Fiche résumé : Command Pattern

Catégorie : Pattern comportemental

Définition du pattern Command

Le pattern Command est un pattern comportemental qui transforme une requête en objet. Cela permet de paramétrer des actions à exécuter, de les enchaîner, et de les annuler facilement si besoin.

Problème que veut résoudre le Pattern

Dans de nombreuses applications, plusieurs actions doivent être exécutées dynamiquement (en fonction des choix de l'utilisateur).

Exemple : un logiciel de dessin où l'utilisateur peut annuler ou rétablir ses actions.

Problème : si toutes les actions sont directement gérées dans la classe principale, le code devient compliqué à maintenir et à modifier.

Solution apportée par le pattern Command

- Créer une interface Command avec une méthode execute().

- Chaque action devient une classe concrète qui implémente cette interface (ConcreteCommand), rendant chaque action indépendante.

- Un objet Invoker (comme un bouton ou un menu) utilise les commandes sans connaître leur contenu exact.

Avantages	Inconvénients
Permet d'ajouter de nouvelles commandes facilement sans modifier le code existant.	Augmente le nombre de classes
Les commandes et leur exécution sont séparées, ce qui rend le code plus facile à maintenir.	Peut alourdir le code pour des actions simples
Idéal pour des fonctionnalités d'annulation et de répétition d'actions.	

Schéma et rôles des classes

Command : Interface pour exécuter une action execute().

ConcreteCommand : Classe concrète qui encapsule une action spécifique et contient un Receiver (objet qui exécute l'action réelle).

Receiver : Contient la logique métier pour exécuter l'action (par exemple, la logique pour copier, coller, ou dessiner un objet).

Invoker : Demande l'exécution de la commande via l'interface `Command` (sans connaître le contenu de l'action).

Client : Crée les commandes et les associe aux Invokers.