


Ecriture des classes (suite)

Les méthodes dites
 « polymorphes » :
 equals et hashCode

XH1



La classe Object

Object

+ Object()
+ getClass() : Class
+ hashCode() : int
+ equals(arg0 : Object) : boolean
clone() : Object
+ toString() : String
+ notify() : void
+ notifyAll() : void
+ wait(arg0 : long) : void
+ wait(arg0 : long, arg1 : int) : void
+ wait() : void
finalize() : void

MaClasse

- propriete1
- propriete2

+ equals(o : Object) : Boolean
+ hashCode() : int
+ toString() : String

XH2



Les méthodes d'Object à redéfinir dans les objets métiers

nom de méthode polymorphe	explications
toString	renvoie une chaîne de caractères décrivant l'état de l'objet courant
equals	renvoie true si deux objets distincts sont égaux
hashCode	renvoie un code de « hachage » entier

XH

3

3



Autres méthodes d'Object

nom de méthode	explications
getClass	renvoie une instance de la classe java.lang.Class qui conceptualise la classe d'un objet
wait et notify	utilisées pour la synchronisation des threads
clone	crée une copie d'un objet
finalize	redéfinie dans une classe pour décrire les traitements spécifiques à effectuer à la destruction d'un objet par le ramasse-miettes

XH

4

4



La méthode toString

- Dans les objets métiers, on peut redéfinir toString
- N'oubliez pas l'annotation @Override
- Sécurité pour la bonne signature de méthode

XH

5

5



La méthode equals

- Dans les objets métiers, on peut redéfinir equals
 - `public boolean equals(Object obj);`
 - Exemple: deux livres sont égaux s'ils ont le même titre et même nombre de pages
- Une collection de type Set va automatiquement empêcher le stockage des objets égaux (au sens d'equals) (doublons)
- « equals » est réflexive, commutative et transitive
- Redéfinir equals implique la redéfinition de hashCode en suivant la même logique métier

XH

6

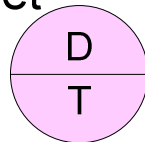
6



Le hashCoding

La méthode hashCode

objet



- hashCode() fournit pour chaque objet, un entier dit « code de hachage »
- sert au placement (et la récupération) de l'objet dans les collections de type HashSet, HashMap, TreeSet , ...
- Les qualités de cet algorithme :
 - rapidité
 - Qualité statistique: les codes de hachage des objets doivent être répartis de manière homogène sur l'ensemble des entiers int

XH

7

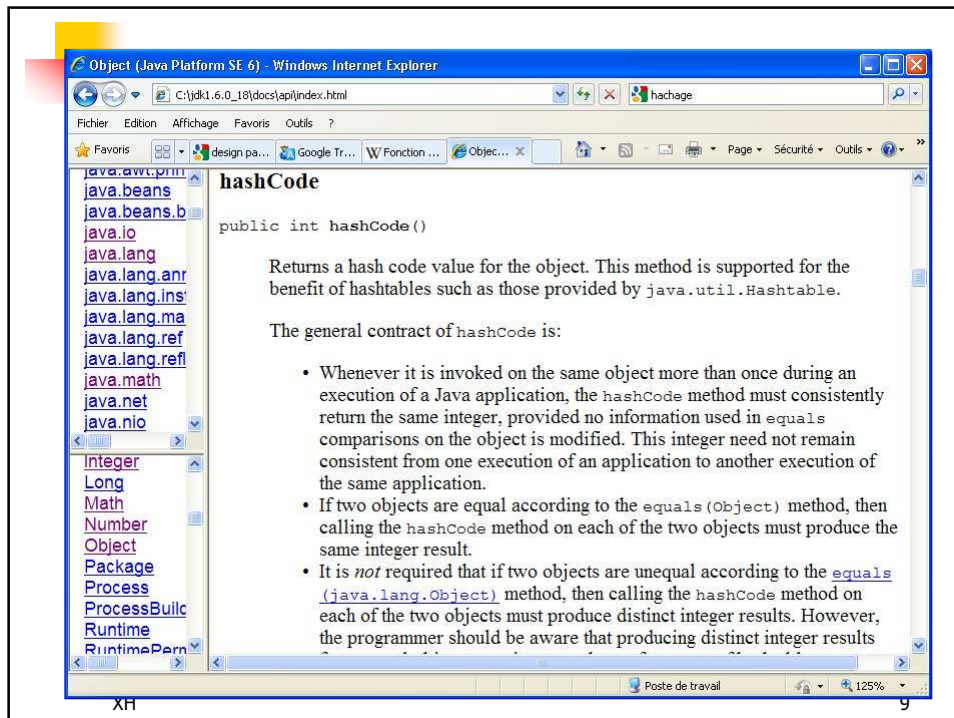
7



XH

8

8



9

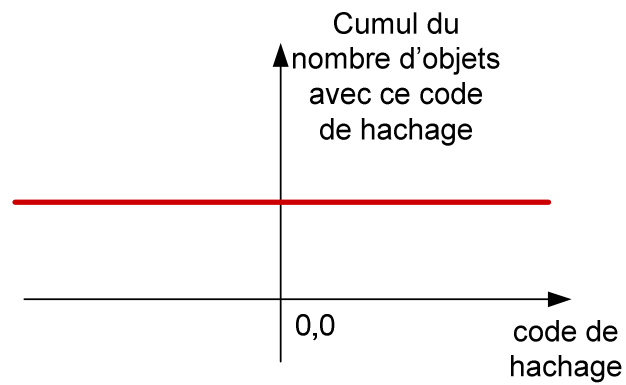
La méthode hashCode

- Redéfinir la méthode hashCode dans votre classe métier
 - public int hashCode();
- Si deux objets sont égaux (au sens d>equals) alors hashCode doit retourner le même entier
 - Mais, deux objets peuvent renvoyer un code de hachage identique tout en étant différents au sens equals !!!
- Règle à tenir pour un bon fonctionnement des collections avec vos objets:
- **redéfinir equals implique la redéfinition de hashCode en suivant la même logique métier**

10



Qualité statistique de l'algorithme de hachage



- Les codes de hachage des objets doivent être répartis sur l'ensemble des entiers

XH

11

11



END



XH

12

12