



DESIGN PATTERN

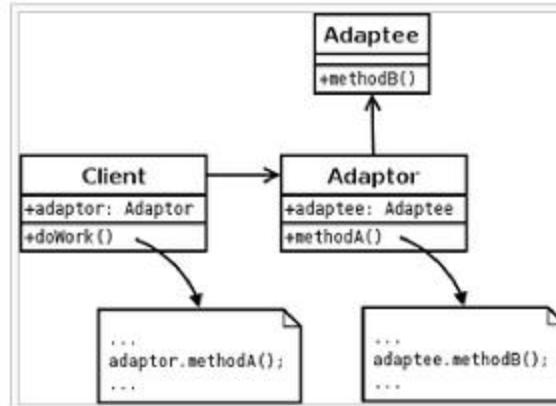
DESIGN PATTERNS

[GoF] Structuraux

- ✿ Comment sont assemblés les objets.
- ✿ Découpler l'interface de l'implémentation.

[GoF] Adapter / Adaptateur

- ◆ Ajuster l'interface d'un objet à celle attendue par le code client.

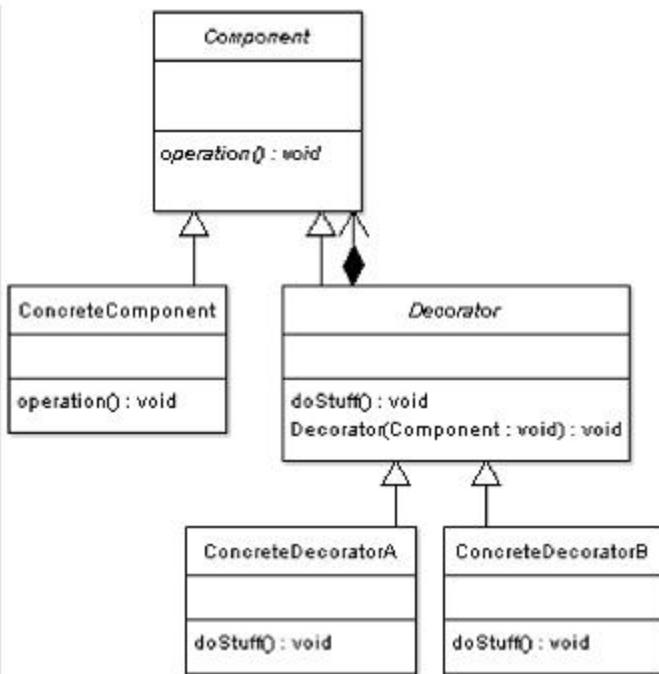


- ◆ Moins d'impacts en PHP que dans des langages à fort typage.

[GoF] Adaptateur, exemple

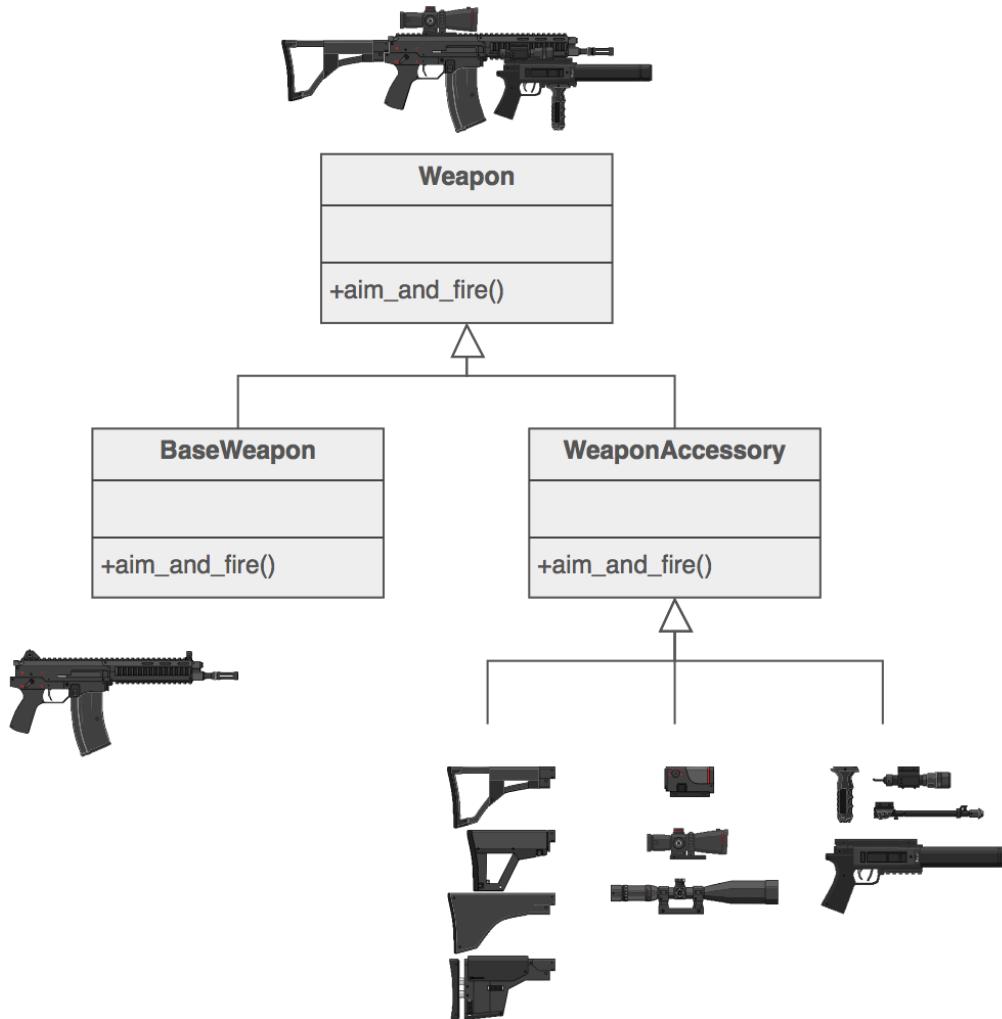
```
class AuthAutreBibliothèque {  
    public function getLevel (){//calcul le niveau de l'utilisateur}  
}  
  
class AuthAdapter implements Auth {  
    private $_adapted = null;  
    function __construct ($login){  
        $this->_adapted = $login;  
    }  
    function isConnected (){  
        return $this->_adapted->getLevel () >  
            AuthAutreApplication::ANONYMOUS;  
    }  
}
```

[GoF] Decorator / Décorateur



- ➊ Rajouter des fonctionnalités à des composants existants
- ➋ Héritage impossible
- ➌ Une arborescence existante

[GoF] Decorator



 [GoF] Decorator, exemple

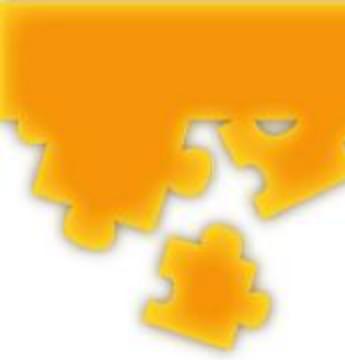
```
class decorated {}

abstract class Decorator {

    private $_decorated;

    function __construct ($decorated){
        $this->_decorated = $decorated;
    }

    function doStuff (){
        return $this->_decorated->doStuff ();
    }
}
```

 [GoF] Decorator exemple (2)

```
class Decorator1 extends Decorator {  
    function doOtherStuff (){  
        //autre traitement  
    }  
}  
  
class Decorator1 extends Decorator {  
    function yaS (){  
        //yet another stuff  
    }  
}
```

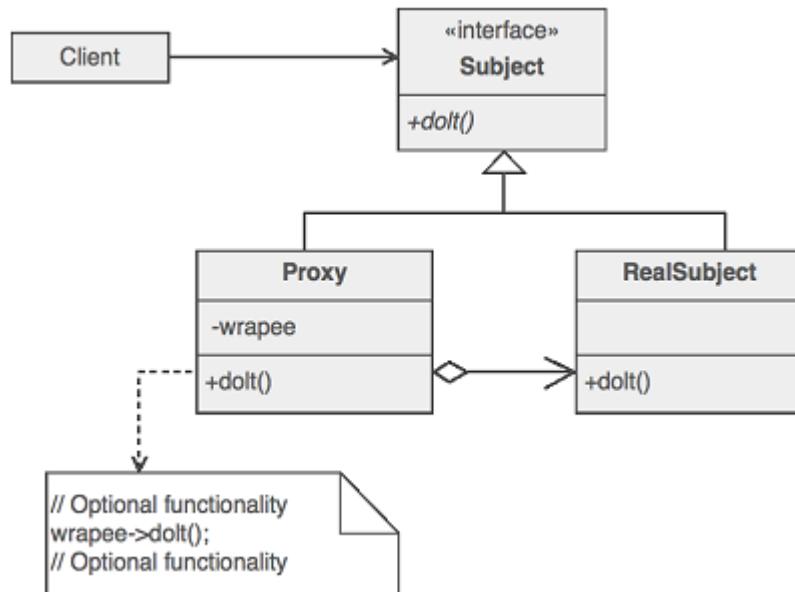
[GoF] Proxy

- ✿ **Objectif : ajout d'un niveau d'indirection pour accéder à un objet**
 - optimisation des ressources
 - simplification des accès (distribution)
 - *protection de l'objet (droits)*
- ✿ **Problème**

allocation « jit » d'objets gourmands en ressources
- ✿ **Solution**
 - encapsulation dans le proxy de l'objet réel*
instantiation à la première utilisation
 - redirection des requêtes du proxy sur l'objet réel*
- ✿ **Exemple**

stub (RPC)

[GoF] Proxy



- ◆ **Subject**
interface permettant de rendre transparent Proxy ou RealSubject
- ◆ **RealSubject**
objet réel
- ◆ **Proxy**
*encapsule l'objet réel (wrapee)
méthode de traitement appelant la méthode de traitement de l'objet réel*
- ◆ **Client**
fait appel au Proxy

 [GoF] Proxy

- ✿ Définir l'interface commune
- ✿ Une factory doit-elle être mise en place
instanciation d'un proxy ou d'un objet réel
- ✿ Ajouter la référence à l'objet réel dans le Proxy
initialisé dans le constructeur ou à la première utilisation
- ✿ Implémenter l'interface dans le Proxy
relayer les appels de méthodes sur l'objet réel

[GoF] Proxy

✿ Proxy vs Adaptor

Proxy : même interface que l'objet géré

Adaptor : interface différente

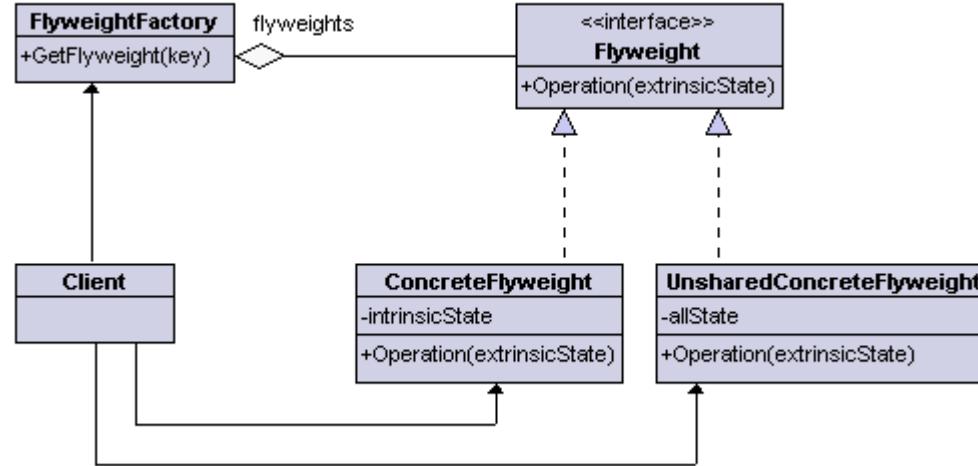
✿ Proxy vs Decorator

- *même structure (indirection, objet encapsulé)*
- *objectif différent*

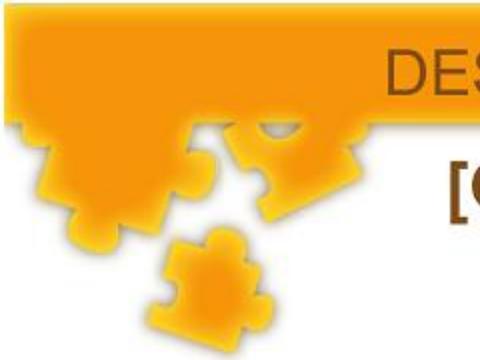
[GoF] Flyweight (Poids mouche)

- ✿ **Objectif : Manipulation de très nombreux petits objets**
granularité
- ✿ **Problème : Coût**
 - performances
 - place mémoire
- ✿ **Solution : Partage d'objets légers (poids mouche)**
parties interne (immuable)
partie externe (stockée et fournie à l'objet léger à l'appel)
- ✿ **Exemple**
réécriture des widgets Motif en gadgets légers

[GoF] Flyweight (Poids mouche)

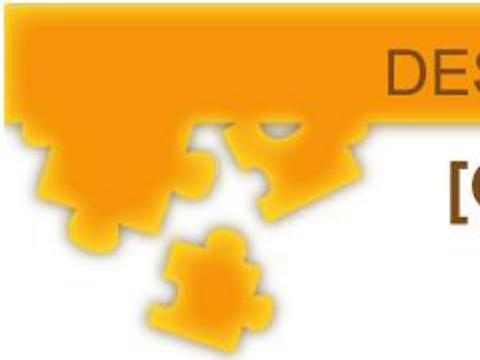


- ✿ **Flyweight**
interface déclarant la méthode de traitement recevant les données externes
- ✿ **UnsharedConcreteFlyweight (implémente Flyweight)**
objets ne nécessitant pas d'être partagés (si besoin)
- ✿ **ConcreteFlyweight (implémente Flyweight)**
objets partagés. Stocke la partie interne
- ✿ **FlyweightFactory**
crée et gère les objets Flyweight (Dictionnaire)
- ✿ **Client**
stocke les objets Flyweight et les données externes



[GoF] Flyweight (Poids mouche)

- ✿ **Mesurer l'importance de la surcharge en objets**
contre-partie : délégation au client de la gestion
- ✿ **Subdiviser la classe en deux**
données internes (partagées) / externes (non partagées)
- ✿ **Supprimer les données non partagées de la classe**
ajout en paramètre à la méthode de calcul
- ✿ **Créer la Factory**
mise en cache des objets déjà créés
- ✿ **Utiliser la Factory dans le Client (pas de new)**
gestion des données externes par le client (calcul, DB,...)

 [GoF] Flyweight (Poids mouche) **Flyweight vs Facade**

Flyweight : manipuler de nombreux petits objets

Facade : un seul objet pour regrouper plusieurs

 **Flyweight + Composite**

Flyweight : gestion des feuilles partagées du Composite

 **Gestion des images chargés par un navigateur**

flyweight : image déjà téléchargée

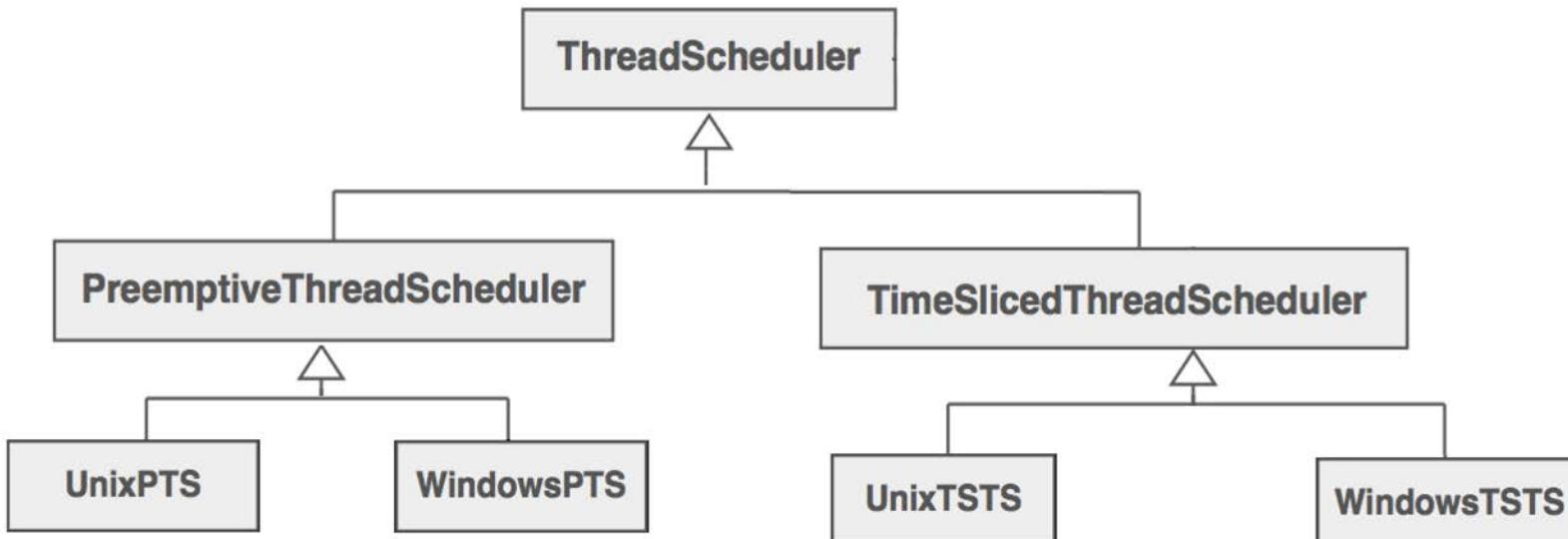
données internes : l'image

données externes : les attributs (position, ...)

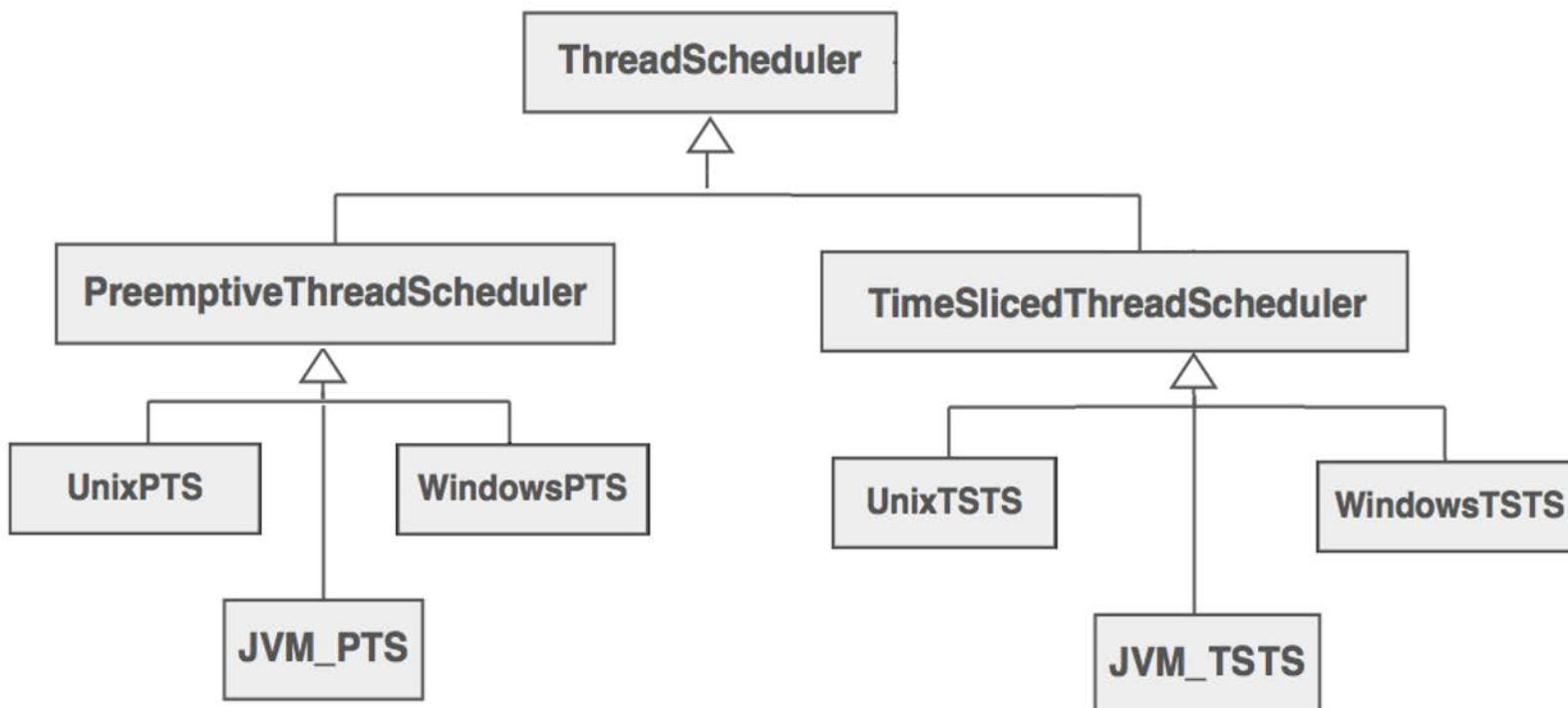
 [GoF] Bridge (pont)

- ✿ **Objectif :** découpler interface et implémentation
permettant leur changement indépendant
- ✿ **Problème :** couplage statique entre interface et implémentation
pas d'extension ou couplage dynamique
→ explosion des couples à prévoir statiquement
- ✿ **Solution :**
double hiérarchie (interface / implémentation)
- ✿ **Exemple**
Interrupteur : *interface (ON(), OFF())*
 implémentation : lampe, téléviseur, borne wifi, ...

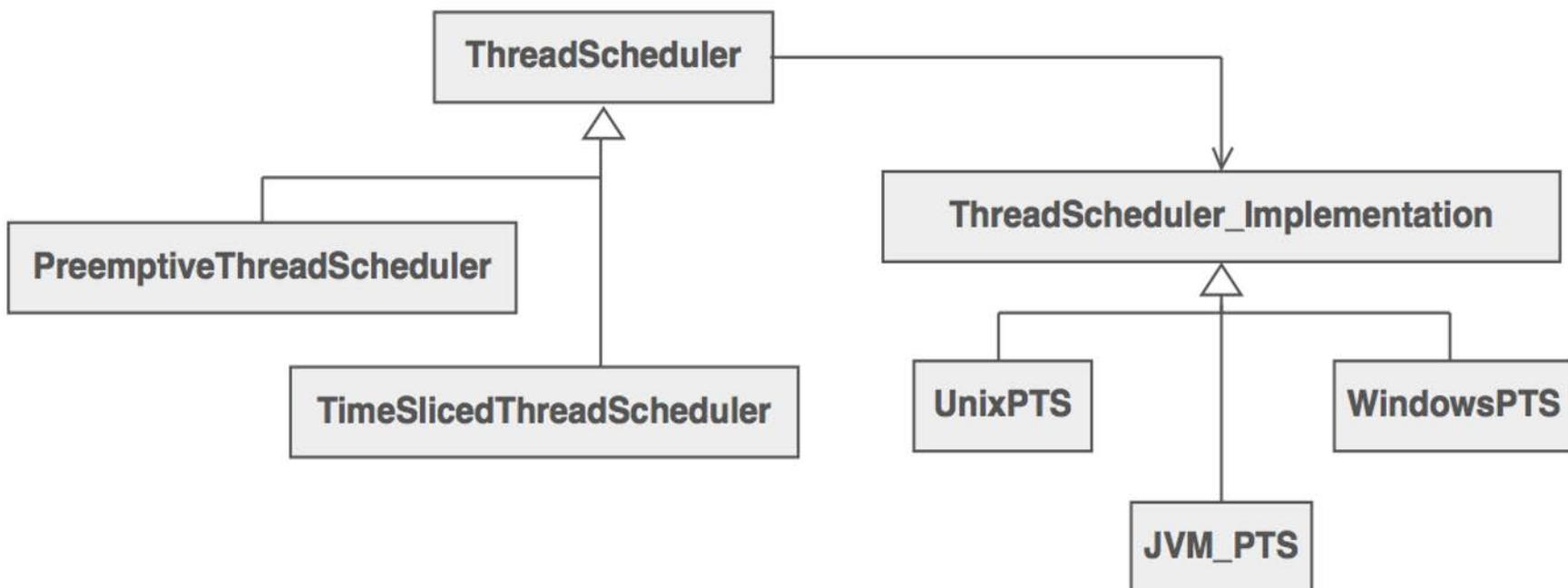
[GoF] Bridge (pont), exemple



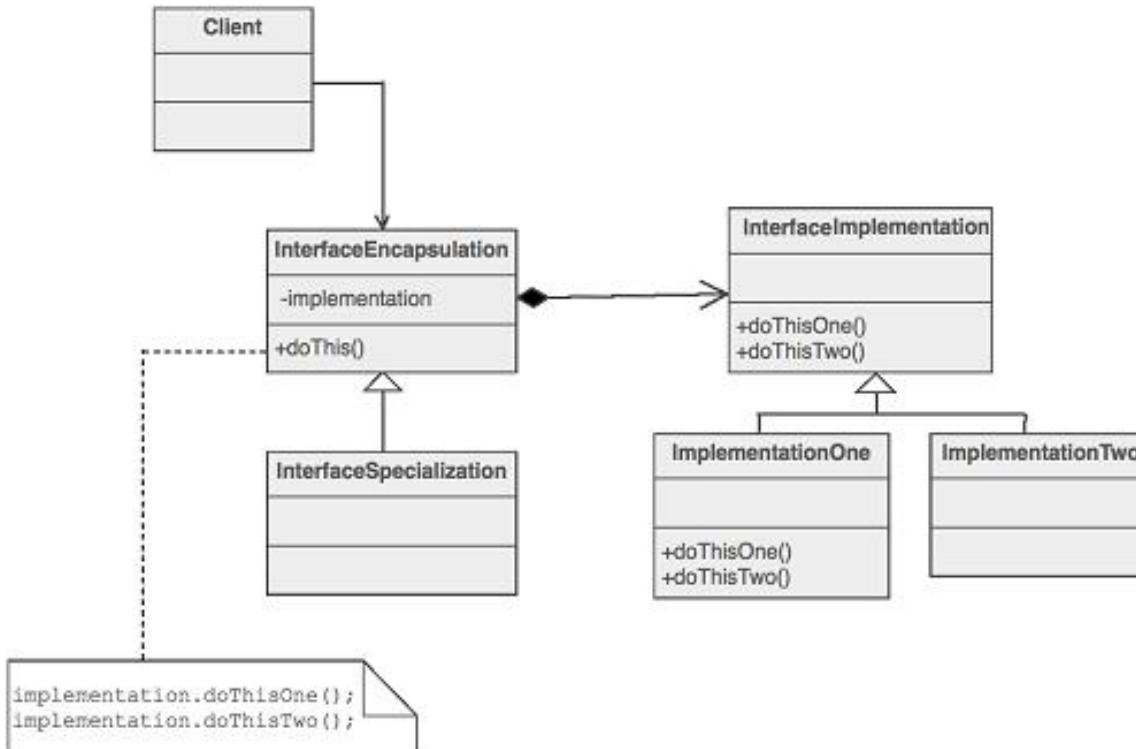
[GoF] Bridge (pont), exemple



[GoF] Bridge (pont), exemple



[GoF] Bridge (pont)



[GoF] Bridge (pont)

- ◆ **Bridge vs Adaptor**

Adaptor : sur des classes qui existent

Bridge : sur des classes à concevoir

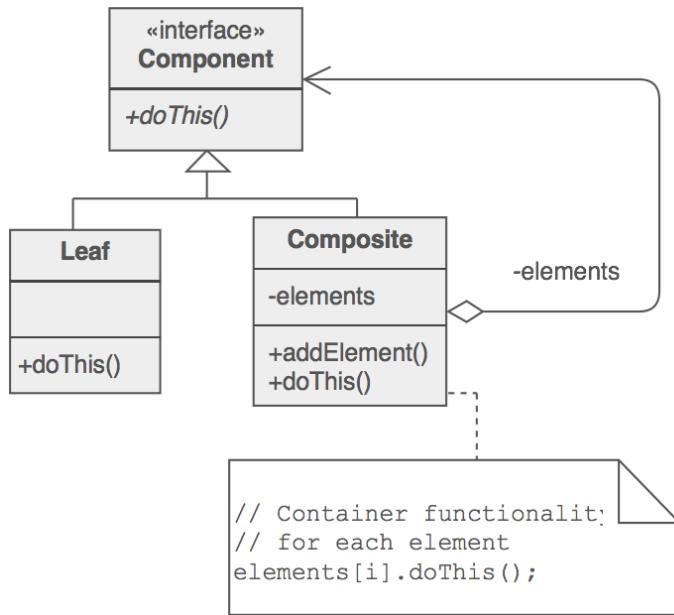
- ◆ **Bridge et Abstract Factory**

*cas où l'interface ne crée pas l'implémentation
(laissée à la charge de la Factory)*

[GoF] Composite

- ✿ **Objectif :** Manipulation de hiérarchie d'objets (arbre)
nœuds (objets composite - récursifs)
feuilles (objets primitifs)
- ✿ **Problème :** uniformité des traitements
sans demander le type des objets (primitifs/composites)
- ✿ **Solution :**
classe de base (abstraite) – méthodes virtuelles
sous-classes (primitive, composite)
méthodes de gestions de l'arbre (addChild, removeChild, getChild)
- ✿ **Exemple**
Expression arithmétique (opérande numérique et opérande composé)

[GoF] Composite



◆ Component

interface déclarant le comportement commun de tous les objets

◆ Leaf

objets primitifs (feuille)

◆ Composite

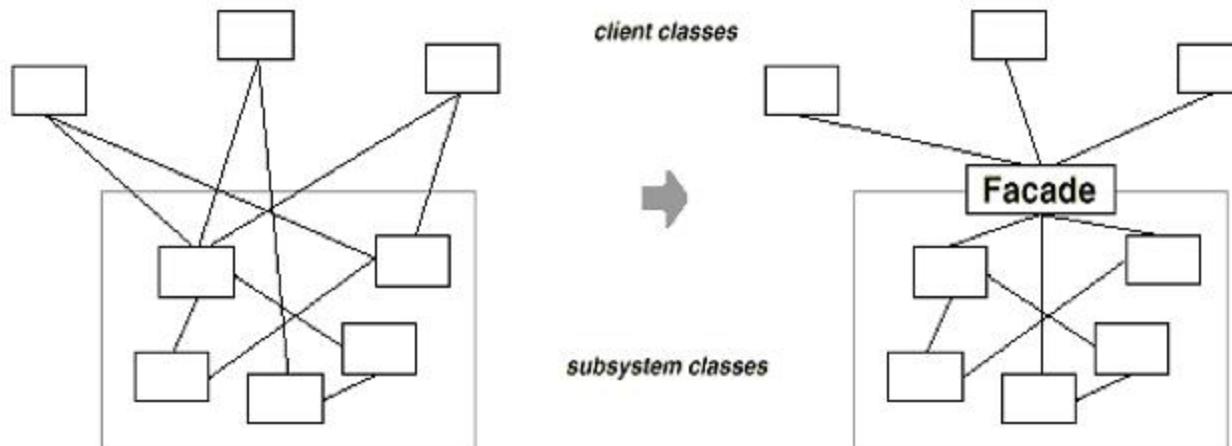
objets composés (agrégation de composants)

 [GoF] Composite

- ✿ Repérer la hiérarchie conteneur/contenu (*pouvant être un conteneur*)
- ✿ Définir l'interface commune minimal de ces classes
- ✿ Définir les classes conteneur/contenu
conteneur et contenu sont des <interfaces> (is a)
conteneur contient plusieurs <interfaces> (has a)
- ✿ Déléguer les traitements aux objets contenus (classe conteneur)
- ✿ Ajouter les méthodes de gestions de l'arbre
en théorie dans l'interface Component

[GoF] Façade

- ✿ Masquer la complexité d'une API.
- ✿ Créer une API simple pour un ensemble complexe.



[GoF] Résumé (2)

✳ Adapter

Adapte l'interface d'une classe à un autre objet.

✳ Decorator

Ajout de fonctionnalités à un objet

✳ Proxy

**Une même interface pour une invocation juste à temps,
invocation à distance.**

✳ Flyweight

Objets réduits en grand nombre

[GoF] Résumé (3)

Bridge

Abstraction découpée de l'implémentation

Composite

Arbres d'objets simples

Façade

API simplifiée pour un système complexe