Programmation Fonctionnelle

ITIG ou VSC. ITIG on peut demander auprès de l’université la version payante. INTILIG. Faire attention à la version de JAVA, il faut qu’elle soit stable. Il nous encourage à prendre la dernière version du 17.1.8.

Jenv qui permet de switcher la version de java. Il faut l’installer. Il nous faut également gradle et un terminal. Activer aussi la sauvegarde automatique des IDE.

# Qu’est-ce que c’est ?

La programmation fonctionnelle est une façon de programmer différente de ce qu’on a pu voir.

**Paradigme impératif** a été fait par java, pascal. En C, on a eu un paradigme impératif auquel on a ajouté les pointeurs.

**Le paradigme objet,** on est dans un paradigme où on a voulu représenter les objets. L’aspect objet en java nous fera penser le monde en terme objet.

**Paradigme de la programmation à base de contrainte** : Prolog. Paul est le père de Jean et Jean le père de Pierre. Donc Paul est le grand-père de Pierre.

**Paradigme fonctionne**l c’est pensé un programme par ce qu’on assemble. Depuis JAVA 8, le JAVA a intégré des éléments fonctionnels. La programmation fonctionnel est plus déclarative, ce qu’on veut que comment le faire.

**Les particularités de la fonctionnel :**

* Les fonctions sont totales, déterministes, absence de réflexion, d’exception.
  + ***Qu’est-ce qu’est la totalité*** ? Elle est totale lorsqu’elle est définie sur ce qu’elle prend en entrée. Il n’y a pas de valeur pour lequel le prog plante. On a une valeur qui est toujours retourner. On verra à la fin du cours le type Option qui rend ce qui n’est pas totale, totale.
  + ***Qu’est ce que déterministe***? Si on fourni les mêmes entrées on obtiendra les mêmes sorties. C’est vrai en ASQUEL. Dans JAVA ce n’est pas un langage qui n’est pas totalement fonctionnelle. On peut étendre le terme déterministe. Pour comprendre ce qui est déterministe, on prend le contraire. Si je fais une entrée sortie, je vous demande de taper deux fois un caractère, on aura alors pas forcément la même sortie pour les deux.
  + ***Qu’estce que pure***? La fonction prend ses entrées et fourni un résultat, elle n’est pas censée modifier autre chose. Mettre un point à l’étudiant, je modifie la mémoire, c’est un effet de bord. A partir du moment que je peux modifier la mémoire, ca va poser pb lorsqu’elle est collective.
  + ***Pas de réflexion***
  + ***Pas d’exception.***

Les avantages :

La programmation fonctionnelle est fortement typée. Tout l’idée de la programmation fonctionnelle c’est que les problèmes soient visibles devant le programmeur. On peut faire des accès à la mémoire sans que cela pose pb.

Une certaine rigueur. Il y a moins de base en fonctionnel qu’en impératif.

# Génériques

Héritage, implantation, transitage(cast), la surcharge(overload) et le masquage (@override = redéfinir une méthode sur un fils et de l’appeler avec le fils), énumération (peut implémenter une interface, c’est une classe dont les itérations sont limités), sous-classe, super-classe, classe mère, classe fille, hiérarchie d’héritage.

List<Integer> xs = new ArrayList<>();

Xs.add();

Int s ;

For (int I = 0; i<xs.size(); i++){ /for(Integer v: xs)

S+=xs.get(i); / s+=v;

}

JEP Café José Paumard, il prend un concept et il va nous parler de ce concept pendant 15 min en JAVA. JAVA8 in action Mario Reda. Effective JAVA joshua Blok

Boxing in boxing : on le reverra dans java Stream lorsqu’on aura des listes d’entier. Classe StreamInteger.

Class Nom<T1,T2,…,Tn>{}

Il faut que les types aient quelque chose en commun dans les génériques pour simplifier.

R=0 ;

For (A f : fs){

R+= f.size() ;

}

Return r

List<A> notes = new ArrayList<>(); //Java est un langage typé et va deviner que c’est le type A.

On peut aussi écrire new ArrayList() ; C’est du Raw type, java l’accepte mais va nous alerter. Déconseiller.

Si **private,** l’instance de la classe. **Protected** les classes filles**. Public** c’est tout le monde. **Rien**

**Static :** il y a un espace mémoire qui est commun à toutes les instances de la classe. Si je mets static devant une méthode alors elle devient commune à toutes les instances de la classe. Les paramètres de généricité parlent au niveau de l’instance.

**Final**: on ne peut pas réaffecter. Une fois qu’on l’a initialisé on ne peut pas changer la valeur. Au sens héritage, pour une méthode cela nous interdit de la override.

## Typage Borne :

<T extends Number>, On ne peut pas mettre trois classe. Classe en premier puis l’interface <T extends C & I1 & I2>.

Si T extends Comparable cad que ce sont des éléments qui peuvent être triés. En java les collections sont forcément homogènes, il n’y a pas de collection hétérogène. Le joker permet de définir n’importe qu’elle classe dans le type générique. Cela permet de fixer le fait d’être en dessous de quelque chose ? extends MyClasse. A l’inverse : ? super MyClasse. C’est lié à la collection. Quand la collection doit contenir des objets de type T elle va appeler les méthodes de type T.

List<B> y=… ;

List <A> x = y ; // Cela ne marche pas, sauf List<A> x= new ArrayList<B>() ;

La convariance c’est dans la même direction vers le haut. A contra-variance c’est l’inverse. L’invariance c’est java qui ne sait pas faire.

## Type erasure : effacement type.

Problème de pont, on a dupliqué une méthode que le compilateur a créé.