Patrons de Conception

-

Introduction

Simon Urli
urli@i3s.unice.fr

Master 1 MIAGE
2014-2015

Motivation

- Faire une conception et un développement de qualité :
  - Extensibilité
  - Flexibilité
  - Maintenabilité
  - Réutilisabilité
  - Clarté

Motivation (suite)

- Aucune recette «magique» pour la conception ...
- ... mais quelques bonnes pratiques :
  - **KIS** : Keep It Simple
  - **DRY** : Don't Repeat Yourself
  - **YAGNI** : You Ain't Gonna Need It
  - **SOLID**
- Patron de conception : un **outil** supplémentaire

SOLID

- **S**ingle responsibility principle : une classe n'a qu'une seule responsabilité (ou préoccupation).
- **O**pen/closed principle : une classe doit être ouverte à l'extension (par héritage, par exemple) mais fermé à la modification (attributs privés, par exemple).
- **L**iskov subtition principle : les objets d'un programme doivent pouvoir être remplacés par des instances de leurs sous-types sans «casser» le programme.
- **I**nterface segregation principle : il vaut mieux plusieurs interfaces spécifiques qu'une unique interface générique.
- **D**ependency inversion principle : il faut dépendre des abstractions, pas des réalisations concrètes.

Patron de conception, quésaco ?

“ Généralisation d’une solution à un problème de conception récurrent par la description des classes et objets communicants ”

Patron de conception : comment sont-ils définis ?

- **Identification** d'un problème de conception récurrent
- **Schématisation** du problème de manière générique
- **Description d'une solution** sous la forme d'un patron

Patron de conception : comment les utiliser ?

- **Identifier un problème** dont le motif a fait l'objet d'une solution
- **Rechercher** le patron de conception adapté
- **Appliquer et adapter** la solution proposée par le patron de conception

Historique

- **1977/79** : Christopher Alexander - Patron de conception pour l'architecture des villes et bâtiments
- **1987** : Kent Beck et Ward Cunningham - Papier à OOPSLA sur l'utilisation de patrons de conception pour la programmation orientée objet
- **1994** : le «Gang of Four» (GoF) (Gamma, Helm, Johnson and Vlissides) publie Design Patterns: Elements of Reusable Object-Oriented Software - Présentation des 23 patterns fondamentaux

Classification

- Patrons de **création** :
  - Dédiés à la création des objets.
  - Indépendance entre création et utilisation des objets.
- Patrons de **structure** :
  - Dédiés à la composition des objets.
  - Conserver une bonne séparation des préoccupations.
- Patrons de **comportement** :
  - Dédiés à la communication entre les objets
  - Conserver de la flexibilité dans les liens de communication.

23 Patrons fondamentaux

- Patrons de création :
  - **Abstract Factory**, **Builder**, **Factory method**, Prototype, **Singleton**
- Patrons de structure :
  - **Adapter**, Bridge, Composite, **Decorator**, Facade, Flyweight, Proxy
- Patrons de comportement :
  - Chain-of-responsibility, Command, Interpreter, Iterator, Mediator, Memento, **Observer**, State, **Strategy**, Template Method, Visitor

Autour des patrons de conception

- Patrons d'architecture (ex : MVC, layers, etc)
- Patrons de gestion de la concurrence (pool de threads, etc)
- Patrons GRASP (General Responsibility Assignment Software Patterns)

Sources

- **Les design patterns en Java** - Steven John Metsker, William C. Wake
- **Head first design patterns** - Eric et Elisabeth Freeman
- **The design pattern Smalltalk Companion** - Sherman R. Alpert, Kyle Brown, Bobby Woolf
- http://blog.codinghorror.com
- http://en.wikipedia.org/wiki/Software_design_pattern