IFT6803:
Génie logiciel du commerce électronique

Chapitre 3: Conception orientée objet

Section 4: Techniques de réutilisation

Julie Vachon,
Hiver 2003

Sommaire

Chapitre 3, Section 4

« Techniques de réutilisation »

Techniques de réutilisation

Pour favoriser la robustesse, la fiabilité et la maintenance

- Réutilisation de designs flexibles (classes génériques)
- Réutilisation de design robustes et fiables (design patterns)

3.4.1 Design patterns

Qu'est-ce qu'un design pattern?
(schéma de conception)

- « Une *solution* à un *problème* dans un *contexte*. »
  - Contexte= Ensemble de situations récurrentes auxquelles le pattern s'applique.
  - Problème = Ensemble de buts et de contraintes qui se présentent dans le contexte.
  - Solution = Schéma de conception canonique ou règle de conception qui peut être utilisé pour résoudre le problème.
- Micro-architecture réutilisable qui décrit les **structures statiques** et **dynamiques** et les collaborations entre les éléments du design.

Design patterns

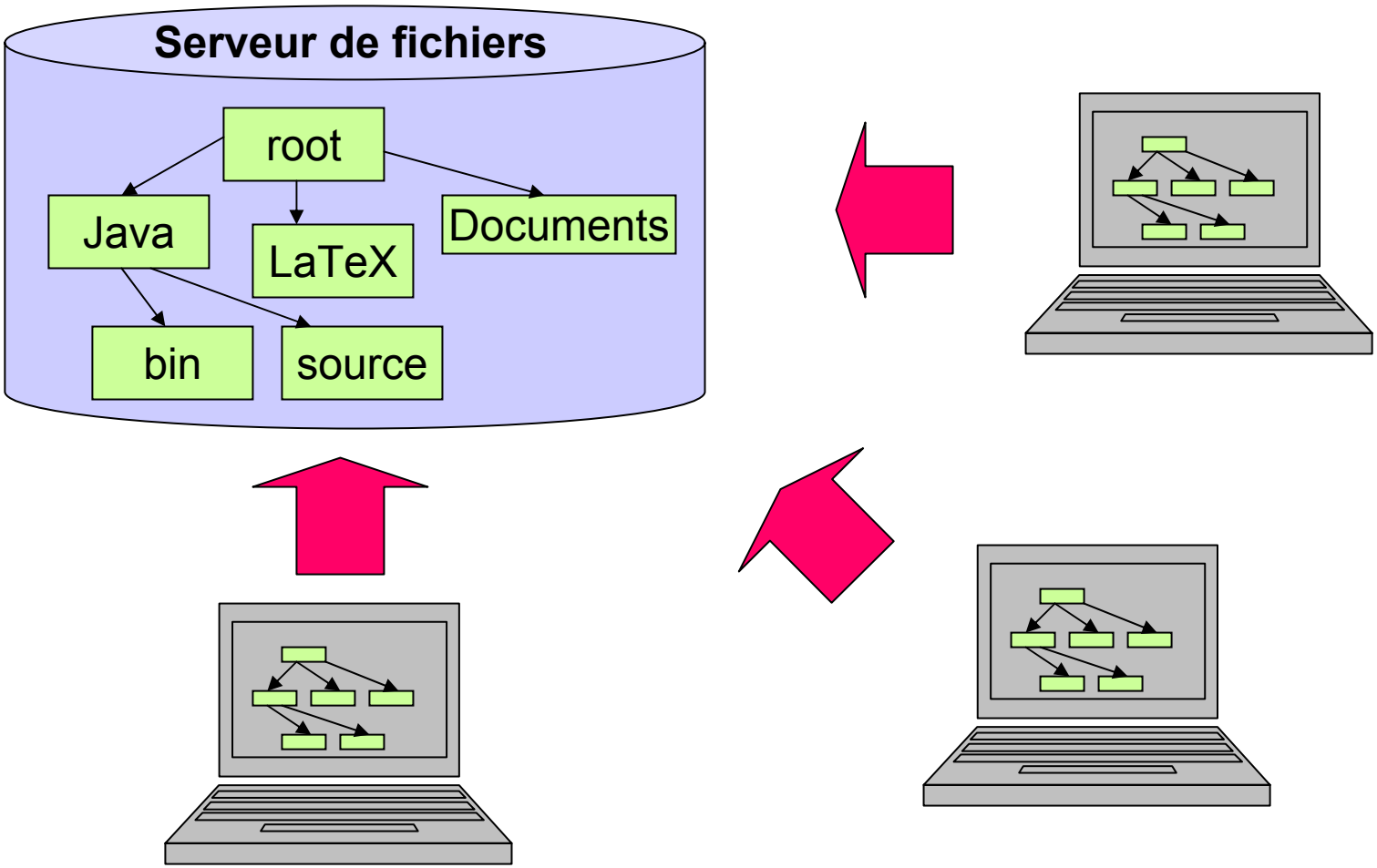
Qu'est-ce qu'un design pattern?

- Utilité:
  - Permet de partager et de réutiliser l'expertise en design et le savoir-faire entre développeurs.
  - Outil de documentation.
  - Moyen de communication conceptuelle.
  - Outil pédagogique. Diffusion des connaissances en design.

- Gamma et al ont classifié les design patterns en trois catégories
  - Patterns de création: SIns permettant de créer des objets de façon flexible.
  - Patterns structuraux: SIns permettant d'organiser l'agencement structurel d'un ensemble d'objets de façon à en faciliter l'accès et la maintenance.
  - Patterns comportementaux: SIns permettant d'organiser les interactions d'un ensemble d'objet de façon efficace et facilement maintenable.
- Catalogue de patterns sur Internet: http://www.dofactory.com/Patterns/Patterns.aspx#list

Design patterns

Exemple – Serveur de fichiers distant



Design patterns

Exemple – Serveur de fichiers distant

Les contraintes

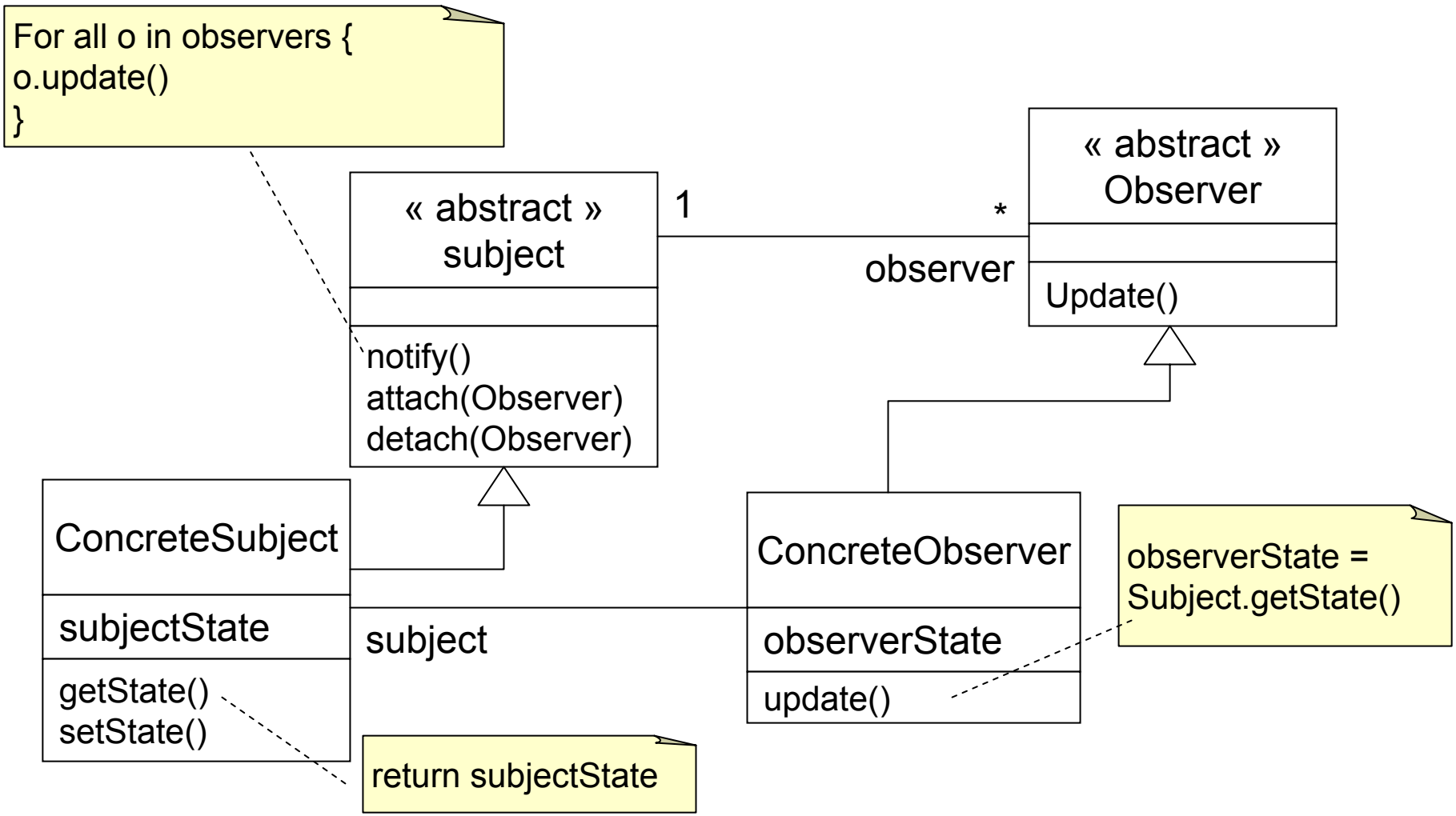
- L'identité et le nombre de receveurs ne sont pas connus à l'avance.
- De nouvelles classes de receveurs pourraient s'ajouter au système.
- Le sondage (polling) n'est pas approprié i.e. impossible ou trop coûteux si grand nombre de receveurs.

Solution ?

Pattern Observer

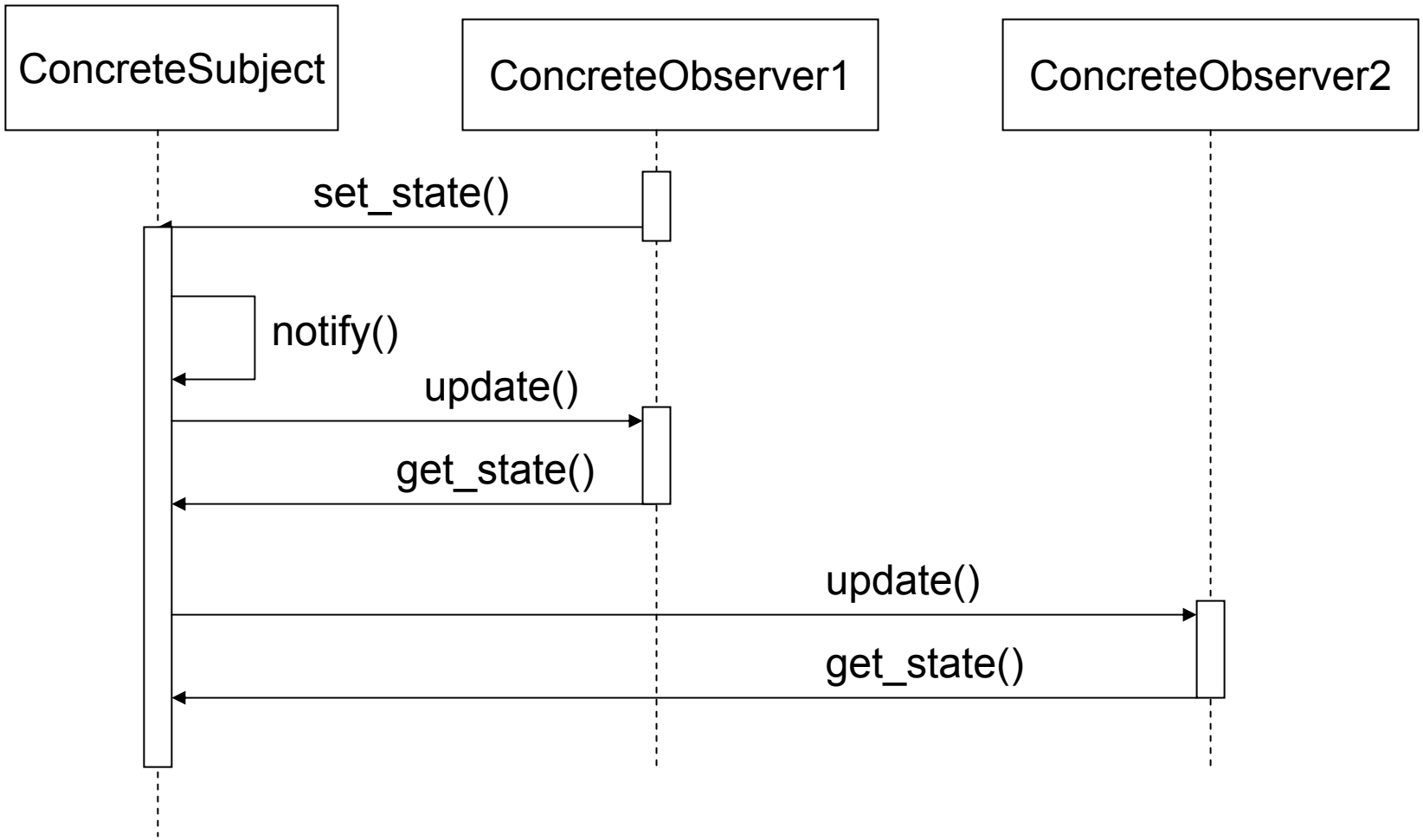
Design patterns

Pattern Observer – Structure statique



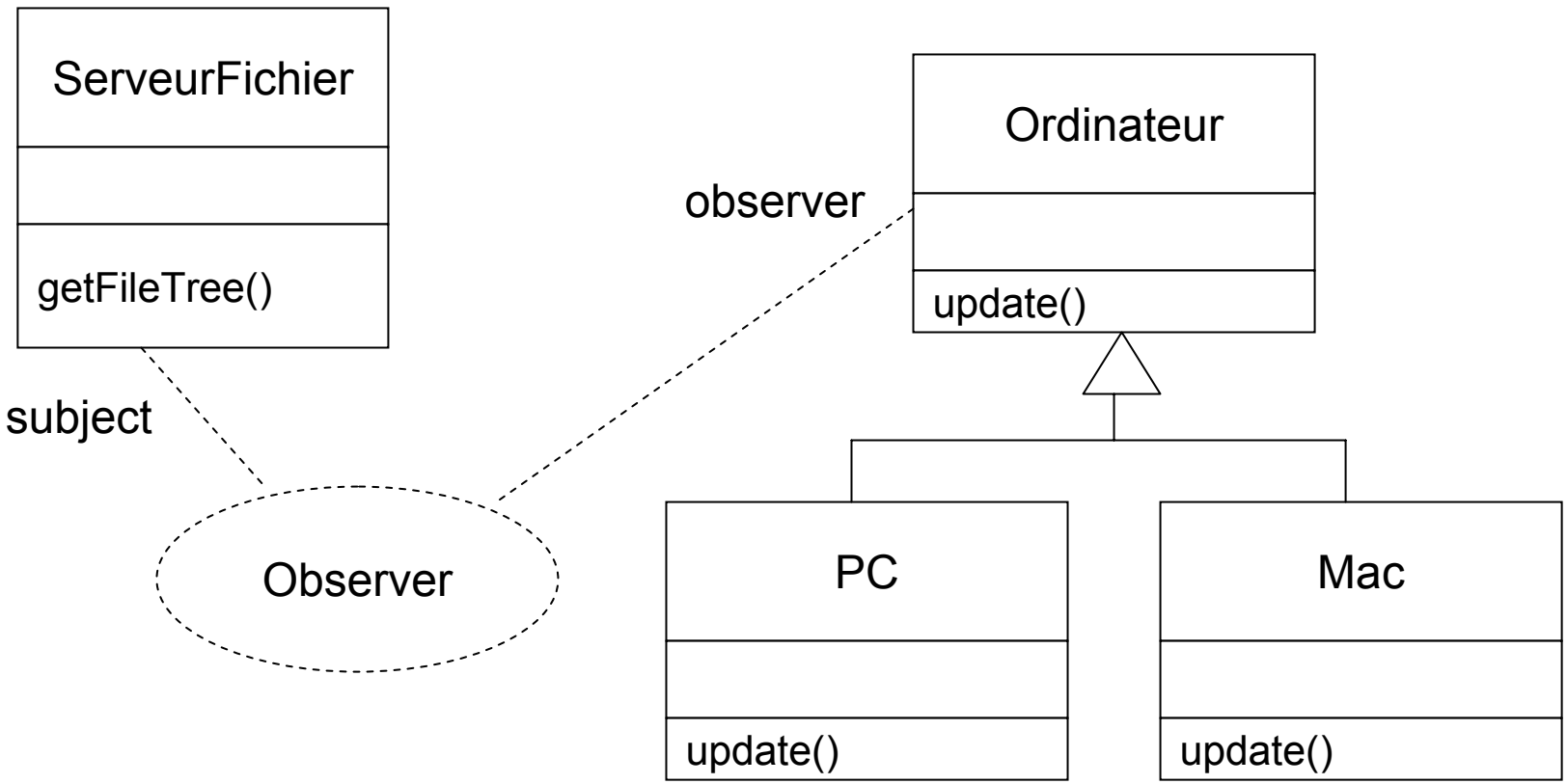
Design patterns

Pattern Observer – Structure dynamique



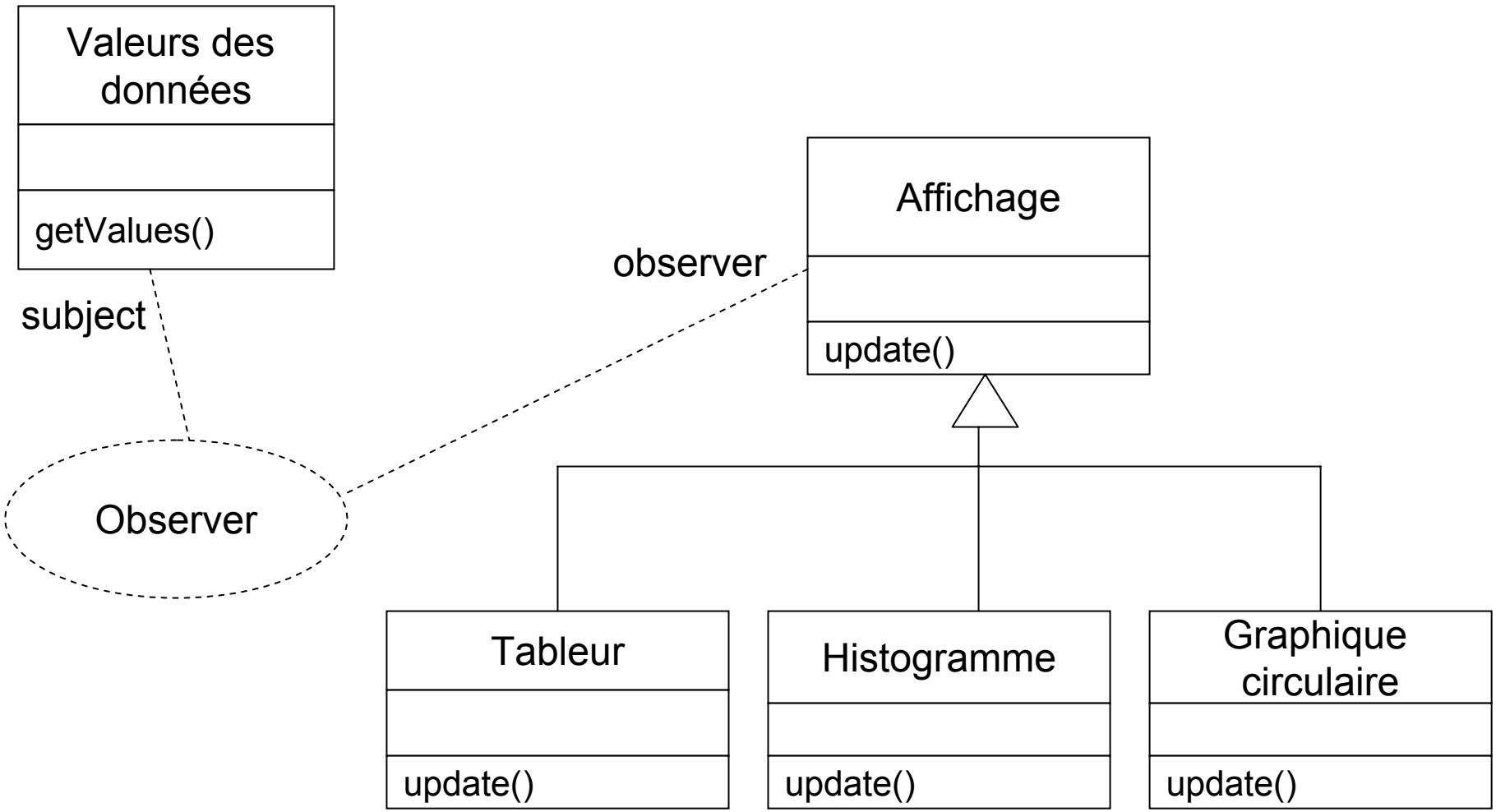
Design patterns

Pattern = collaboration



Design patterns

Pattern = collaboration



Design patterns

Description d'un design pattern

Si

- vous vous trouvez dans ce [contexte]
- par exemple [exemples]
- aux prises avec ce [problème]
- impliquant ces [buts et contraintes]

Alors

- pour ces [raisons]
- appliquer le [schéma de conception] et/ou [règle] suivants
- pour construire cette [solution]
- conduisant à ce [nouveau contexte] et [autres patterns]

Design patterns

Template de description d'un design pattern

- Nom du pattern
- Objectif
- Alias
- Motivation
- Applicabilité
- Structure
- Participants
- Collaborations
- Conséquences
- Implémentation
- Exemple de code
- Utilisations connues
- Patterns connexes

Design patterns de création

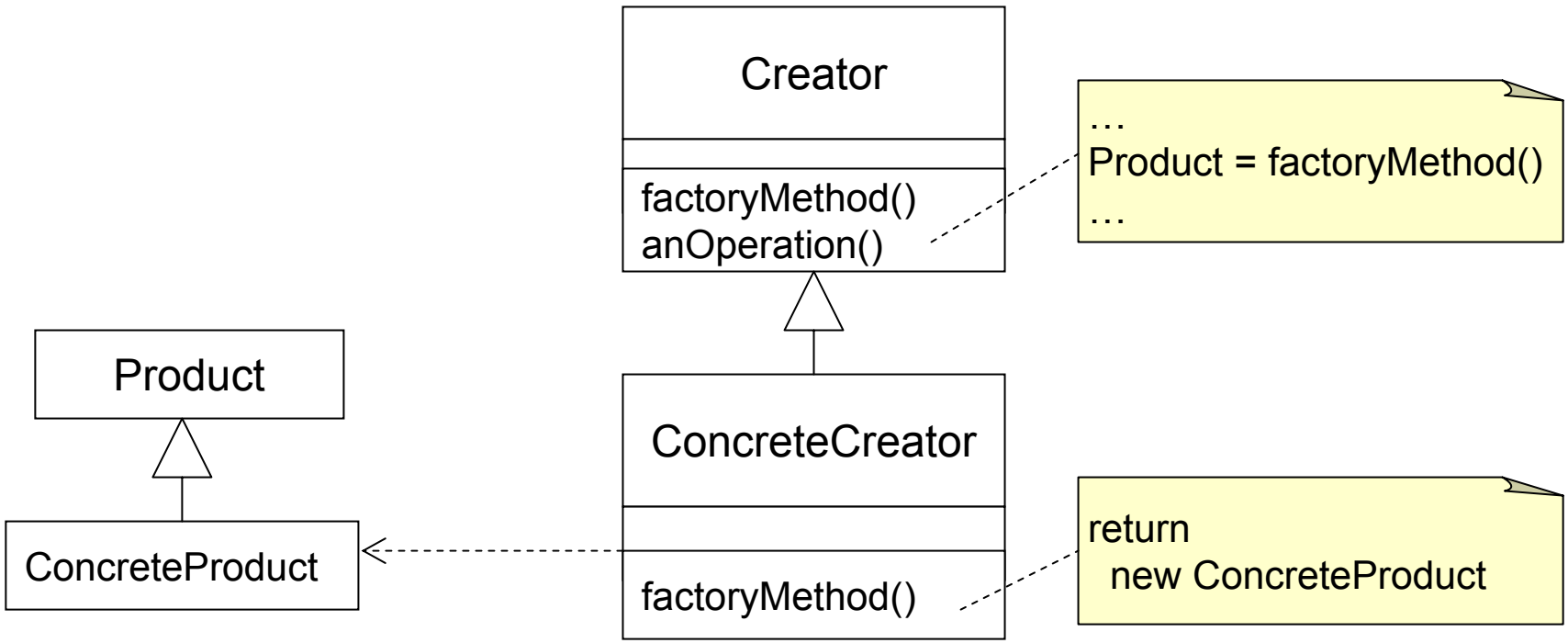
Patterns de création

- Utilité générale:

  Permet de découpler la connaissance du type d'un objet (*quoi*) du processus de création de cet objet (*quand et comment*).
- Pattern GoF:
  - Abstract factory
  - Builder
  - **Factory Method**
  - Prototype
  - Singleton

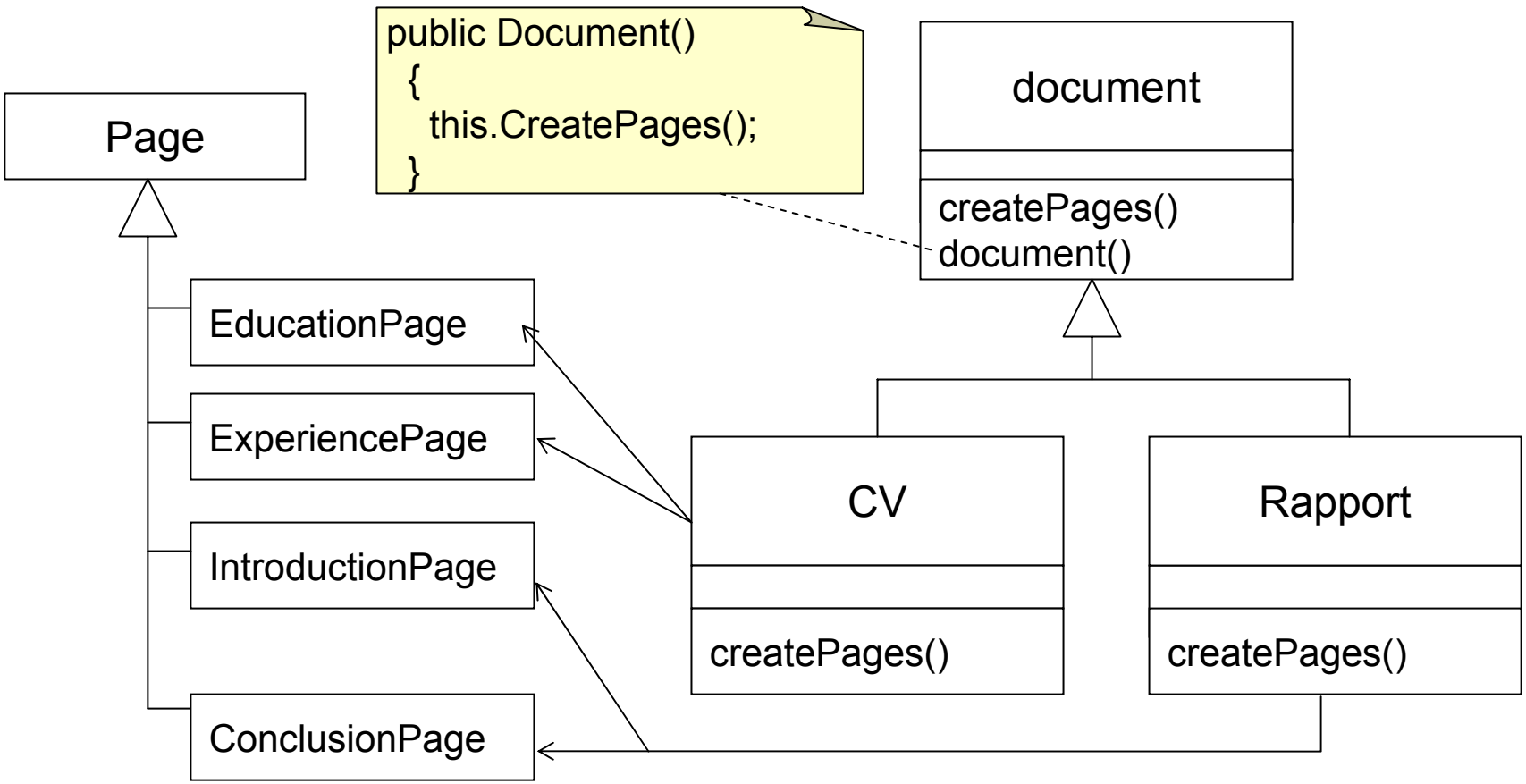
Design patterns de création

Pattern Factory Method



Design patterns de création

Pattern Factory Method



Design patterns structurels

Patterns structurels

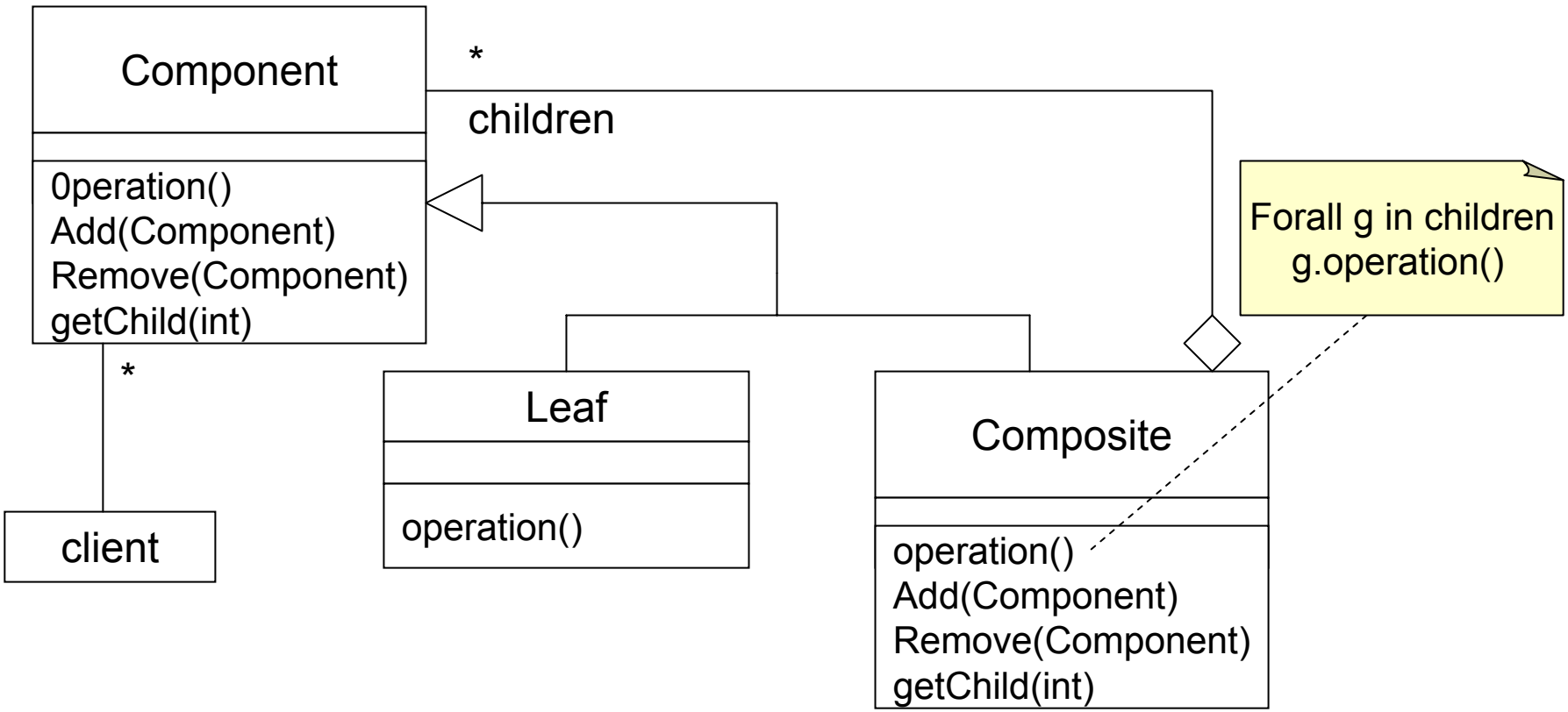
- Utilité générale:

  Spécifie comment les classes et les objets sont associés pour former de plus grandes structures.
- Patterns GoF:
  - Adapter
  - Bridge
  - **Composite**
  - Decorator
  - Facade
  - Flyweight
  - Proxy

Design patterns de création

Pattern Composite

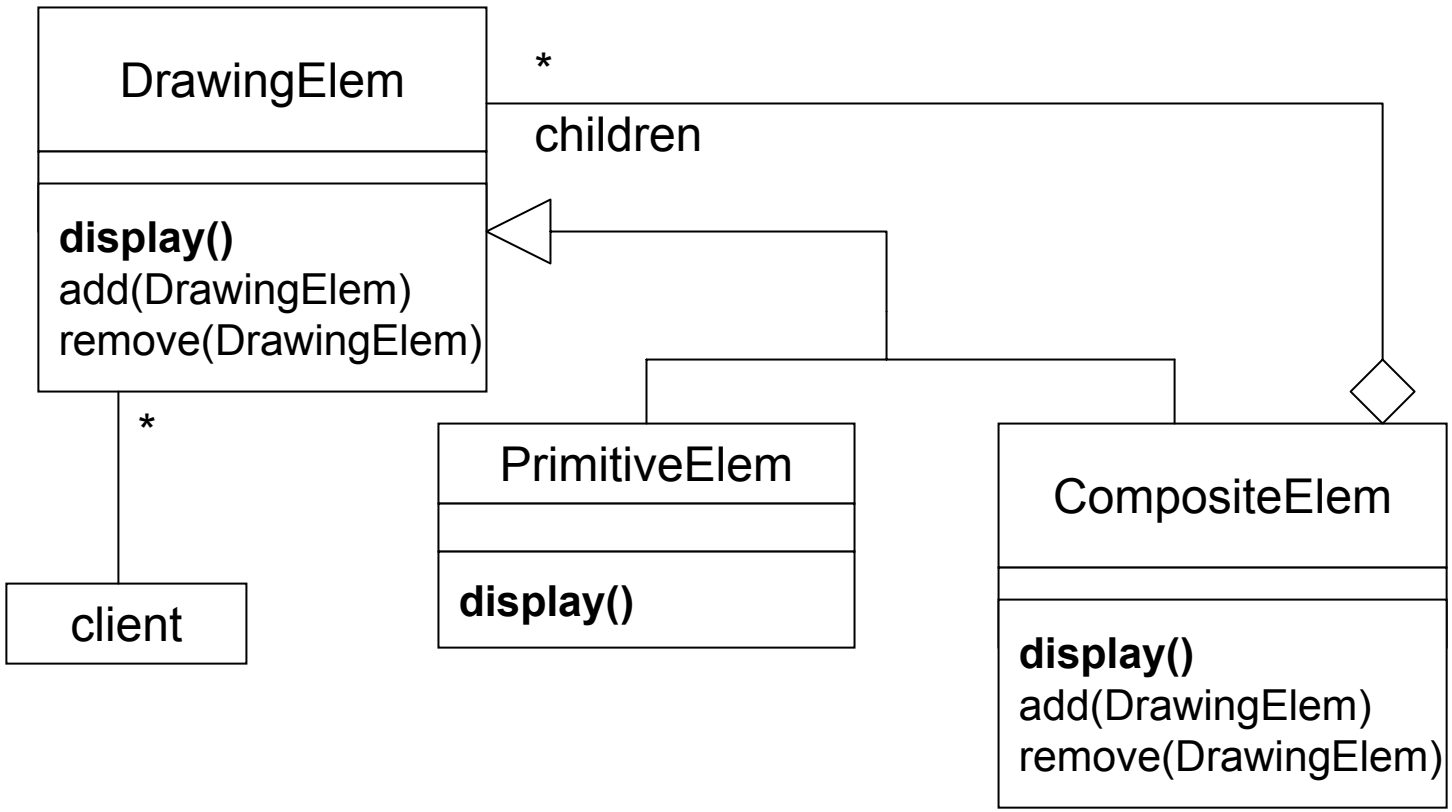
- Objectif: Agencer les objets dans une structure en arbre de façon à définir un hiérarchie partie/tout.



Design patterns de création

Pattern Composite

- Objectif: Agencer les objets dans une structure en arbre de façon à définir un hiérarchie partie/tout.



Design patterns comportementaux

Patterns comportementaux

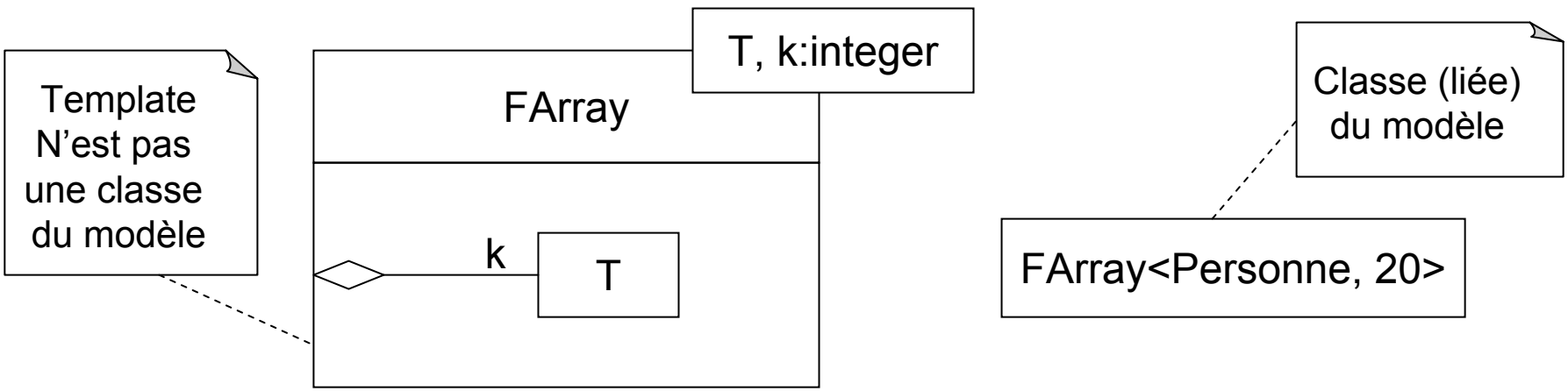
- Utilité générale:

  S'intéresse aux algorithmes et à l'attribution de responsabilités aux objets.
- Patterns GoF:
  - Chain of responsibility
  - **Command**
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - **Observer**
  - State, Strategy, Template method, visitor

3.4.1 Classe générique

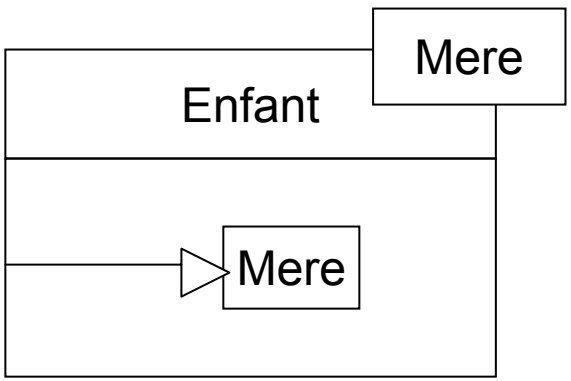
Template

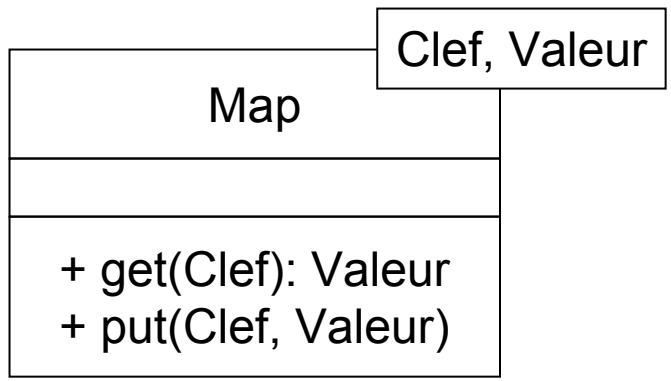
- Description **paramétrée** d'un groupe d'éléments ***potentiels***.
  - Une classe générique (paramétrée) décrit un ensemble de classes potentielles.
- Pour qu'un élément existe ***effectivement*** dans le modèle, il faut ***lier*** les paramètres du *template* à des valeurs ***effectives***.
  - Un *template* n'est pas une classe. Il doit être ***instancié*** pour représenter une classe effective dans un modèle.
  - Les valeurs effectives des paramètres sont généralement des types classes ou types primitifs (integer, string).



Classe générique

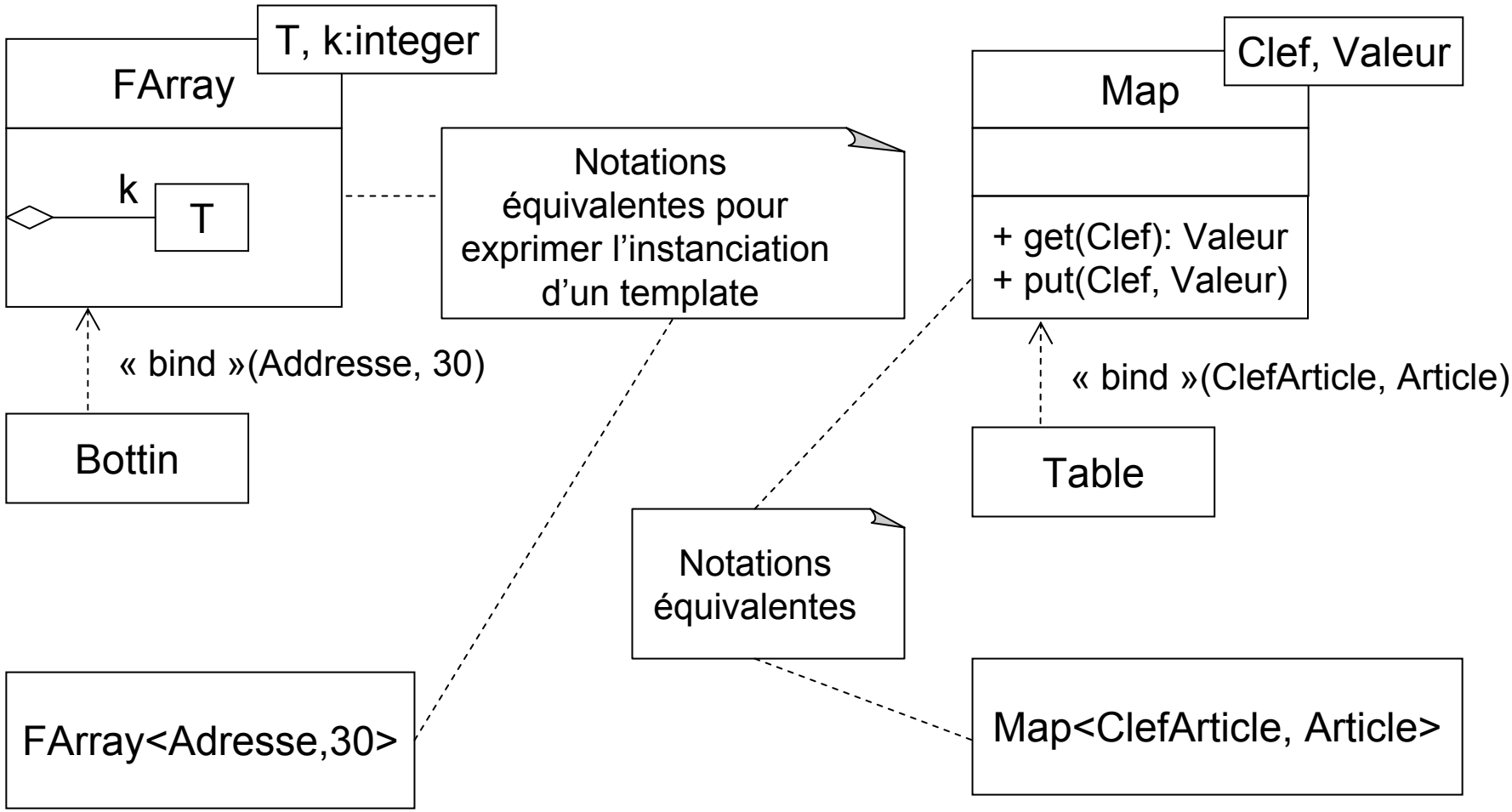
- Paramètres
  - Les valeurs effectives des paramètres sont généralement des types classes ou types primitifs (integer, string).
  - Peuvent apparaître dans
    - le type des attributs de la classe
    - la signature des opérations de la classe
    - La définition d'une relation de la classe (généralisation, association, agrégation ou composition)





Classe générique

Exemple - Instanciation



Classe générique

Exemple