DESIGN PATTERNS - FAMILLE

Patrons structurels

Assembler des objets en définissant leurs interfaces, leurs délégations et leurs données partagées.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Comparer les intentions des patrons de cette famille.
- Choisir un premier exemple selon le problème rencontré.

Prérequis. Connaître les classes, les objets, les fonctions et les exceptions Python.

Le besoin commun

Assembler des objets en définissant leurs interfaces, leurs délégations et leurs données partagées.

Faut-il rendre une interface compatible, séparer des variations ou enrichir et contrôler une délégation ?

Commencez par le contrat du client. Décrivez les entrées, le résultat, les erreurs attendues et ce qui doit pouvoir changer. Le nom d'un patron aide ensuite à discuter la solution et ses compromis.

Les patrons à étudier

- **7. Adapter**

  Traduire une interface existante vers les appels et les données attendus par un client.

- **8. Bridge**

  Séparer deux axes de variation en reliant une abstraction à un objet d'implémentation.

- **9. Composite**

  Traiter des feuilles et des groupes avec une opération commune et une agrégation récursive.

- **10. Decorator**

  Ajouter des responsabilités à un objet en préservant son interface par délégation.

- **11. Facade**

  Coordonner plusieurs composants derrière une opération simple pour le client.

- **12. Flyweight**

  Partager un état intrinsèque réutilisable en gardant l'état de chaque usage à l'extérieur.

- **13. Proxy**

  Contrôler l'accès à un service tout en exposant la même interface.

Version web : https://www.clearcode.fr/courses/programming/design-patterns/structure