|  |  |  |  |
| --- | --- | --- | --- |
| Le Mot | L’utilisation | Remarque | Utilité |
| Final | Indique que la classe, l’attribut ou bien la méthode ne peut pas être modifié dans la suite du programme | Dans le cas d’une méthode, cette méthode ne sera pas redéfinie dans les classes dérivant de cette méthode. |  |
| Abstract | C’est une classe que le client ne peut pas créer une instance à partir d’elle. | Pour déclarer une méthode abstract i faut déclarer la classe comme abstract. | Interdire l’instanciation |
| Noninstaciable | C’est une classe |  |  |
| Static | Définir qu’une méthode ou un attribut peut être utilisé sans avoir besoin d’instancier la classe. |  |  |
| Subclass | C’est une classe qui hérite d’une autre class |  |  |
| Singleton | Le premier design pattern : 1-créer une unique instance ;2-Assurer l’accés public à cette instance | Utiliser la technique de synchronisation(Local ou Global) ou bien le holder et à base de static factory method (méthode utilitaire) |  |
| Serialisation | Convertir un objet en une serie d’octets |  |  |
| Immutable | C’est un objet qui ne peut pas changer leur état après l’instanciation |  |  |
| Adapter | Design patern consiste à adapter un type à un autre en utilisant une interface. |  |  |
| autoboxing | automatic conversion that the Java compiler makes between the primitive types and their corresponding object |  |  |
| hashcode | Méthode commun pour tous les objets, permet de transférer les valeurs d’un objet en un code en entire. |  |  |
| List | Collection ordonnées, dont l’utilisateur à le control total, il peut récupérer le n ème élément, recherché, |  |  |
| Set | Ne contient pas des éléments dupliqués, et contient au plus un élément null. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Item 1 : Static Factory Method

Une static factory method permet de déclarer une classe sans avoir besoin du constructeur, , de plus il sont préférer dans l’utilisation. Donc éviter la réflexion directe vers le constructeur sans avoir pensé au statics factory method.

## Avantages

1. Contient un nom :: éviter l’ambiguïté dans l’appel du constructeur :
2. pas besoin de redéfinir la méthode equals et pas besoin de déclarer new à chaque fois
3. Peut retourner plusieurs types exemple EnumSet : RegularEnumSet et JumboEnumSet
4. reduce the berbocity of creating parameterized type instances: exemple : Map<string, list<String >> m = new HashMap<string, list<String>>(); public static <K,V> HashMap<K,V> newInstance() {

return new HashMap<K,V>();

map<string, list<String >> =Hashmap.newinstanciate(); }

## Inconvégnant:

1. on peut pas hériter de cette class si la classe ne contient pas le mot (Public, Protected) et c’est le cas la framework Collections.
2. On ne peut pas les distinguer par rapport au d’autre méthode statique.

Quelque Static Factory method :

Valueof, of, GetInstance, NewInstance,Gettype,newtype

# Item2 Builder Pattern:

## Telescoping constructor pattern:

Déclarer simplement plusieurs constructeur ; exemple 6 champs; 5 constructeurs (2champ, 3champs,4champs,5champs,6champs)

## JavaBeans pattern :

Donner des valeurs par défaut à chaque attributs, puis lors de a déclaration utiliser les setters pour modifier les valeurs.

Builder Pattern : combiner entre les deux méthodes précédentes : CAD : créer une classe static interne, initialiser le constructeur avec les valeurs à 0, puis créer des méthodes qui retourne la classe et qui initialise un champs, puis créer une méthode qui retourne la classe initiale en utilisant le constructeur privé de la classe parent.

Et créer un constructeur privé basé sur la classe interne.

En résumé, le modèle Builder est un bon choix lors de la conception de classes dont les constructeurs auraient plus d'une poignée de paramètres, en particulier si la plupart de ces paramètres sont facultatifs. Le code client est beaucoup plus facile à lire et à écrire avec les Builder qu'avec le modèle de constructeur télescopique traditionnel, et les Builder sont beaucoup plus sûrs que JavaBeans.

# Item 3 : Singleton

## 2 méthodes ancien

1-mot final : mettre le constructeur private et déclarer un static final class en interne avec ce constructeur. (Utilisateur privilégie peut accéder au constructeur en utilisant la méthode **AccessibleObject.setAccessible**

2- Static factory method : avantage (faciliter la personnalisation de l’utilisation(unique instance pour chaque thread),

Afin d’éviter que un autre objet sera créer si la classe implement serialisable, il faut ajouter la méthode readresolve() qui va retourner l’objet elle-même.

3- la meilleur approche est d’utiliser une énumération avec un seul élement. car elle fournit la machine de sérialisation gratuitement, et fournit une garantie à toute épreuve contre l'instanciation multiple,

# Item 4 : uninstanciable classe

Il suffit d’ajouter le mot private au niveau du constructeur de la classe voulu.

# Item 5 : avoid creating unnecessary object

Si la création d’une classe ne nécessite pas l’utilisation de constructeur, on évite d’utiliser celle-ci. Exemple : String.

Utiliser les Item1 au lieu du constructeur pour la création des immutable classe.

Ne pas déclarer les variables qui ne sont pas modifiables au niveau des méthodes mais sera mieux de les déclarer au niveau de la classe en tant que statique.

L’utilisation de l’auto boxing permet parfois de résoudre le problème d’adaptation.

“Don’t create a new object when you should reuse an existing one”

# Item 6 : eliminate obsolute object

1-Exemple de la partie non null du tabeau pour le garbage, et qui est considérer obsolète pour le programmeur.

2-la 2 source de la perte de mémoire est le cache

3-la 3 sources se sont les listener et les callbacks

# Item 9 : Override Hashcode when you override equals

Il est obligatoire que 2 object qui sont égaux par la méthode Equals, doivent returner le meme entier.

Le contraire n’est pas obigatoire mais il peut améliorer les performances des **Hash-Based** .

Dans les hash-base, si le hashcode est différent alors java ne compare pas avec equal, il prend que les 2 objets sont différents

Dans les classes immutables on peut définir le hahcode au-dedans de la classe, si le cout de calcul du hashcode est important, ceci en déclarant le variable comme volatile, si le variable egale à 0, on calcule le hashcode sinon, on return le hashcode précédent.

# Item 10 : Override ToString

Il faut redéfinir la methode toString pour que le code soit le plus lisible possible.

Lorsque vous appeler les fonctions d’affichage, la fonction toString est automatiquement appelé.

Vous pouvez définir le format de la description affichée lors de l’invocation de la méthode to-string en utilisant la méthode ( **String.format()** ).

# Item11 Override Clone judiciously

clones() est diffèrent de X ;

X.equals(X.clone())must return true

A méthode clone doit appeler super.clone (), jusqu’à l’arrivée à la classe object, elle doit aussi être public et retourner la classe elle -même.

Il faut être attention lors du clonage d’une classe contenant une liste, ou tableau, ou hashmap (exemple du clone de la classe hashtabe, et de la pile)

La méthode clone ne doit pas invoquer une méthode non final, car le clonage peut ne pa être terminer

Si une classe est déclarer pour être hérité, elle doit suivre le même comportement de la classe (Object: Engendre l’exception clonenotsupportedexception) et elle ne doit pas implémenter l’interface clonable.

Si on doit utiliser un Safe Thread class, on doit déclarer un synchronized Clone method.

On peut copier on utilisant CopyConstructor ou bien CopyFactory, cela donne plusieurs avantages, mais elle ne peut pas jouer le rôle d’une interface.

Vous pouvez transformer une HashSet en TreeSet, en utilisant CopyConstructor ou CopyFactory.

Les interfaces ne doivent pas héritées de cloneable et les classes conçues pour l’héritage aussi. Les experts n’utilisent jamais cloneable sauf pour le clonage des tableaux.

# Item 12 : Consider Implementing Comparable

La fonction compareto engendre une exception de type ClassCastException, ou bien retourne -1,0,1.

1-sgn (x.compareTo(y)) == -sgn(y.compareTo(x))

2-(x.compareTo(y) > 0 && y.compareTo(z) > 0) implies x.compareTo(z) > 0.

3-x.compareTo(y) == 0 implies that sgn (x.compareTo(z)) == sgn(y.compareTo(z))

4-(x.compareTo(y) == 0) == (x.equals(y))

Les collections trié utilise le test CompareTo au lieu de Equals (exemple HashSet et TreeSet avec la classe bigdicimal)

Faire attention si la différence est calculé à partir de deux entier négatifs, ou que la différence soit plus grande que le rang integer 2\*31-1

# Item 13 : minimize accessibility to Class and its members

C’est le principe de « hidding information » ou « encapsulation », ce principe est parmi les avantages et les bonnes pratiques du programmeur, car il assure l’isolation des modules, ce qui rend la programmation plus rapide (Plsr Module en parallèle) et plus facile à maintenir.

Mettre chaque classe le plus possible inaccessible

Si une méthode est hérité d’une classe parent, on ne peut pas avoir un niveau d’accès plus bas que celle de la méthode parent.

Une instance d’une sousclasse doit avoir un niveau d’accès plus haut que l’instance de la superclasse

**Remarque :** Ne jamais déclarer un attribut ou un objet mutable comme public.(Exemple de la table de String)

# Item 14 : Use accessors method and mutators

Afin de facilité la modification de la représentation interne de la classe.

Parfois, il est plus simple de déclarer les attributs comme public, dans le cas des private-package et les classe imbriqué ‘Nested-Class’.

**Remarque :** Pour les attributs static final, pas de problèmes pour leur public déclaration.