1. What is stream pipelining?

Chaining of stream operation together is known as stream pipelining. There are two types of stream operations: Terminal and Intermediate.

- **Intermediate Operation**:- Intermediate operation is the operation that returns streams i.e. it cannot be chained. These operations are always lazy.
- **Terminal Operation**:- Terminal operation end the pipeline and begins stream processing.

2. What are the new features introduced in Java 8?

The new features introduced in Java 8 are:

- Stream API - Parallel
- for Each() Method
- Lambda Expression
- default and static methods
- @Functional Interface
- Functional Expressions
- Java Time API
- Collection API improvements
- Concurrency API improvements
- Java IO improvements

3. Why static method are introduced in interfaces?

Static methods are introduced because if we want to provide a default implementation that cannot be changed we have to make it static in interfaces. Also, the static method in interfaces cannot be overridden.

4. What is the advantage of utilizing the new Date API?

Before Java 8, we didn't have any time-zone support. So what we have to do basically is to programmatically change the time zone with the help of the if-else statement. But with the new Date API, there is support for the time-zone. The important classes most frequently used are: LocalDate, LocalDateTime, ZonedDateTime

5. What is a lambda expression?

Lambda expression is another component in Java 8. Basically, lambda expressions are functions that have no name and identifier. The Lambda feature encourages functional programming.

6. What do you understand by @Functional Interface annotation Java 8?

@Functional Interface is a feature of Java * which will force the compiler to check whether the interface given has an abstract method or not. If the given interface has no abstract method then the compiler will show you the error "unexpected @FunctionalInterface annotation".

7. How to avoid Nullpointer exception in Java 8?

Basically, Java 8 provides the concept of optional. Optionals are used which help us to avoid Nullpointer exception.

8. What do you mean by Stream?

Streams are basically a sequence of data coming from a source. In Java 8 we can manipulate data using stream API.

For example:- when you are watching any video on Youtube or any other platform, the content is not loaded at once but as time elapses the content gets loaded. This is known as stream.

9. What is polymorphism?

Polymorphism is the ability to exist in different multiple forms. In Java context, polymorphism allows to perform a single action in various ways.

10. What are the difference between Collection API and Stream API?

Collection API:- Collection API are utilized for categorizing data from various data structures. They can store only a limited number of elements in their data structure.

Stream API:- Apart from Collection API, Stream API is utilized for the calculation of a huge set of data. They can store unlimited numbers of elements in their data structure.

Java 8, **BiFunction** is a functional interface; it takes two arguments and returns an object

Java 8 Main Agenda

- Functional programming

- Conciseness in the code -less line of code

Functional Programming
- Lambada expression to create consice code base.

Java 8 New Feature

Lambada Expression
Stream Api
Default methods in interface
static methods
Functional Interface
Optional
Method References
Date API
Nashorn Javascript Engine

Advantage of Java8

Compact code(Less boiler plate code)
More readable and reusable code
More Testable code
Parallel Operation

Lambada Expression
Annonyomus function without name body type
function void add(int a,int b){System.out.println(a+b)}
(a,b) -> System.out.println(a+b);

Bifunction
Represents a function that accepts two arguments and produces a result. This is the two-arity specialization of Function.
biconsumer one method accept
supplier

Functional Interfaces
- Interfaces which have only one abstarct method.
- It can have any number of static and default method.
- runaable run

Functional interface is reference to the lambada expression.
Comparator<String> e = (S1,S2)->S1.compareTo(S2)
Functional interface Lambada expression

Own functional interface
- Create functional interface
- One single abstract method
- There is no restriction for the default and static method
- @FunctionalInterface
above do not allow to add multiple abstract method

Method referencing use existing code
MyFunctionalint abc = demo ::test;
abc.my();

Default method
default void my()()
It will not give implementation errors.
Dont satisfy then own
It is not access modifier

Can't override hashCode method in object method in interface.

Diamond Problem
Interface m1(
Default m1()(
System.out.println(m1)
)
)

Interface m2(
Default m2()(
System.out.println(m2)
)
)

Interface m3(

)