**Java 8** (18 mars 2014) => if faut le JDK8 + nouveau IDE Eclipse prend en charge JDK8

## 1. Interface fonctionnelle (Single abstract method)

Une interface qui n'a qu'une seule méthode abstraite (non implémentée).

SAM interfaces peuvent contenir des méthodes par défaut et statiques qui ont une implémentation.

L'expression lambda/référence de méthode fournit l'implémentation d'une interface fonctionnelle.

@**FunctionalInterface**, utilisée pour déclarer une interface comme interface fonctionnelle.

## 2. Expressions lambda

Utilisé dans la programmation fonctionnelle => EL = fonction qui n'appartient pas à une classe.

Utiliser pour Implémenter Listeners, Collbacks, SAM avec l'API Java Stream (Utilisé avec collections pour itérer, filtrer et extraire des données)

Une expression lambda peut être transmise comme objet et exécutée à la demande.

EL est traitée comme une fonction, donc le compilateur ne crée pas de fichier .class

EL pour : implémenté une interface fonctionnelle, Moins de code.

Syntaxe EL : (**Liste d'arguments 0,n) -> {corps}**

**Type de paramètres facultative** : Le compilateur peut déduire le type de la valeur du paramètre.

**Parenthèse facultative** autour du paramètre : Pas besoin de déclarer un seul paramètre entre parenthèses. Si vous avez plusieurs paramètres, des parenthèses sont requises.

**Accolades facultatives** : Pas besoin d'utiliser des accolades dans le corps d'une expression si le corps contient une seule instruction.

Le mot clé « **return » est facultatif** : Le compilateur renvoie automatiquement la valeur si le corps a une seule expression. Des accolades sont nécessaires pour indiquer que l'expression renvoie une valeur.

## 3. Référence de méthode

Une référence de méthode est utilisée pour faire référence à une méthode d'un SAM

Un type spécial des EL et plus facile qu'une EL.

3 types de RDM : Référence à une méthode statique, une méthode d'instance, un constructeur.

ClassName::staticMethodName / ObjectName::instanceMethodName / ClassName::**new**

## 4. Class Optional

Nouvelle class dans java 8 utilisée pour traiter l'exception NullPointerException (Il faut importer le package java.util).

Il fournit des méthodes pour vérifier la présence d'une valeur pour une variable particulière.

La méthode isPresent() pour vérifier s'il existe une valeur à l'intérieur de l'objet Optional. Une valeur n'est présente que si nous avons créé Optional avec une valeur non nulle.

## 5. forEach

Il est défini dans les interfaces Iterable et Stream

Les classes Collection qui héritent l'interface Iterable peuvent utiliser la méthode forEach() pour itérer les éléments.

Cette méthode prend un seul paramètre qui est une interface fonctionnelle. Ainsi, vous pouvez passer une expression lambda comme argument.

## 6. API Date/Time

Le package java.time contient des classes de date et d'heure en Java 8.

## 7. Méthode par défaut

Les méthodes définies à l'intérieur d'une interface et étiquetées avec le mot-clé « **default** » sont appelées méthodes par défaut. Ces méthodes sont des méthodes non abstraites et peuvent avoir un corps de méthode.

## 8. API Stream

Le package java.util.stream se compose de classes et d'interfaces pour permettre des opérations de style fonctionnel sur les éléments pour traiter les collections.

Stream prend une collection, un tableau ou des E/S.

Les Stream ne modifient pas la structure de données d'origine, ils fournissent uniquement le résultat selon les méthodes enchaînées.

Chaque opération intermédiaire est exécutée et renvoie un stream en conséquence, de sorte que diverses opérations intermédiaires peuvent être enchaînées. Les opérations terminales marquent la fin du flux(stream) et renvoient le résultat.

La méthode **map** est utilisée pour renvoyer un flux constitué des résultats de l'application d'une fonction donnée aux éléments de ce flux.

La méthode **filter** est utilisée pour sélectionner les éléments selon le prédicat passé en argument.

La méthode **sorted** est utilisée pour trier un flux.

La méthode **collect** pour renvoyer le résultat des opérations intermédiaires effectuées sur le flux.

La méthode **forEach** est utilisée pour parcourir chaque élément du flux.

La méthode **reduce** est utilisée pour réduire les éléments d'un flux à une seule valeur.