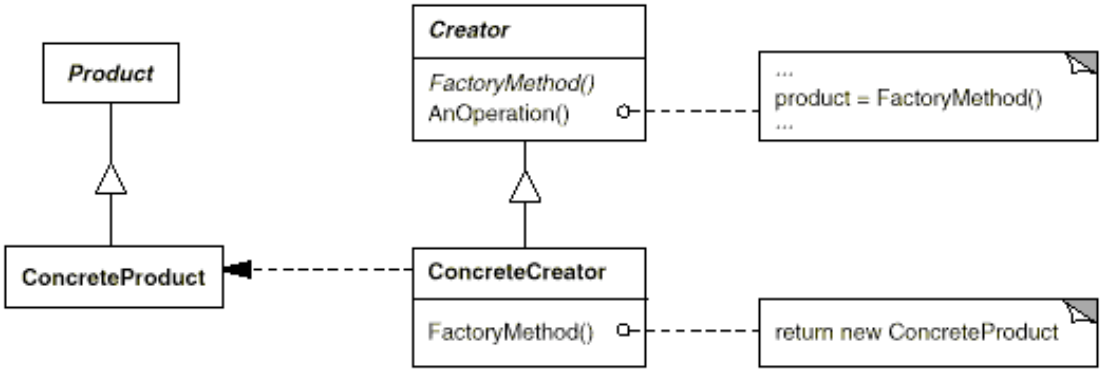
Fabrique (Factory Method)



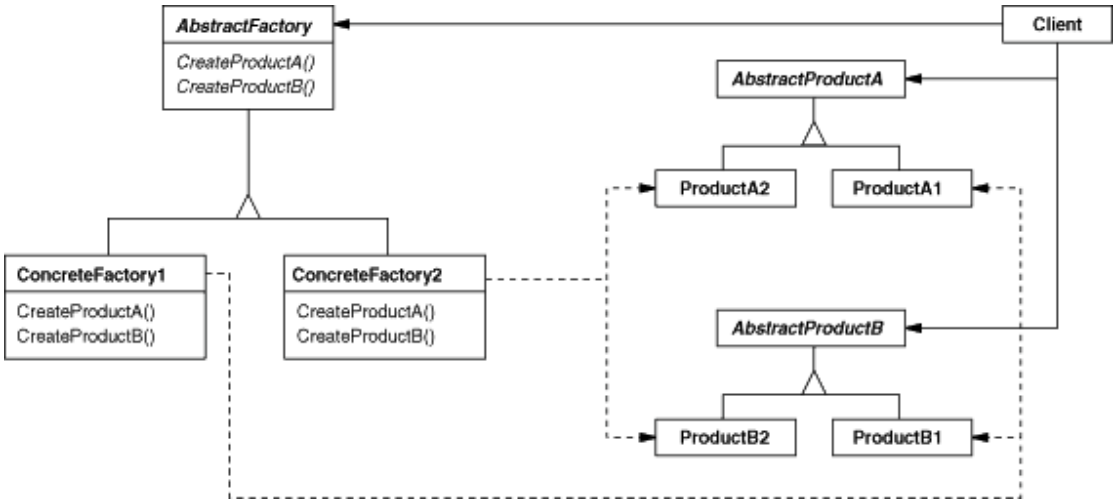
Product: defines the interface of objects the factory method creates.

ConcreteProduct: implements the Product interface.

Creator: declares the factory method, which returns an object of type Product. Creator may also define a default implementation of the factory method that returns a default ConcreteProduct object. It may call the factory method to create a Product object.

ConcreteCreator: overrides the factory method to return an instance of a ConcreteProduct.

Fabrique abstraite (Factory Method)



AbstractFactory: declares an interface for operations that create abstract product objects.

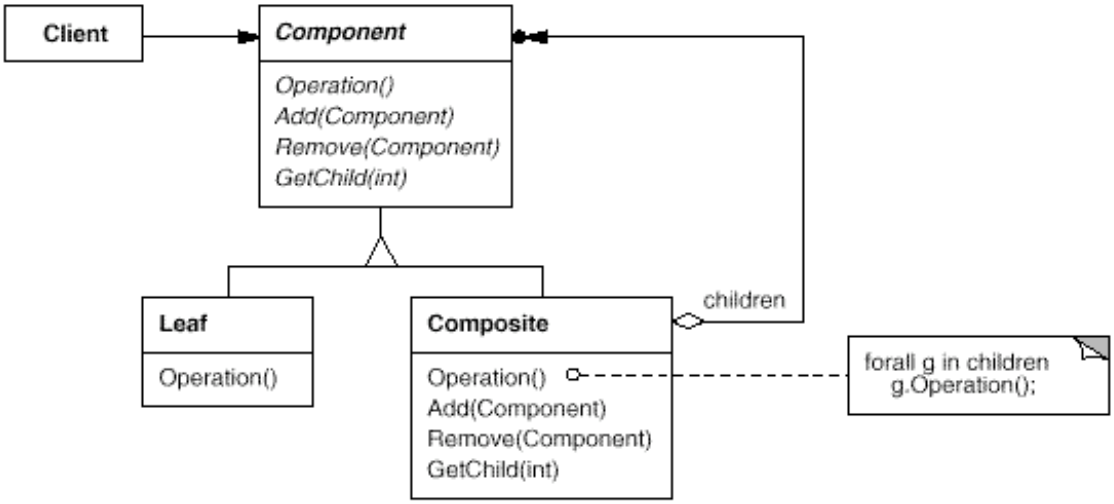
ConcreteFactory: implements the operations to create concrete product objects.

AbstractProduct: declares an interface for a type of product object.

ConcreteProduct: defines a product object to be created by the corresponding concrete factory. It implements the AbstractProduct interface.

Client: uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Objet composite (Composite)



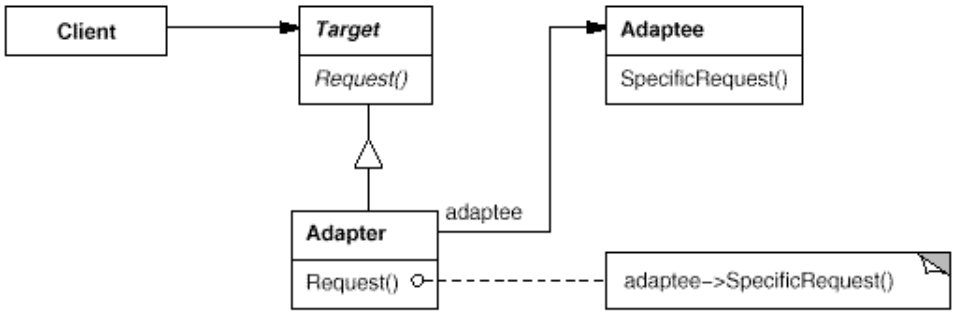
Component: declares the interface for objects in the composition,.It implements default behavior for the interface common to all classes, as appropriate. It declares an interface for accessing and managing its child components. It defines an interface for accessing a component's parent in the recursive structure, and implements it if that's appropriate (optional).

Leaf: represents leaf objects in the composition. A leaf has no children. It defines behavior for primitive objects in the composition.

Composite: it defines behavior for components having children. It stores child components. It implements child-related operations in the Component interface.

Client: manipulates objects in the composition through the Component interface.

Adaptateur (Adapter)



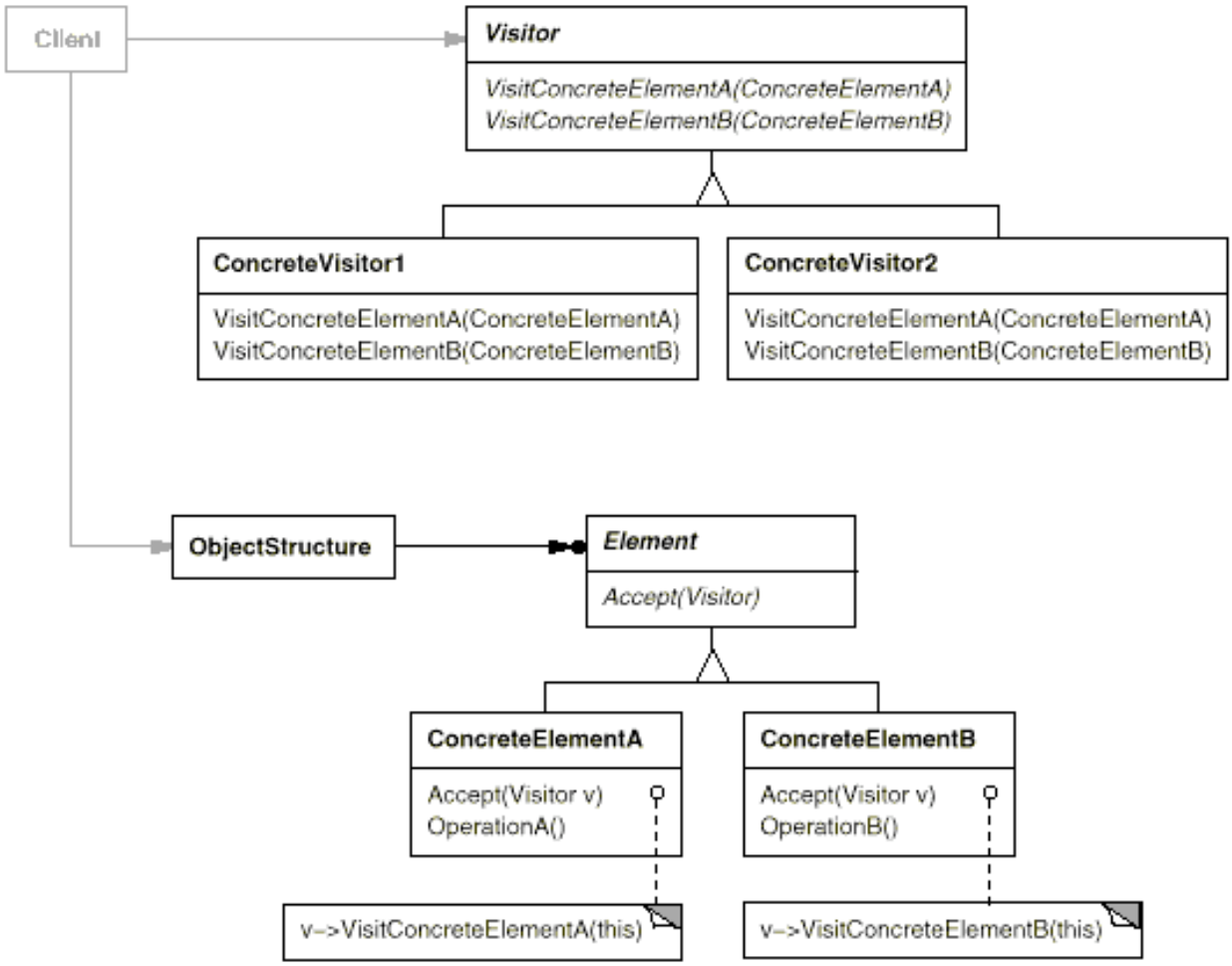
Target: defines the domain-specific interface that Client uses.

Client: collaborates with objects conforming to the Target interface.

Adaptee: defines an existing interface that needs adapting.

Adapter: adapts the interface of Adaptee to the Target interface.

Visiteur (Visitor)



Visitor: declares a Visit operation for each class of ConcreteElement in the object structure. The operation's name and signature identifies the class that sends the Visit request to the visitor. That lets the visitor determine the concrete class of the element being visited. Then the visitor can access the element directly through its particular interface.

ConcreteVisitor: implements each operation declared by Visitor. Each operation implements a fragment of the algorithm defined for the corresponding class of object in the structure. ConcreteVisitor provides the context for the algorithm and stores its local state. This state often accumulates results during the traversal of the structure.

Element: defines an Accept operation that takes a visitor as an argument.

ConcreteElement: implements an Accept operation that takes a visitor as an argument.

ObjectStructure: can enumerate its elements. It may provide a high-level interface to allow the visitor to visit its elements. It may either be a composite or a collection such as a list or a set.

ATM

Exercice 1. On veut modéliser chaque communication du distributeur avec une banque par un objet. Dans une communication on utilise des messages spécifiques pour la banque. Cet objet devrait être capable à créer de nouveaux messages. Donner la diagramme de classes qui décrit votre choix de conception pour la modélisation de cet aspect.

EDT

Exercice 2. On veut modéliser chaque réservation de salle par un objet. Dans l'université il y a quatre types de cours : informatiques, mathématiques, chimie, et cinéma. Les salles associées à chaque cours ont des particularités différentes : les salles d'informatiques contient des ordinateurs, les salles de chimie contient différents types de substances, et les salles de cinéma contient des vidéoprojecteurs. Evidements, les professeurs associés a chaque cours sont aussi différents. Comment on pourrait implémenter la création des réservations. Donner la diagramme de classes correspondant.

LYBSYS

Exercice 3. La bibliothèque contient des livres et des journaux mais on voudrait accéder aussi à des chapitres de livre ou à des articles de journal. Comment on devrait représenter les éléments de la bibliothèque par des objets JAVA ? Donner la diagramme de classe correspondant à votre choix de conception.

Exercice 4. Comment on peut implémenter une fonctionnalité qui compte le nombre de pages de toutes les journaux et les livres contenus dans la bibliothèque ? Donner la diagramme de classe correspondant à votre choix de conception.