Programmation Orientée Objet

# Chapitre 1

Une classe, c’est un type de données abstrait, elle représente une entité, on doit dire ce que l’on doit mettre dedans (de la donnée : un état (d’un employé etc)).

On encapsule pour éviter qu’un autre développeur affecte une donnée erronée à une variable par exemple.

Ex : - en public : emp.salary = -1 ; mauvaise donnée mais possible

- en private : emp.salary= - 1 ; -> impossible

On doit utiliser un setter : emp.setSalary(-1) -> on peut mettre un if sur ce setter lorsqu’on implémente la variable pour qu’il soit toujours positif et ce sera impossible d elui affecter -1.

Constructeur trivial : constructeur créé par défaut si on n’en a pas.

## I/ Classe

Modificateurs :

- droits : - public (pour tout le monde), - package (que pour le package)

- autres : - final (interdit aux autres classe de dériver cette classe), - abstract (classe qu’on ne peut pas instancier (exemple : fait le même effet si elle a un constructeur privé, des fonctions vides, etc))

- protected : classes dérivées et aussi dans le même package (signifie aussi package)

- static : peut être utilisé sans instancier la classe

Fonction lambda, fonction définie dans une autre fonction. Si on définit une fonction anonyme, on est obligé de déclarer une variable qu’on veut réutiliser en final.

Propriétés :

- constructeurs : fonction membre d’une classe qui intialise des données, cette fonction se reconnait car elle est non-typée. 3 types de constructeurs : - par défaut (sans paramètres) - d’initialisation (String name) - par copie (Employee empl)

Bloc d’initialisation : exemple : public class X {

🡪 si la classe n’a pas de nom (anonyme) : {y = new X();}

🡪 si on n’a pas d’objet : static {z = new Z()}

}

Ils servent aussi à initialiser une donnée dans une classe.

Sert à faire un constructeur pour une classe qui n’a pas de nom.

Méthode equals :

public class Student {

private String name ;

private Double avg ;

public boolean equals (Objetct o){

if ( o instance of Student){ 🡨 pour éviter qu’on compare l’objet avc une string

Student s = (Student) o ; 🡨 on caste l’objet o en Student

if (s.name == name && s.avg == avg) {

return true ;

}