Fiche Résumé – Design Pattern Visitor

Définition:

Le Visitor permet d'**ajouter des opérations** sur une hiérarchie d'objets **sans modifier leurs classes**, en centralisant les comportements dans des objets visiteurs.

1 - intention du pattern

Ajouter de nouvelles opérations sur une hiérarchie d'objets sans modifier leur code.
→ Séparer la structure des données des opérations appliquées à ces données.

2 - Type

Pattern comportemental (Gang of Four).

3 - Exemple métier

- Une hiérarchie de formes géométriques : Cercle, Carré, Triangle.
- On veut ajouter différentes opérations :
  - Calculer l'aire
  - Dessiner à l'écran
  - Exporter en JSON

→ Sans Visitor, on devrait modifier chaque classe à chaque nouvelle opération.
→ Avec Visitor, on crée simplement un nouveau Visitor.

4 - SOLIDité du Visitor

- Open/Closed Principle (OCP) : on peut ajouter de nouvelles opérations en créant un nouveau visiteur, sans modifier les classes existantes.
- Single Responsibility Principle (SRP) : séparation claire entre données (éléments) et comportements (visiteurs).

5 - Avantages

- Ajout facile de nouvelles opérations.
- Séparation nette entre structure de données et logique métier.
- Peut s'utiliser avec Composite pour parcourir des arbres.

6. Limites

- Difficile si les classes d'éléments changent souvent (il faut modifier tous les visiteurs).
- Peut complexifier le code (beaucoup de méthodes visit).

7. Relations avec d'autres patterns

- **Décorateur** : ajoute dynamiquement des responsabilités à un objet → différence : Visitor ajoute de nouvelles opérations globales sur une hiérarchie.
- **Composite** : souvent combiné avec Visitor pour parcourir une structure d'arbre.