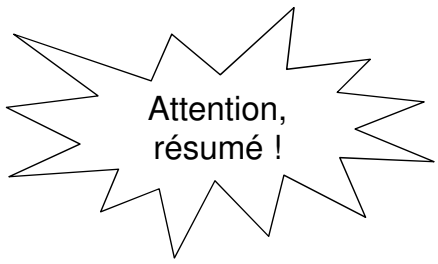
3. Catalogue : quelques design patterns

Le modèle « Stratégie » (1/6)



- Identification du pattern
  1. Stratégie
     - Modèle « comportemental » de niveau « objet »
  2. Alias : Politique

Le modèle « Stratégie » (2/6)

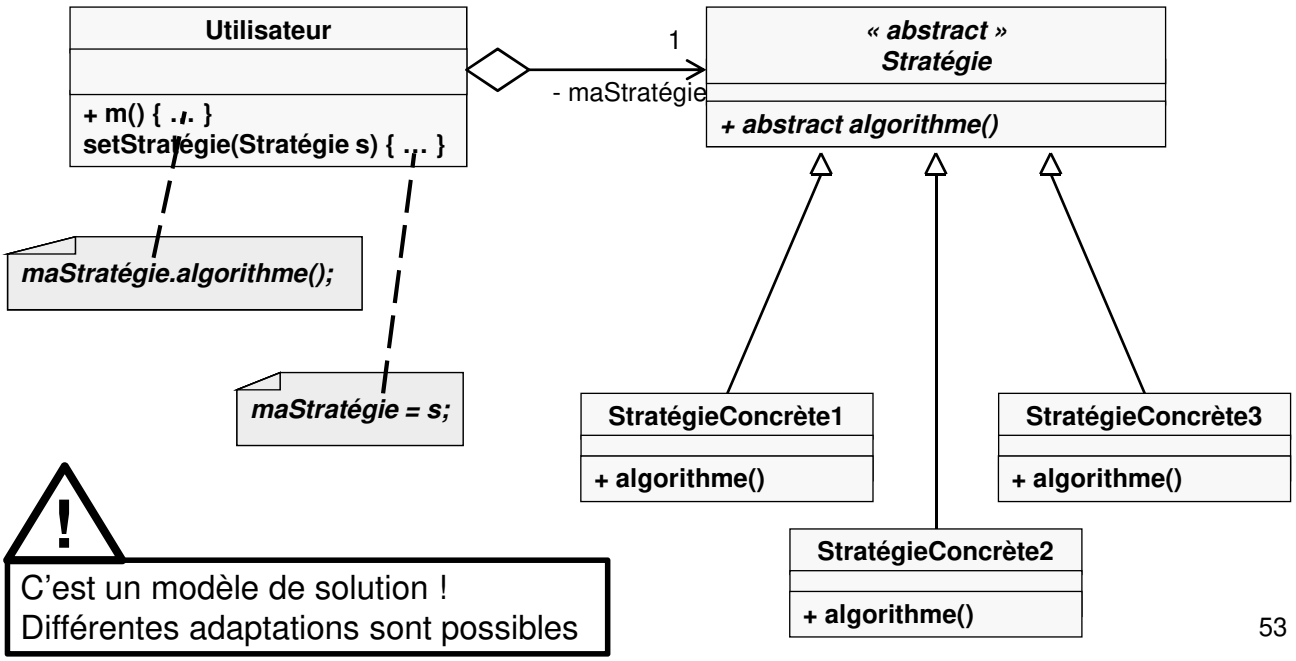
- Problème et contexte
  3. Intention
     - Découpler la classe qui utilise un « algorithme » de celle qui implante cet algorithme (principe de séparation des responsabilités)
     - Permettre de définir des objets qui exécutent algorithmes inconnus à la conception et/ou interchangeables
       - Pouvoir faire évoluer indépendamment les algorithmes et les objets qui les utilisent : ajouter de nouveaux algorithmes, ou modifier ou retirer des algorithmes existants, jusqu'à changer d'algorithme dynamiquement
  4. Motivation
     - Pour éviter de coder « en dur » les algorithmes au sein des classes qui les utilisent
     - Le comportement peut être implanté par différents algorithmes mais le choix au moyen d'une structure conditionnelle peut ne pas être adapté

Le modèle « Stratégie » (3/6)

- Problème et contexte (suite)
  5. Indications d'utilisation
     - Par exemple, dans une fenêtre de texte, pour gérer l'affichage des lignes et la césure des mots, on peut employer différents algorithmes
       - Une famille d'algorithmes pour l'affichage, conforme à une même interface « stratégie »
       - Un algorithme est encapsulé dans un objet de type « stratégie »
       - L'objet de type « stratégie » est un délégué de l'objet utilisateur

Le modèle « Stratégie » (4/6)

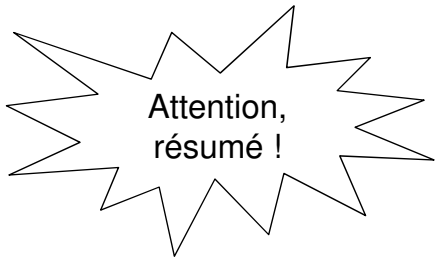
- Solution
  6. Structure



Le modèle « Stratégie » (5/6)

- Solution
  7. Participants
     - *Stratégie* : classe abstraite ou interface qui déclare une interface commune aux algorithmes (super-type)
     - *StratégieConcrète* : implémente l'algorithme conformément à l'interface *Stratégie*
     - *Utilisateur* : classe utilisatrice (cliente) de l'algorithme qui gère une référence sur un objet de type Stratégie
  8. Collaborations
     - Un objet *utilisateur* transmet les requêtes à l'objet *stratégie* (sous-traitance ou « délégation »)
     - L'objet *stratégie* peut éventuellement accéder à des données propres à l'objet *utilisateur* à travers une méthode dédiée (« callback »)

Le modèle « Stratégie » (6/6)



- Conséquences et réalisation
  9. Conséquences…
  10. Implémentation…
  11. Exemples de code...
- Compléments
  12. Utilisations remarquables…
  13. Modèles apparentés
      - Patron de méthode...
      - Etc.