Introduction au Génie Logiciel

Séance 7 : Qualité et design patterns

L. Laversa
laversa@irif.fr

Université Paris Cité

18 mars 2025

. Remerciements à E. Bigeon et J. Lefebvre

Diagramme de classe

Éléments :

- Classes, avec attributs de type simple et méthodes
- Association entre classes

Pas d'attribut dont le type est une classe du diagramme, mais une relation !

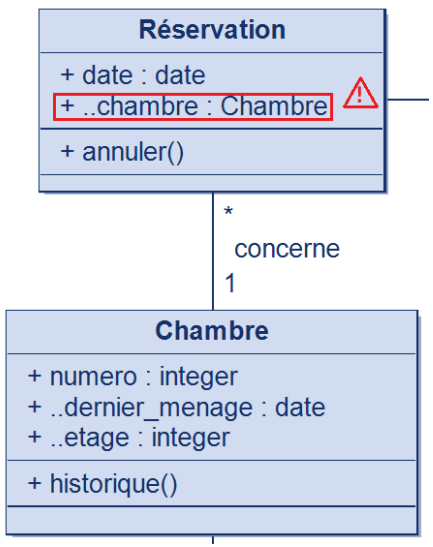
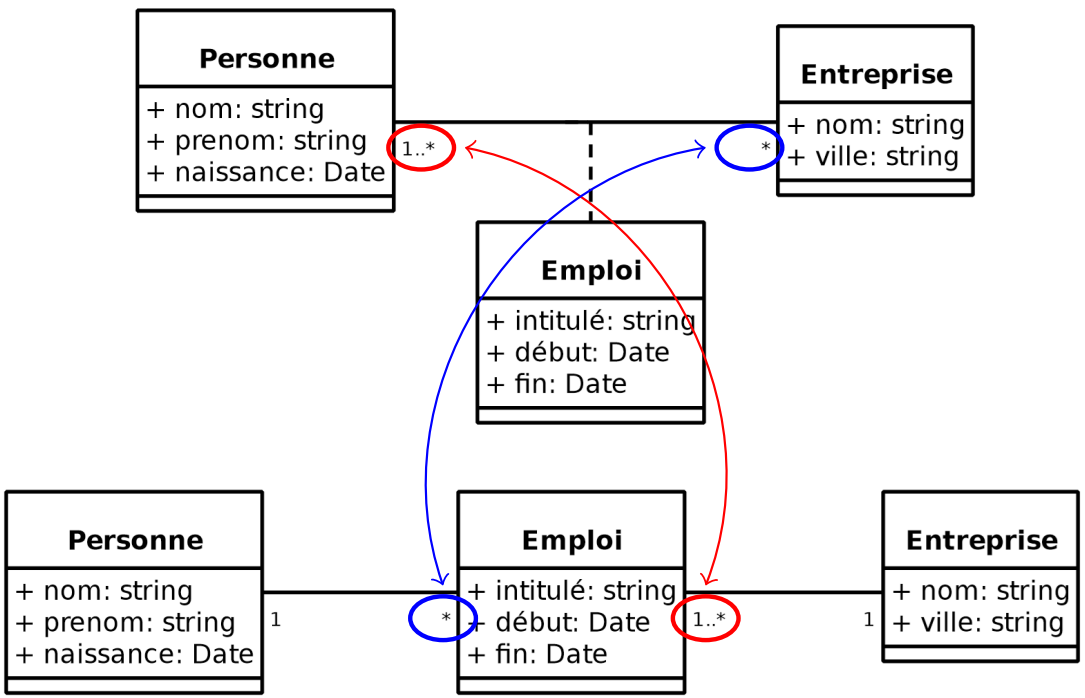


Diagramme de classe

Si une classe correspond exactement à une relation entre deux autres classes, elle devient une **Classe-association**.



Attributs de qualité

Norme ISO25010 définit les attributs de qualité

- **Validité** : Capacité du logiciel à fournir les fonctions attendues
- **Performance** : Temps de réponse, utilisation des ressources, capacité de mise à l'échelle
- **Compatibilité** : Capacité à fonctionner avec d'autres systèmes ou environnements
- **Utilisabilité** : Facilité d'apprentissage et d'utilisation
- **Fiabilité** : Stabilité, tolérance aux pannes, récupération après incident
- **Sécurité** : Protection contre attaques, respect de la confidentialité
- **Maintenabilité** : Facilité à être modifié, testé, et amélioré
- **Portabilité** : Capacité à fonctionner dans différents environnements

Intérêts

- Permettent de comparer des logiciels entre eux dans leurs caractéristiques *non-fonctionnelles*
- Donne une idée du travail nécessaire à la maintenance du logiciel dans le temps

Limitations

- Termes génériques, et pas toujours testables dans l'absolu
- Un même aspect peut être classé en plusieurs catégories
- Vocabulaire spécifique à chaque catégorie, qui ne facilite pas la prise en compte de différents attributs en même temps

Design Patterns

Les patrons de conceptions sont des solutions connues et déjà testées pour des problèmes récurrents en développement.

Design Patterns

Les patrons de conceptions sont des solutions connues et déjà testées pour des problèmes récurrents en développement.

Diminuent le temps de développement.

Améliorent la qualité logicielle.

Avantages

Point de vue développeur :

- Solutions génériques et extensibles : qu'on peut réutiliser
- On ne réinvente pas la roue : on gagne du temps
- Standardisation : langage commun

Qualité logicielle et patrons de conception

- Maintenabilité :
  - Code structuré, modulaire et plus facilement compréhensible
  - Séparent les responsabilités (tests unitaires + simples)
- Performance :
  - Optimisation de la gestion des ressources et l'exécution
- Sécurité :
  - Structure limitant les failles
- Fiabilité :
  - Diminutions du risque de bug en utilisant des solutions éprouvées

GoF [1] et catégories de design patterns

Définis et nommés dans

> *Elements of Reusable Object-Oriented Software*
> par E. Gamma, R. Helm, R. Johnson et J. Vlissides

Trois catégories :

- **Créationnel** : Gestion de la créaction d'objet
- **Structurel** : Organisation et relations entre classes
- **Comportemental** : Communication et interaction entre objets

1. Gang of Four

Design patterns - Liste non exhaustive

Créationnels	Singleton
	Factory method
	Builder
Structurel	Decorator
	Proxy
	Adapter
Comportemental	Observer
	Strategy

Exemple - Singleton

Problème à résoudre

Un service doit être accessible globalement sans créer plusieurs instances

Structure

- Un constructeur privé : empêche la création d'instances en dehors de la classe
- Une instance statique, stockée dans la classe
- Une méthode publique pour l'accès à l'instance unique (`getInstance()`)

Mise en garde

- Tout problème n'a pas de solution via un design pattern

- Plusieurs solutions peuvent être possibles :
  - Penser à l'architecture globale et/ou l'évolution

  $\Rightarrow$ Compromis entre résolution du problème local et compatibilité avec le projet

Architecture VS Design Pattern

- Architecture logicielle : Plan du projet
- Patrons de conceptions : Gestion d'un problème dans le projet

Exemples d'architecture

Modèle-Vue-Contrôleur (MVC), monolithique, en microservices, orientée évènements, ...