Java 8 new feature

Java 8 has significant changes since java 5. It embeds many popular libraries like DateTime implementation of joda, concurrency of stream etc.

New Features

There are dozens of features added to Java 8, the most significant ones are mentioned below −

* **Lambda expression** − Adds functional processing capability to Java.
* **Method references** − Referencing functions by their names instead of invoking them directly. Using functions as parameter.
* **Default method** − Interface to have default method implementation.
* **New tools** − New compiler tools and utilities are added like ‘jdeps’ to figure out dependencies.
* **Stream API** − New stream API to facilitate pipeline processing.
* **Date Time API** − Improved date time API.
* **Optional** − Emphasis on best practices to handle null values properly.
* **Nashorn, JavaScript Engine** − A Java-based engine to execute JavaScript code.

Along with these new features, lots of feature enhancements are done under-the-hood, at both compiler and JVM level.

Lambda expression:

. Lambda expressions are used primarily to define inline implementation of functional interface, ie. an interface with a single method only (default methods are not counted).

. Lambda expression eliminates the need of anonymous class and gives a very simple yet powerful functional programming capability to java

Be aware of that using lambda expression, we can refer to final variable or effectively final variable (which is assigned only once). Lambda expression throws a compilation error, if a variable is assigned to a value the second time.

. Method reference

Method references help to point to methods by their names. A method reference is described using :: (double colon) symbol. A method reference can be used to point the following types of methods –

Static method, instance methods, constructors using new operator (TreeSet::new)

Java 8 provides java.util.function package which contains the common functional interface

**Purpose of java.util.function package**: For common use cases where a lambda expression or a method reference is needed, these are generally assigned to a target type of a Functional Interface which has its [function descriptor](http://www.javabrahman.com/java-8/function-descriptors-java-8-explained/)( abstract method’s signature in terms of parameter & return types) which should match the signature of the lambda expression.

**java.util.function package provides a set of re-usable common functional interfaces( and their corresponding lambda) definitions which can be used by the programmer in his code instead of creating brand new functional interfaces.**

To sum up this section – **java.util.function** package provides standard library-based functional interface for common requirements as an alternative to creating brand new functional interfaces every time one is needed.

Contexts in which Functional Interfaces can be assigned/used: There are 3 ways\contexts in which functional interfaces are used as target types in code

* **Assignment context** – This refers to cases where a lambda/method reference is assigned to the functional interface **Predicate<T>**’s instance –

**Predicate predicateToTest=(Integer integer -> integer > 10);**

* **Method invocation context** – This is when a lambda/method reference is passed to a method parameter which accepts the equivalent functional interface’s instance –

**stream.filter(String::isEmpty)**

* **Cast context** – where a lambda/method reference of one type is explicitly typecast to a functional interface of another type –

**stream.map((ToIntFunction) e -> e.getSize())**

good ref for lamda and method reference

<https://dzone.com/articles/java-lambda-expressions-vs>

. Functional interface

Functional interfaces have a single functionality to exhibit. Eg. a comparable interface with a single method ‘compareTo’ is used for comparison purpose. Java 8 has defined lots of functional interfaces to be used extensively in lambda expressions. @FunctionalInterface annotation is for self-declared functional interface annotation.

. Default method

We can implement default method in interface to make backward compatible. Eg. java 8 added forEach method for each collection class. This method is implemented in Collection interface, so the classes don’t have any change. Default method cannot completely replace abstract class since abstract can define private properties which are state of the class.

. Stream

Java 8 introduced the concept of stream that lets the developer to process data declaratively and leverage multicore architecture without the need to write any specific code for it.

Stream is one of the most important new feature in java 8. It is under java.util.stream package.

. Optional class

. javascript engine

. DateTime

. Base64 encode and decode

All these functions stream, lambda, and DateTime are most often used. We can search for the detail when we make use of these functions.