JAVA 8 Features

* Lambda expressions,
* Method references,
* Functional interfaces,
* Stream API,
* Default methods,
* Base64 Encode Decode,
* Static methods in interface,
* Optional class,
* Collectors class,
* ForEach() method,
* Nashorn JavaScript Engine,
* Parallel Array Sorting,
* Type and Repating Annotations,
* IO Enhancements,
* Concurrency Enhancements,
* JDBC Enhancements etc.

Java Default Methods

Java provides a facility to create default methods inside the interface. Methods which are defined inside the interface and tagged with default are known as default methods. These methods are non-abstract methods.

Java Default Method Example

In the following example, Sayable is a functional interface that contains a default and an abstract method. The concept of default method is used to define a method with default implementation. You can override default method also to provide more specific implementation for the method.

Let's see a simple

1. **interface** Sayable{
2. // Default method
3. **default** **void** say(){
4. System.out.println("Hello, this is default method");
5. }
6. // Abstract method
7. **void** sayMore(String msg);
8. }
9. **public** **class** DefaultMethods **implements** Sayable{
10. **public** **void** sayMore(String msg){        // implementing abstract method
11. System.out.println(msg);
12. }
13. **public** **static** **void** main(String[] args) {
14. DefaultMethods dm = **new** DefaultMethods();
15. dm.say();   // calling default method
16. dm.sayMore("Work is worship");  // calling abstract method
18. }
19. }

In case we have same default method in different interfaces and its implemented the both interfaces in the one class. It is mandatory to override the default methods.

Default method is also called as defender method or virtual extension method

Functional Interface

1. The interface that to be said as a functional interface it should be have only one abstract method

+ One or more default methods

+ One or more Static methods

1. It can be annotated ( We can use @ )
   1. Annotations
      1. Annotations start with ‘**@**’.
      2. Annotations do not change the action of a compiled program.
      3. Annotations help to associate *metadata* (information) to the program elements i.e. instance variables, constructors, methods, classes, etc.
      4. Annotations are not pure comments as they can change the way a program is treated by the compiler. See below code for example.
      5. Annotations basically are used to provide additional information, so could be an alternative to XML and Java marker interfaces.

Functional Programming

In java 8 we can use a functions inside the other functions. I t can be achieved by using the lambda expression and method reference

Lambda Expression

The open close brackets with the arrow is called as the lambda expression () ->