Java SE 8 New Features?

- Lambda Expressions
- Functional Interfaces
- Stream API
- Date and Time API
- Interface Default Methods and Static Method
- Method and Constructor References
- Collections API Enhancements
- javac Enhancements
- JVM Changes
- JDBC 4.2

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 Database Like Operations
- Better Performance Applications
- More Productive code

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?

A Lambda Expression contains 3 parts:

- Parameter List

  A Lambda Expression can contain zero or one or more parameters. It is optional.

- Lambda Arrow Operator

  "->" is known as Lambda Arrow operator. It separates parameters list and body.

- Lambda Expression Body

The type of “Journal Dev” is java.lang.String. The type of “true” is Boolean. In the same way, what is the type of a Lambda Expression?
The Type of a Lambda Expression is a Functional Interface.

Example:- What is the type of the following Lambda Expression?

```
() -> System.out.println("Hello World");
```

This Lambda Expression does not have parameters and does return any results. So it’s type is “java.lang.Runnable” Functional Interface.

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. Java SE 8 API has defined many Functional Interfaces.

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.

We need to follow these rules to define a 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.

Is @FunctionalInterface annotation mandatory to define a Functional Interface? What is the use of @FunctionalInterface annotation? Why do we need Functional Interfaces in Java?

It is not mandatory to define a Functional Interface with @FunctionalInterface annotation. If we don’t want, We can omit this annotation. However, if we use it in Functional Interface definition, Java Compiler forces to use one and only one abstract method inside that interface.