INF2050 – Outils et pratiques de développement logiciel

Code propre

Jacques Berger

Objectifs

Écrire du code propre

Prérequis

Programmation

Nomenclature

Révéler l‘intention

Éviter la désinformation

Nom prononçable / sans encodage

Facile à rechercher

Nomenclature

Un nom de classe devrait débuter par un nom

Un nom de méthode devrait débuter par un verbe

Aucun synonyme

Utiliser la terminologie du domaine

Fonction

Courte

Ne fait qu‘une seule chose

Un seul niveau d‘abstraction

Nom descriptif

Fonction

Limiter les paramètres (max. 3)

Aucun flag dans les paramètres

Gérer les erreurs avec des exceptions lorsque le langage le permet

Fonction

DRY : Don‘t Repeat Yourself

switch

Un switch en particulier ne devrait apparaître qu‘à un seule endroit dans le code

Penser à l‘encapsuler dans une classe

Commentaires

Bons commentaires

Commentaires légaux
Information utile
Clarification
Avertissement d‘un danger
TODO
Javadoc pour une librairie

Commentaires

S‘expliquer avec du code clair plutôt que des commentaires

Éviter la redondance (DRY)

Ne pas mettre de code en commentaire

Commentaires

Inutile sur une accolade fermante

Aération

Aération verticale

Aération horizontale

Lignes directrices

Analogie de la fenêtre brisée

DRY – Don‘t Repeat Yourself

YAGNI – You Ain‘t Gonna Need It

Minimiser le couplage entre les composants

Lignes directrices

Le programme parfait n‘existe pas

Concevoir pour faciliter les tests

Faire du refactoring fréquemment

Refactoring

Modifier, améliorer, réécrire une partie du code dans le but de le rendre plus facile à lire et à comprendre

Ne change pas la fonctionnalité

Dette technique

Analogie utilisée pour faire comprendre aux gestionnaires le prix à payer pour un code malpropre

SRP

Single Responsibility Principle

Une classe ne devrait avoir qu‘une seule responsabilité; une seule raison de changer

Loi de Demeter

Principe de connaissance minimale

Sert à minimiser le couplage

Smells

Un smell est un élément que l‘on remarque dans le code qui semble indiquer un problème de réalisation ou de conception

Un smell ne doit pas être évité à tout prix, mais c‘est souvent signe qu‘il est possible d‘améliorer le code

Smells

Le build requiert plus qu‘une étape

Fonction avec trop d‘arguments

Fonction avec un flag

Smells

Un bout de code au mauvais niveau d‘abstraction

Trop d‘information dans une interface

Du code mort

Smells

Inconsistance

Responsabilité mal placée

Un nom de fonction qui ne dit pas ce qu‘elle fait

Smells

Une fonction qui fait plus qu‘une chose

Plus loin...

Clean Code
A Handbook of Agile Software Craftsmanship
Robert C. Martin
Prentice Hall, 2009
Chapitres 2 à 5

The Pragmatic Programmer
Andrew Hunt, David Thomas
Addison-Wesley, 1999