INF5153 – Génie logiciel : conception

Patrons SOLID

Jacques Berger

Objectifs

Introduire les principes de conception SOLID

Prérequis

Aucun

SOLID

SOLID

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

SOLID

Cinq principes de base de la conception orienté-objet

SOLID

Ces principes permettent de créer des systèmes plus faciles à maintenir et étendre

Guides pouvant être appliqués au quotidien pour réduire les erreurs de conception

SRP

Single Responsibility Principle

Chaque classe ne devrait avoir qu'une seule responsabilité

Un seul morceau de fonctionnalité du logiciel

SRP

La responsabilité doit être encapsulée dans la classe

Tous les services de la classe doivent être alignés dans le but d'accomplir cette responsabilité

SRP

Basé sur le principe de cohésion

SRP

Robert C. Martin indique qu'une responsabilité est une raison de changer

Une classe ne devrait avoir qu'une, et une seule, raison de changer

OCP

Open/Closed Principle

Les entités logicielles doivent être ouvertes à l'extension et fermées aux modifications

Entités logicielles : classes, modules, fonctions

OCP

Permet d'étendre le comportement des entités sans changer leur code source

OCP

Avec les classes, on utilise l'héritage pour réaliser ce principe

LSP

Liskov Substitution Principle

Toute classe dérivée d'une classe de base peut être utilisée à la place de la classe de base

Sans altérer les propriétés souhaitées du programme

LSP

Nom formel : Strong Behavioral Subtyping

Garantir l'interopérabilité sémantique des types dans une hiérarchie d'héritage

ISP

Interface Segregation Principle

Aucun client ne devrait dépendre des méthodes qu'il n'utilise pas

ISP

Séparation des interfaces volumineuses dans le but d'en créer plusieurs plus petites qui ne contiennent que les réelles préoccupations des objets qui les implémente

ISP

Favorise le découplage des classes

Similaire au patron GRASP Forte Cohésion

DIP

Dependency Inversion Principle

Les modules de haut niveau ne doivent pas dépendre des modules de bas niveau

Les deux doivent dépendre d'abstractions

DIP

Les abstractions ne doivent pas dépendre des détails

Les détails doivent dépendre des abstractions

Plus loin...

Agile Software Development, Principles, Patterns, and Practices
Robert C. Martin
Pearson, 2002