundled with Java 8. It provides tighter compliances with ECMA JavaScript specifications and has runtime performance that beats Rhino, its predecessor.

19. What is the use of the optional keyword in Java 8?

The optional keyword is used in Java 8 to avoid the occurrence of the NullPointerException.

An optional object can be created easily using the empty() static method as shown below:

```
@Test

public void whenCreatesEmptyOptional_thenCorrect() {

Optional<String> empty = Optional.empty();
```

```
assertFalse(empty.isPresent());

}
```

20. What is stream pipelining used for?

Stream pipelining is a concept that is implemented in Java 8 so that users can chain more than one operation at a time. This works on the principle of splitting the operation into two categories:

- **Intermediate operations**: Return the instance of the stream when running
- **Terminal operations**: Used to terminate the operation and return the final value

21. What is JJS in Java 8?

JJS is the common line tool that comes packaged with Java 8. It is used to run JavaScript code seamlessly using just the console.

22. What is the code to sort strings using the Java 8 lambda expression?

The below piece of code sorts strings using the lambda expression:

//Sorting using Java 8 lambda expression

```
private void sortUsingJava8(List<String> names) {

Collections.sort(names, (s1, s2) -> s1.compareTo(s2));

}
```

23. Is it possible to call a static method of any interface in a class using Java 8?

Yes, it is possible to call a static method in a class by making use of the name as shown below:

```
interface Vehicle {
static void lightsOn() {
System.out.println("Turning on Lights!");
```

```
}
}

class Car implements Vehicle {
public void print() {
Vehicle.lightsOn();
}
}
```

Next up in this Blog on Java 8 Interview Questions for the experienced, we have to check out an important concept regarding keywords.

24. Can you briefly explain the working of the random keyword in Java 8?

The random keyword, as the name suggests, is used to generate random values for computations and operations in Java 8.

The following piece of code is used to print out 20 random numbers using the forEach loop:

```
Random random = new Random();
random.ints().limit(20).forEach(System.out::println);
```

25. What are collectors in Java 8?

Collectors are mainly used to combine the final result after the processing of elements in a stream. They are used to return lists or strings.

The following piece of code denotes how collectors work:

```
List<String>strings = Arrays.asList("abc", "", "bc", "efg", "abcd","",
"jkl");

List<String> filtered = strings.stream().filter(string ->
!string.isEmpty()).collect(Collectors.toList());

System.out.println("Filtered List: " + filtered);
```

```
String mergedString = strings.stream().filter(string ->
!string.isEmpty()).collect(Collectors.joining(", "));


System.out.println("Merged String: " + mergedString);
```

If you are looking forward to becoming proficient in Java, make sure to check out Intellipaat's **Java Course** program.

26. What is the easiest way to print the sum of all of the numbers present in a list using Java 8?

In Java 8, the following code is used to print the sum of all of the numbers that are present in a list:

```
List<Integer> numbers = Arrays.asList(5, 4, 10, 12, 87, 33, 75);
IntSummaryStatistics stats = integers.stream().mapToInt((x) –>
x).summaryStatistics();
System.out.println("Sum of all numbers : " + stats.getSum());
```

27. Can JavaScript code be executed from Java 8 codebase?

Yes, JavaScript code can be easily executed using codebase by making use of ScriptEngineManger. This is used to interpret the code in Java 8.

28. When is an ideal situation to use the Stream API in Java 8?

The Stream API in Java 8 can be effectively used if the Java project calls for the following operations:

1. Perform database operations
2. Execute operations lazily
3. Write functional-style programming
4. Perform parallel processing
5. Use pipeline operations
6. Use internal iteration

29. How can you print the date of the next occurring Wednesday using Java 8?

The following piece of code prints the next occurring Wednesday in the calendar:

```
//Print the next occurring Wednesday
LocalDate today = LocalDate.now();
LocalDate nextWednesday =
today.with(TemporalAdjusters.next(DayOfWeek.WEDNESDAY));
System.out.println("Next Wednesday on : " + nextWednesday);
```

30. Which class implements the encoder used for encoding byte data in Java 8?

The static class Base64.Encoder is used in Java 8 for implementing an encoder, which can encode byte data using the Base64 encoding scheme easily.

31. How is a Base64 decoder created in Java 8?

The getDecoder() method, which is a part of the Base64 class, is used to return a Base64.Decoder. This decodes by making use of the Base64 encoding scheme.

32. How is a Base64 encoder that encodes URLs created in Java 8?

The getUrlEncoder() method, again a part of the Base64 class, is used to encode the URLs.

33. What is a supplier in Java 8?

A supplier is a simple functional interface in Java 8 that does not take in any argument. It is used as an assignment target when making use of lambda expressions.

The following is an example that denotes the usage of a supplier:

```
import java.util.function.Supplier;
public class Java8SupplierExample {
public static void main(String[] args) {
Supplier<String> supplier= ()-> "HelloLearners";
System.out.println(supplier.get());
```

```
}
}
```

34. What is a consumer in Java 8?

Just like a predicate, a consumer is a single argument functional interface present in Java 8. However, the consumer does not return any value and can be used for lambda expressions.

The following piece of example code denotes the usage of the consumer interface to print a string:

```
import java.util.function.Consumer;
public class Java8ConsumerExample {
public static void main(String[] args) {
Consumer<String> consumerString=s->System.out.println(s);
consumerString.accept("HelloWorld");
}
}
```

35. Differentiate between findFirst() and findAny() in the Stream API of Java 8.

findFirst()	findAny()
Always returns the first element from a stream	Can choose any element present in the stream
Deterministic behavior	Non-deterministic behavior

36. Differentiate between Collection API and Stream API in Java 8.

Collection API	Stream API
Helps store object data	Helps in the computation of data
Stores a limited number of elements	Can store an unlimited number of elements
Eager execution	Lazy execution

37. What is the meaning of a Spliterator in Java 8?

Spliterator is a newly introduced iterator interface for Java 8. It is very efficient and handles API-related operations seamlessly across the runtime environment.

38. Differentiate between Spliterator and a regular iterator in Java 8.

Spliterator	Iterator
Introduced with Java 8	Present since Java 1.2
Used in Stream API	Used in Collection API
Helps in the iteration of streams in a parallel and sequential order	Iterates collections in a sequential order only
Example: tryAdvance()	Examples: next() and hasNext()

39. Can you name the common types of functional interfaces in the standard library?

There are many functional interface types in the standard library, and some of them are as follows:

- BiFuction
- BinaryOperator
- Consumer
- Predicate
- Supplier
- UnaryOperator

40. What are the similarities between map and flatMap stream operations in Java 8?

Both map and flatMap operations are a form of intermediate stream operations that take in a function and use the input function for performing various activities in the stream.

41. Can you give examples of intermediate operations in Java 8?

The examples that are widely used in intermediate operations are:

- Distinct()
- Limit(long n)
- Filter(Predicate)
- Map(Function)
- skip(long n)

42. What are some of the examples of terminal operations in Java 8?

Following are some of the examples of terminal operations in Java 8:

- Count
- Min
- Max
- Reduce
- toArray
- anymatch
- allMatch

43. What is the easiest way to find and remove duplicate elements from a list using Java 8?

Duplicate elements can be listed and removed easily by applying stream operations and performing a collection, later using the Collections.toSet() method. This should remove all of the duplicate elements present in the list.

44. What is the use of the peek() method in Java 8?

The peek() method is a part of the stream class in Java 8, which is used to see actions performed through a stream pipeline. Peeking can be done at every step to print messages about the code being executed onto the console.

Peeking has a wide amount of usage when efficiency is a requirement, when debugging code with the lambda expression, or when performing stream processing.

45. What is the syntax of a predicate interface in Java 8?

Predicate is a functional interface used in Java to take in an object and return a Boolean value.

The following is the syntax of a predicate function:

```
public boolean test(T object){

return boolean;

}
```

46. What is the easiest way to convert an array into a stream in Java 8?

Any array in Java 8 can be converted into a stream easily using the stream class. The creation of a stream using a factory method is as shown below:

```
String[] testarray = {"Hello", "Intellipaat", "learners"};
Stream numbers = Stream.of(testarray);
numbers.forEach(System.out::println);
```

Next up on this set of top Java 8 interview questions and answers, we have to check out questions that are dependent on your learning and experience.

47. Why do you think you are the right fit for this Java Developer role?

This is a very commonly asked question in a Java 8 interview. With this, the interviewer wants to understand your proficiency in Java and how you want to take it up for your career path and growth.

It would be advantageous to answer this question based on your interests in Java programming and how you plan to use it to the best of your abilities when you join the firm.

48. Do you have any past work experience in a production environment involving Java?

This question has a high probability of coming up if you are an experienced developer getting interviewed. Make sure to answer it to the best of your abilities and provide an

honest answer about your experience with Java development in a production environment.

49. Do you have any experience working in the same industry as ours?

This is a question based on the type of company you have applied for. There are a variety of industries that make use of Java 8 for their respective requirements. Understanding the job description and what is expected of you will help you in answering this question, alongside talking about any working experience you have previously in the same field.

If you are coming from another industry, make sure to talk about how you are planning to adopt and bring the best of you to the table.

50. Do you have any certification to boost your candidature for this Java Developer role?

Having a certification from a reputed organization will add a lot of weightage to your resume, alongside providing you in-depth knowledge about the technology you want to master.

With such a certification comes industry-level projects to work on, which will help you immensely to prove to the interviewer that you have put in a solid amount of time and effort to master the technology.

This will also add credibility to your career growth in the technology that you have certified in. This should give you a huge boost in terms of knowledge, learning, and the competition you face during the interview.