

Programmation Orientée Objet Avancée

Arnaud Lewandowski

A series of horizontal lines of varying lengths and shades of blue and white, extending from the right side of the slide.

Contenu du module

- Code conventions
- Refactoring
- Réflexivité, Introspection, Méta classes
- Modèles de conception (*Design patterns*)

Code Conventions

Code Conventions

- Pourquoi s'embêter avec ça ?!
 - 80% du coût de la durée de vie d'un soft est consacré à la maintenance
 - Rare sont les soft maintenus par leur auteur
 - Améliorent la lisibilité
(=> compréhension + rapide et + complète)
 - Produit de qualité, Professionnalisme

Code Conventions

- Références :

- Java :

- <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>

- Linux Kernel :

- <https://www.kernel.org/doc/html/v4.10/process/coding-style.html>

- GNU :

- <http://www.gnu.org/prep/standards/standards.html>

Quelques conventions (1)

- Une seule classe ou interface publique par fichier
- Unité d'indentation = 4 espaces
- Lignes de 80 caractères max
 - Passage à la ligne
 - Après une virgule
 - Avant un opérateur
 - Indentation:
 - Au début de l'expression du même niveau
 - Ou 8 espaces

Quelques conventions (2)

- Commentaires
 - Dans le code, uniquement des infos nécessaires à la compréhension
 - Sinon = javadoc
- **Remarque** : si vous sentez le besoin d'ajouter un commentaire, réécrivez le code +simplement et +clairement

<http://www.codinghorror.com/blog/2008/07/coding-without-comments.html>

Quelques conventions (3)

- Déclarations
 - Variables
 - Uniquement en début de bloc
 - Une seule variable par ligne
 - Initialiser lors de la déclaration si possible
 - Méthodes
 - Pas d'espace entre un nom de méthode et la parenthèse (qui suit
 - Classes, if, for, while, ...
 - L'accolade { à la fin de la ligne
 - L'accolade } seule sur une nouvelle ligne

Quelques conventions (4)

- Un espace
 - Après la virgule
 - Entre un mot-clé et une parenthèse (
 - Autour de tous les opérateurs (sauf ++, --, .)
 - Avant/après une accolade
 - Après un cast

Quelques conventions (5)

```
public class MaBelleClasse extends Object {  
  
    static final int MA_CONSTANTE = 8;  
  
    int monAttribut;  
  
    public MaBelleClasse() {  
        // ... implémentation  
    }  
  
    public Object faitQuelqueChose() {  
        // ... implémentation  
    }  
  
}
```

Quelques conventions (6)

- Liste non exhaustive
 - <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>
 - <https://www.securecoding.cert.org/confluence/display/java/Java+Coding+Guidelines>
- **Le principe : avoir du code PROPRE**
- Pour vérifier : <http://checkstyle.sourceforge.net/>
- Voir aussi PMD (contrôle du respect des bonnes pratiques)

Refactoring

Le Refactoring (1)

- Principe :
 - Modification du code sans ajout de fonctionnalité
- Objectif :
 - Améliorer la lisibilité
 - Rendre le code plus maintenable/évolutif
- Réorganisation / restructuration du code

Le Refactoring (2)

- En principe
 - On ne change pas un code qui marche
 - Surtout si aucune fonctionnalité en plus
 - Risque d'introduction de bugs
- MAIS
 - Dev. itératif incrémental, corrections de bugs
 - Les modifs successives complexifient le code
 - On n'a pas le choix !

Le Refactoring (3)

- NB: Le refactoring n'est pas une solution miracle...
 - Utilisation de tests unitaires (non régression)

Les niveaux de Refactoring

- Modification de la présentation
 - Commentaires, mise en page, etc.
- Modification de l'algorithmique
 - Objectif : méthodes aussi simples que possible
- Relocalisation de procédures
- Refonte de la conception
 - Modification de la hiérarchie de classes

Activités de Refactoring (1)

- Suppression du code mort
 - Détection de code non utilisé: *grep*, analyseur de réf. croisées, outil de mesure de couverture de code
 - Le code commenté « au cas où »
- Ajout d'assertions
 - Règles à respecter
 - Facilitent le débogage, aident à la compréhension
- Renommage
 - Rôles des méthodes / classes
- Commentaires

Activités de Refactoring (2)

- Extraction (classe, méthode, variable, constante, interface)
- Conversion (variable locale vers attribut)
- Déplacement (vers une sous-classe, une super-classe, ou une autre classe)
- Encapsulation
- Généralisation
- etc.

Le Refactoring : Références

- <https://www.jmdoudoux.fr/java/dejae/chap009.htm>
- <http://blog.excilys.com/2010/08/03/refactoring-par-la-pratique/>

TP/Tutoriel : découverte de PMD...

TP : refactoring

<http://blog.excilys.com/2010/08/03/refactoring-par-la-pratique/>

<http://www.cs.virginia.edu/~horton/cs494/s05/slides/lab-exercise-refactoring.htm>