Describe the newly added features in Java 8?

* Lambda Expressions
* Functional Interfaces
* [Stream API](https://www.journaldev.com/2774/java-8-stream)
* Date and Time API
* [Interface Default Methods and Static Methods](https://www.journaldev.com/2752/java-8-interface-changes-static-method-default-method)
* Method References

What is Lambda Expression?

Lambda Expression is an anonymous function which accepts a set of input parameters and returns results.Lambda Expression is a block of code without any name, with or without parameters and with or without results. This block of code is executed on demand

What are the three parts of a Lambda Expression? What is the type of Lambda Expression?

Can you explain the syntax of Lambda expressions?

Lambda expressions has 3 parts

part is parameter or argument

arrow operator

statement or lambda expression body

What are the main characteristics of the Lambda Function?

* A method that is defined as Lambda Expression can be passed as a parameter to another method.
* A method can exist standalone without belonging to a class.
* There is no need to declare the parameter type because the compiler can fetch the type from the parameter’s value.
* We can use parentheses when using multiple parameters but there is no need to have parenthesis when we use a single parameter.
* If the body of expression has a single statement then there is no need to include curly braces.

What is a Functional Interface? What is SAM Interface?

A Functional Interface is an interface, which contains one and only one abstract method. Functional Interface is also known as SAM Interface because it contains only one abstract method.

SAM Interface stands for Single Abstract Method Interface

What is the use of a functional interface?

One of the major benefits of functional interface is we can use **lambda expressions** to instantiate them. So lambda expressions are meant to create anonymous classes of functional interfaces easily. There are no runtime benefits of using lambda expressions, but it makes code lines less

Can a functional interface extend/inherit another interface?

A functional interface cannot extend another interface with abstract methods as it will void the rule of one abstract method per functional interface.

It can extend other interfaces which do not have any abstract method and only have the default, static, another class is overridden, and normal methods

Is it possible to define our own Functional Interface? What is @FunctionalInterface? What are the rules to define a Functional Interface?

* Yes, it is possible to define our own Functional Interfaces. We use Java SE 8’s @FunctionalInterface annotation to mark an interface as Functional Interface.
* Define an interface with one and only one abstract method.
* We cannot define more than one abstract method.
* Use @FunctionalInterface annotation in interface definition.
* We can define any number of other methods like Default methods, Static methods.

What is the default method, and why is it required?

A method in the interface that has a predefined body is known as the default method. It uses the keyword default. default methods were introduced in Java 8 to have

* It allows us to provide method’s implementation in Interfaces.
* To add new Functionality to Interface without breaking the Classes which implement that Interface.
* To provide elegant Backwards Compatibility Feature.
* To ease the extension of the existing Functionality.
* To ease of Maintain the existing Functionality

'Backward Compatibility in case JDK modifies any interfaces. In case a new abstract method is added to the interface, all classes implementing the interface will break and will have to implement the new method. With default methods, there will not be any impact on the interface implementing classes. default methods can be overridden if needed in the implementation. Also, it does not qualify as synchronized or final.

What is a Static Method? Why do we need Static methods in Java 8 Interfaces?

A Static Method is an Utility method or Helper method, which is associated with a class (or interface). It is not associated with any object.

* We can keep Helper or Utility methods specific to an interface in the same interface rather than in a separate Utility class.
* We do not need separate Utility Classes like Collections, Arrays etc to keep Utility methods.
* Clear separation of Responsibilities. That is we do not need one Utility class to keep all Utility methods of Collection API like Collections etc.
* Easy to extend the API.
* Easy to Maintain the API.

What is method reference in java 8?

Method reference is a shorthand notation of a lambda expression to call a method.

Method reference to an instance method of an object – object::instanceMethod  
 Method reference to an instance method of an arbitrary object of a particular type – Class::instanceMethod

Method reference to a constructor – Class::new

Method reference to a static method of a class – Class::staticMethod

What are some standard Java predefined functional interfaces?

Some of the famous predefined functional interfaces from previous Java versions are Runnable, Comparator, and Comparable.

What are the various categories of pre-defined function interfaces?

* Predicate<T> return boolean,test(), and(), or(),negate().
* Consumer<T>accepts a single input argument and returns no result,accept()
* Function<T, R>input argument of a certain type and produces a result of another type. apply()
* Supplier<T>t takes no arguments and only returns some value.get()
* UnaryOperator<T>when the argument and the result are of the same type. Function()

**What is a Predicate? State the difference between a Predicate and a Function?**

predicate return boolean, function return type as Object.

Are you aware of Date and Time API introduced in Java 8? What the issues with Old Date and time API?

For old date time api, it is not thread safe, performance is poor, old api such as calendar and date are poorly designed and hard to understand.

Can you provide some APIs of Java 8 Date and TIme?

LocalDate, LocalTime, LocalDateTime

How will you get the current date and time using Java 8 Date and TIme API?

LocalDate.now()

What are the Differences between Java’s OLD Java Date API and Java 8’s Date and Time API?

| S.No. | Java’s OLD Java Date API | Java 8’s Date and Time API |
| --- | --- | --- |
| 1. | Available before Java 8 too. | It is introduced in Java SE 8 |
| 2. | Not Thread Safe. | Thread Safe. |
| 3. | Mutable API. | Immutable API. |
| 4. | Less Performance. | Better Performance. |
| 5. | Less Readability. | More Readability. |
| 6. | It’s not recommended to use as its deprecated. | It’s always recommended to use. |
| 7. | Not Extendable. | Easy to Extend. |
| 8. | It defines months values from 0 to 11, that is January = 0. | It defines months values from 1 to 12, that is January = 1. |
| 9. | It’s an old API. | It’s a new API. |

What is the Diamond Problem in interfaces due to default methods? How Java 8 Solves this problem?

class D implements B, C {

@Override

public void display() {

B.super.display();

}

}

What are Java 8 streams?

The streams are not collections. streams are WRAPPERS for collections and arrays.They wrap an EXISTING collection (or another data source) to support operations expressed with LAMBDAS, so you specify what you want to do, not how to do it

Streams are immutable

Streams are not reusable

Streams don't store their elements

Streams work perfectly with lambdas

Why do we require stream API?

* It supports Functional-Style programming.
* It does faster processing. Hence, it is apt for better performance.
* It allows parallel operations.

What are the main components of a Stream?

* A data source
* Set of Intermediate Operations to process the data source
* Single Terminal Operation that produces the result

What are Intermediate and Terminal operations?

intermediate operations:

lazy operator, transforms a stream to another stream,processes the stream data, mostly lambda function, could have one or more intermediate operators.

filter(), map(), distinct(), sorted(), limit()

Terminal Operations

always return something other than a stream.

forEach(), toArray(), reduce(), collect(), min(), max(), count(),

(difference between map and flatmap)

map() function produces one output for one input value, whereas flatMap() function produces an arbitrary no of values as output (ie zero or more than zero) for each input value

How to convert String, Collection, Array into stream?

First, we have to cover it into stream first.

Array.asList(new Integer[]{12,3,4,5}).stream.forEach(System.out::println);

Strin s=”hello”;

s.chars().mapToObj(x->(char)x).forEach(System.out::println);

This operator is covered s into characters then mapping each element into x and type casting. We have a character type of string.

Remove duplicate string

s.chars().mapToObj(x-> (char)x).distinct.forEach(System.out::println);

Integer[] array = {1,2,3,4,5};

Stream.of(array).forEach(System.out::println);

**Intermediate Operations :**

map(), filter(), distinct(), sorted(), limit(), skip()

**Terminal Operations :**

forEach(), toArray(), reduce(), collect(), min(), max(), count(), anyMatch(), allMatch(), noneMatch(), findFirst(), findAny()