3. Catalogue : quelques modèles de conception

Le modèle « Stratégie » (1/4)

- Stratégie
  - Pattern comportemental de niveau objet

- Intention
  - Découpler l'algorithme utilisé de la classe qui l'utilise
  - Permettre de définir des objets qui exécutent algorithmes inconnus à la conception et/ou interchangeables
    - Les algorithmes peuvent évoluer indépendamment des objets qui les utilisent

Le modèle « Stratégie » (2/4)

- Motivation
  - Pour éviter de coder « en dur » les algorithmes au sein des classes qui les utilisent
    - Pour séparer les codes des algorithmes et les codes des classes utilisatrices
    - Pour pouvoir ajouter de nouveaux algorithmes, ou modifier ou retirer des algorithmes existants
    - Pour pouvoir changer d'algorithme dynamiquement
  - 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 » (3/4)

- 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
- Collaborations
  - un *utilisateur* transmet les requêtes à l'objet *stratégie* (sous-traitance)
  - l'objet *stratégie* peut éventuellement accéder à des données propres à l'*utilisateur* à travers une méthode dédiée

Le modèle « Stratégie » (4/4)

- Structure