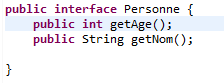
**Langage de programmation :**

**Lambda Expressions** : une nouvelle fonctionnalité linguistique, a été introduite dans cette version. Ils nous permettent de traiter la fonctionnalité comme un argument de méthode.

* Les méthodes par défaut permettent d'ajouter de nouvelles fonctionnalités aux interfaces des bibliothèques.

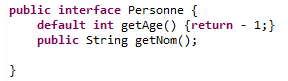
Exemple : on a une interface *Personne* qui possède *getAge()* comme une méthode.

On déclare la manière classique avec java 7 :



On a une classe nommée *PersonneMorale*par exemple. Cette classe va implémenter l’interface Personne mais logiquement la classe *PersonneMorale* ne va pas avoir la méthode *getAge().* Pour cela, on revient sur l’interface *Personne* pour ajouter la nouvelle fonctionnalité de java 8 aux interfaces.

on obtient un résultat ainsi :



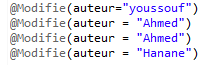
* Les références de méthodes fournissent des expressions lambda faciles à lire pour les méthodes déjà nommées.

Exemple : dans la classe *Principale,* on fait appel a la méthode par :: ;

Arrays.sort(tab, Personne :: compare) ;

Les annotations répétées permettent d'appliquer le même type d'annotation plusieurs fois à la même déclaration ou au même type d'utilisation.

Exemple avec les annotations en java 8 :



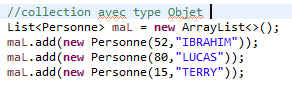
* Amélioration de l'inférence de type : Par exemple la déclaration de *List* , on a plus besoin de répéter les types du tableau dans *ArrayList*.



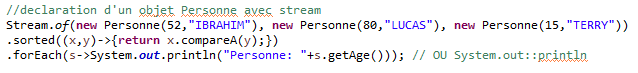
* [**Les collections**](http://docs.oracle.com/javase/8/docs/technotes/guides/collections/changes8.html)**:**

Les classes flux :

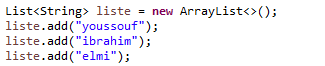
Les classes du nouveau *java.util.stream* package fournissent une API de flux permettant de prendre en charge des opérations de style fonctionnel sur des flux d'éléments.



Ou en peux déclarer directement avec *Stream*;



Si on utilise des collections classiques avec une chaine des caractères ;



Maintenant, si on veut qu’on affiche nos données avec un *forEach,* voici la syntaxe :

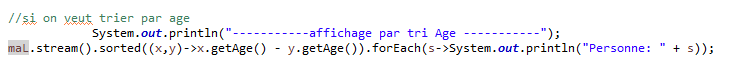


Ou afficher de cette manière ;

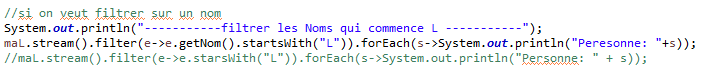


**L'API Stream** est intégrée à l'API Collections, permet de réaliser plusieurs fonctionnalités :

* **sorted** (Triés) par exemple :



* **filter**(Filtrer), qui possède plusieurs fonctions. On verra quelque utilisation avec *filter*





* **collect**, pour collecter un ensemble des calcules et de les affichés ensuite ;



* **reduce** (réduire), nous devons effectuer des opérations où un flux réduit à une seule valeur résultante, par exemple, maximum, minimum, somme, produit, etc. La réduction est le processus répété de combinaison de tous les éléments.



* Les interface Fonctionnelle : c’est une interface qui représente une fonction et elle contient
* **Predicate** : qui est un type d’une interface fonctionnelle. Le prédicat prend au paramètre un type et retourne rien ou au pire retourne une booléen.

Voici un exemple :



* **Function** : elles prennent un paramètre et retourne quelque chose.

Exemple d’une fonction qui prend une température Celsius et retourne en fahrenheit.



* **BiPredicate** : c’est comme la fonction prédicat mais elle prenne deux paramètres en entrer.

Exemple de comparaison de deux chaines des caractères :



* **BiFunction** : c’est comme la fonction *Function*, elle prenne trois paramètres pour retourne un résulte en utilisant une lambda expression comme les autres aussi.

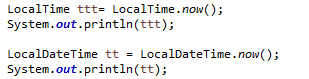
Exemple :la somme de deux chaine des caractères.



* Les dates et les Temps :

LocalDate : qui nous donne la date ;

LocalTime : qui nous donne l’heure ;



Si on veut voire l’heure dans d’autre pays, rien n’est plus simple on précise seulement la zone comme ceci :

