Génie logiciel

Notes du cours de 09/12

L3 Informatique appliquée 2022-2023

MABROUK Fayez

23 décembre 2022

1 Conception de logiciels

1.1 Définitions

* Définition : La conception logicielle est le processus de définition de la structure globale et de interaction de votre code afin que le produit résultant satisfasse aux exigences.
* La modélisation est différente de la conception !.
* La modélisation : un moyen d'exprimer la conception.
* C'est une étape importante pour se mettre d'accord.
    * sur l'utilisation du produit logiciel.
    * sur la structure du produit logiciel.

1.2 Concepts de conception de logiciels

* L'abstraction.
* Couplage et cohésion.
* Décomposition et modularisation.
* Encapsulation et masquage de l'information.
* Séparation de l'interface et de l'implémentation.
* Suffisance, complétude et primitivisme.
* Séparation des préoccupations.

1.3 Abstraction

* **Abstraction** : une vue d'un objet qui se concentre sur l'information pertinente pour un objectif particulier et ignore le reste de l'information.
* Sans abstraction, votre programme devient trop complexe et inutilisable.
* Vous devez toujours vous poser la question suivante : "Quelle est la complexité adaptée à mon objectif ?"

1.4 Encapsulation

* **Encapsulation** : cacher les détails d'une abstraction à une entité externe.
* Composant essentiel d'une abstraction, mais différent de l'abstraction.

1.5 Modularisation

* Modularisation : diviser le logiciel en petits composants, chacun ayant une interface bien définie.
* Principe "diviser pour régner".
* Réalisé grâce à l'encapsulation : ce qui n'est pas pertinent pour l'utilisateur externe est caché.

* Advantages :
    * Plus facile à maintenir.
    * Plus facile de travailler en parallèle.
    * Gère bien la complexité.

1.6 Séparation des préoccupations

* La séparation des préoccupations est un principe de conception qui stipule qu'un programme doit être divisé en sections, chacune répondant à une préoccupation différente.
* Terme inventé par Dijkstra en 1974.
* Chaque section est un module.
* Permet la réutilisation des modules.

1.7 Séparation de l'interface et de l'implémentation

* Séparation de l'interface et de l'implémentation : l'interface est publique, mais pas l'implémentation. détails de l'implémentation.
* L'interface décrit les services qu'un client de la classe peut utiliser, et comment les les demander.
* L'implémentation décrit comment ces services sont fournis.

1.8 Couplage et cohésion

* Le couplage est la mesure du degré d'interdépendance entre les modules du logiciel.
* En général, il faut viser un faible couplage.
* La cohésion est la mesure du degré de relation fonctionnelle entre les éléments du module du logiciel sont fonctionnellement liés.
* En général, il faut viser une cohésion élevée.

1.9 Suffisance, exhaustivité et primitivité

* Un composant logiciel doit être :
    * suffisant et complet : il capture toutes les caractéristiques importantes d'une abstraction et rien de plus.
    * Primitif : la conception doit être basée sur des modèles faciles à mettre en œuvre.

1.10 Patrons de conception : pourquoi ?

* La plupart des problèmes de conception de logiciels sont communs à de nombreux projets
* Pas besoin de réinventer la roue
* Permettre des solutions communes

1.11 Modèles de conception : quoi ?

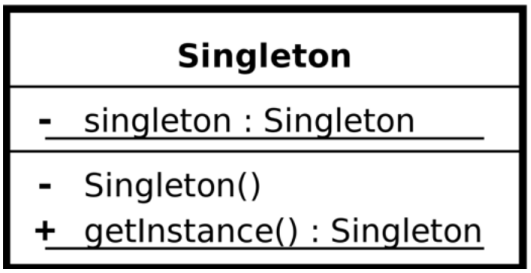
* Définition : " solution générale, réutilisable, à un problème qui se pose couramment dans un contexte donné de conception de logiciels ".
* Concept proposé par Christopher Alexander en 1977
* Formalisé (et popularisé) par le " Gang of Four " en :

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1995).
Design Patterns: Elements of Reusable Object-Oriented Software.

* 23 modèles de conception répartis en 3 catégories :
  * Creational : patterns permettant de créer un objet
  * Structurel : patterns qui composent des classes pour obtenir de nouvelles fonctionnalités
  * Behavorial : patterns qui traitent de la communication entre objets.

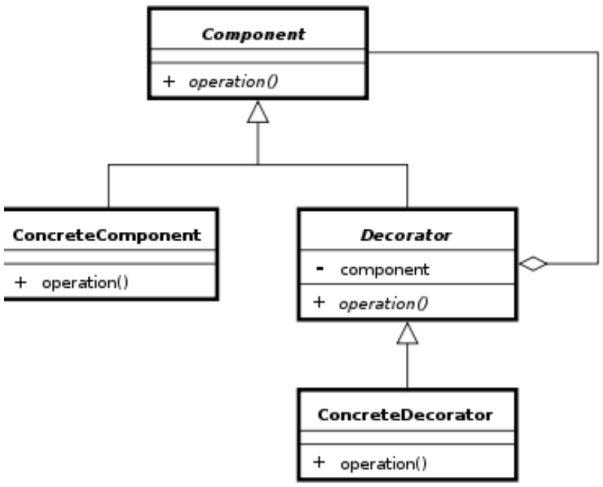
1.12 Modèles de conception créatifs - Singleton

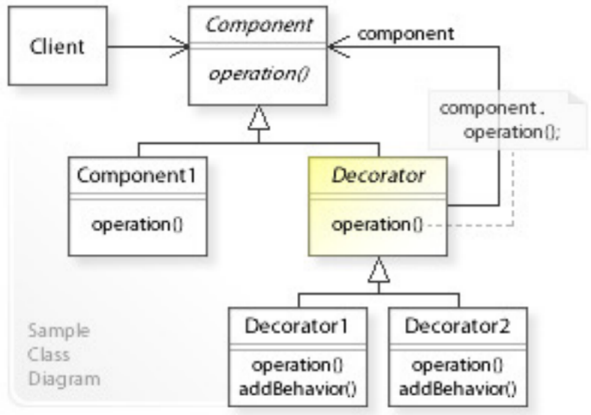
* Garantit qu'une classe n'a qu'une seule instance.
* Meilleure alternative aux variables globales.
* Exemple de cas d'utilisation : logger

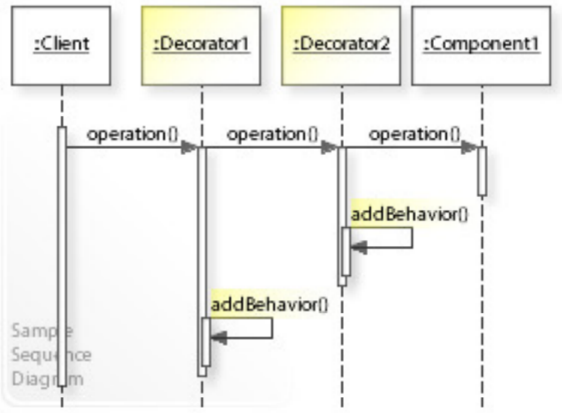


1.13 Modèles de conception structurels - Décorateur

* Ajoute un comportement à un objet individuel individuel de manière dynamique.
* Peut être considéré comme une spécialisation
* Souvent utilisé pour les interfaces graphiques.

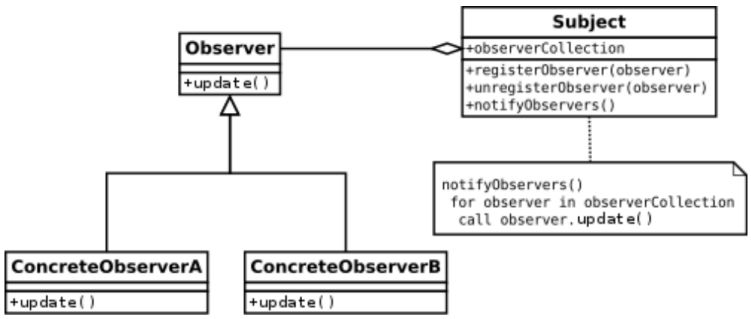






1.14 Modèles de conception comportementale - Observer

* Permet les dépendances sans couplage
* Exemple de cas d'utilisation : s'abonner pour la disponibilité d'un produit.



1.15 Critiques sur les modèles de conception

* Le langage de programmation de haut niveau apporte souvent des solutions à ces problèmes.
* Souvent, les gens essaient de trop les appliquer. Les design patterns ne sont pas les solutions à tous les problèmes.
* Une solution spécifique pourrait mieux convenir à votre projet.
* Utilisez-les avec parcimonie

1.16 Conclusion

* La conception de logiciels est une étape critique vers la création de logiciels de qualité.
* Elle permet de s'assurer que les exigences seront satisfaites.
* Permet une mise en œuvre plus facile.
* Vous devez :
    * Viser à remplir les concepts classiques de conception
    * Viser un modèle sur lequel tout le monde est d'accord.
    * opter pour les solutions les plus simples possibles