# La modélisation avec UML: introduction aux patrons de conception ACDA – CPOO (M3105)

### Mathieu Sassolas

IUT de Sénart Fontainebleau Département Informatique

> Année 2015-2016 Cours 4









### Plan de la séance

- Les patrons de conception
- Patrons de création
  - Fabrique abstraite (abstract factory)
  - Singleton
- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy
- Patrons de comportement (semaine prochaine)



### Plan de la séance

Les patrons de conception

Patrons de conception

Patrons de comportement (semaine prochaine)



### Maintenant qu'on « connaît » UML...

Design Patterns

M. Sassolas M3105

Cours 4

Patrons de conception

Créati

Structuratio

Comportement

- ▶ On a vu la syntaxe d'UML.
- ▶ Le meilleur moyen de s'exercer à la conception est de concevoir.



### Maintenant qu'on « connaît » UML...

Design Patterns

M3105

Patrons de conception

Cication

tructuration

Comportement

On a vu la syntaxe d'UML.

 Le meilleur moyen de s'exercer à la conception est de concevoir.

➤ Problème :

Professeur : « Construisez le diagramme de classe du

système qui...

Étudiant : Je ne sais pas par où commencer ».



### Maintenant qu'on « connaît » UML...

Design Patterns

M3105

Patrons de conception

Création

Structuration

Comportemen

- ▶ On a vu la syntaxe d'UML.
  - Le meilleur moyen de s'exercer à la conception est de concevoir.
- ▶ Problème :

Professeur : « Construisez le diagramme de classe du

système qui...

Étudiant : Je ne sais pas par où commencer ».

► Solution : fournir quelques recettes de base.



# Historique des patrons de conception

Design Patterns

M3105 Cours 4

Patrons de conception

Creation

ucturatio

Comportemen

Lorsque les modèles ont commencé à s'unifier dans la syntaxe, on a remarqué que les même problèmes revenaient souvent.

Les même solutions s'appliquent.



# Historique des patrons de conception

Design Patterns M. Sassolas

## Patrons de conception

Création

C -----

- Lorsque les modèles ont commencé à s'unifier dans la syntaxe, on a remarqué que les même problèmes revenaient souvent.
- ► Les même solutions s'appliquent.
- Des motifs (patterns)
   apparaissent dans la conception (design) des programmes : les design patterns.

En français, le terme patron de conception est utilisé, car on s'en sert comme modèle.



# Historique des patrons de conception

### Patrons de conception

- ► Lorsque les modèles ont commencé à s'unifier dans la syntaxe, on a remarqué que les même problèmes revenaient souvent.
- ► Les même solutions s'appliquent.
- ▶ Des motifs (patterns) apparaissent dans la conception (design) des programmes : les design patterns.



 $\hookrightarrow$  Le Gang of Four (GoF) en identifie 23.

En français, le terme patron de conception est utilisé, car on s'en sert comme modèle.



### Les types de patrons de conception

Design Patterns

M. Sassola:

M3105 Cours 4

Patrons de conception

Créatior

C .....

Construction : donnent des manières de créer de nouveaux

objets.

Structuration : donnent des structure de programmes.

Comportement : donnent des moyens d'interaction entre objets

(qui du coup contraignent la structure).



### Les types de patrons de conception

Design

M. Sassola M3105

Patrons de conception

Création

Structuration

Construction : donnent des manières de créer de nouveaux

objets.

Structuration : donnent des structure de programmes.

Comportement : donnent des moyens d'interaction entre objets (qui du coup contraignent la structure).

### Cette année

On ne verra que quelques-un des patrons de conception.



## Les types de patrons de conception

Patrons de

Construction : donnent des manières de créer de nouveaux

objets.

Structuration: donnent des structure de programmes.

Comportement : donnent des moyens d'interaction entre objets (qui du coup contraignent la structure).

### Cette année

On ne verra que quelques-un des patrons de conception.

Souvent, on utilise ces patrons (ou plutôt motifs...) sans s'en rendre compte.





### Plan de la séance

Design Patterns

M3105 Cours 4

Patrons d

#### Création

Fabrique abstrait Singleton

Structuration

Comportemen

- Les patrons de conception
- Patrons de création
  - Fabrique abstraite (abstract factory)
  - Singleton
- Patrons de structuration
- 4 Patrons de comportement (semaine prochaine



### Plan de la séance

- Patrons de création
  - Fabrique abstraite (abstract factory)
  - Singleton



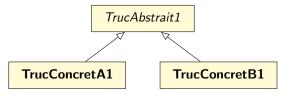


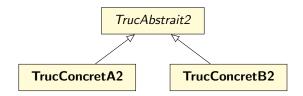


>>> Des objets dont le type précis n'est pas forcément connu sont créés par des classes constructrices.



>>> Des objets dont le type précis n'est pas forcément connu sont créés par des classes constructrices.

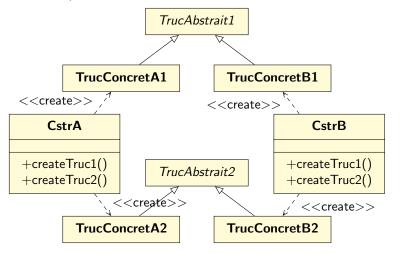






Fabrique abstraite

>>> Des objets dont le type précis n'est pas forcément connu sont créés par des classes constructrices.





### ldée de la solution

Design Patterns

M3105

Patrons o

Créati

Fabrique abstr

----8-----

Structuratio

Comportemen<sup>-</sup>

 Puisque les objets que l'on crée sont des concrétisations d'objets abstraits, on peut également abstraire les constructeurs.

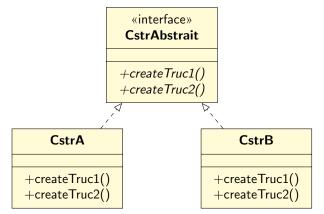


### Idée de la solution



Fabrique abstraite

- ▶ Puisque les objets que l'on crée sont des concrétisations d'objets abstraits, on peut également abstraire les constructeurs.
- On abstrait les constructeurs par une interface.





# Avantages de cette abstraction

Design Patterns M. Sassolas M3105 Cours 4

conception

Fabrique abstraite

Structuration

- ► Un « client » qui manipule des Trucs n'a pas besoin de connaître les détails de l'implémentation du Truc.
- ➤ On peut aisément étendre les Trucs avec une implémentation TrucConcretC simplement en fournissant la classe constructeur idoine CstrC.



# Le diagramme de classe complet Avec un seul Truc

Design Patterns

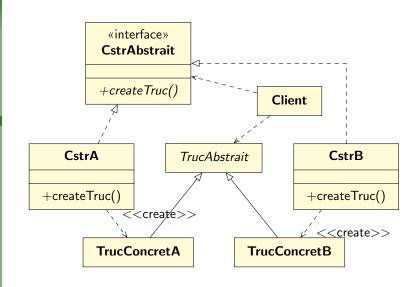
M3105

conception

Fabrique abstrait

Structuration

Comportement





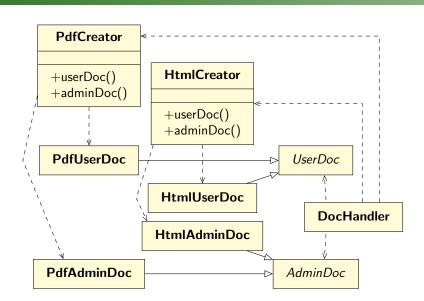
## Exemple : le problème



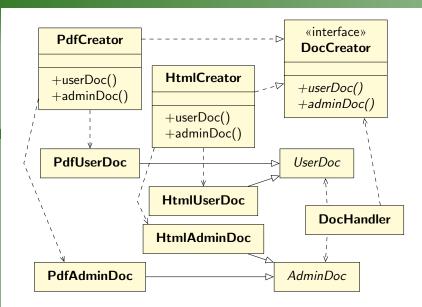
- Fabrique abstraite
- Structuration
  Comportement

- ▶ Dans notre système, on veut pouvoir fournir de la documentation en PDF ou en HTML. On envisage un jour d'ajouter d'autres formats.
- ► La documentation peut concerner l'utilisation de l'interface ou la maintenance du serveur. Ces documentations différentes sont manipulée différemment; par exemple, l'accès à la documentation de maintenance peut inclure une vérification des permissions.
- ► Elle est générée à chaque demande afin de tenir compte des différentes versions de tous les modules installés.











### Plan de la séance

Patterns

M3105 Cours 4

Patrons d conceptio

Création

Singleton

Structuration

Comportement

- 1 Les patrons de conception
- Patrons de création
  - Fabrique abstraite (abstract factory)
  - Singleton
- Patrons de structuration
- 4 Patrons de comportement (semaine prochaine)



Design Patterns

M3105 Cours 4

conceptions c

Création

Singleton

tructuratio

Comportemen

- ▶ Une classe dont on ne veut qu'une seule instance.
- ➤ Exemples courants : classe principale, classe de connexion à la base de donnée, constructeurs d'objets (cf fabrique abstraite)....

ClasseÀInstanceUnique



### Solution

Design Patterns M. Sassolas

Patrons de conception

Creatio

Singleton

Structuration

### ldée

- La classe dispose d'un attribut de classe (static) du même type qui pointe vers l'unique instance.
- À la première instanciation, cet attribut est mis à jour.
- ▶ Pour les suivantes, on renvoie en fait le premier.
- Le véritable constructeur est privé.

### Singleton

- instance : Singleton = null
- +Instance()
- « constructor »-Singleton()

```
if (instance == null)
  // première instanciation
  instance = new Singleton();
return instance;
```



# Remarques

Design Patterns M. Sassolas M3105 Cours 4

conceptic Création

Singleton

Comportement

- ▶ L'appel de l'instance de la classe se fait donc par Singleton.Instance() et non par new Singleton().
- ➤ On utilise un autre « constructeur » Instance() car sinon on aurait une récursion infinie lors de la première instanciation.



Design Patterns

M. Sassolas

Cours 4

Patrons d conception

Création

Fabrique ab Singleton

Structuratio

Comportement

### **PdfCreator**

instance : PdfCreator = null

« constructor »-PdfCreator()

 $\underline{+ \mathsf{Instance}() : \mathsf{PdfCreator}}$ 

 $\overline{+ userDoc() : UserDoc}$ 

+adminDoc() : AdminDoc

Singleton

### **PdfCreator**

```
instance · PdfCreator = null
public class PdfCreator implements DocCreator {
  private static PdfCreator instance = null;
  private PdfCreator(){}
  public static PdfCreator Instance(){
    if (instance == null)
      // première instanciation
      instance = new PdfCreator();
    return instance;
  public UserDoc userDoc(){...}
  public AdminDoc adminDoc(){...}
```

Design Patterns M. Sassolas

Cours 4 Patrons de

conception Création

Singleton

Structuratio

```
pdfCreator
instance · PdfCreator = null
public class PdfCreator implements DocCreator {
   private static PdfCreator instance = null;
   ...
thePdfCreator = PdfCreator.Instance();
AdminDoc freshPdfAdmDoc = thePdfCreator.adminDoc();
   ...
instance = new PdfCreator();
```

return instance;

public UserDoc userDoc(){...}
public AdminDoc adminDoc(){...}



### Plan de la séance

Design Patterns

M3105 Cours 4

Patrons c conceptio

Cicati

### Structuration

Composite
Adaptateur
Décorateur
Façade
Proxy

Comportemen

- Les patrons de conception
- Patrons de création
- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy
- 4 Patrons de comportement (semaine prochaine)



### Plan de la séance

Design Patterns

M3105 Cours 4

Patrons d conceptio

\_\_\_\_\_

Structuratio

### Composite

Décorateur Façade Proxy

Comportemen

- Les patrons de conception
- 2 Patrons de création
- 3 Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy
- 4 Patrons de comportement (semaine prochaine)



parcourir.

Design Patterns M. Sassolas

Patrons de conception Création

Structuration

Composite

Adaptateur Décorateur Façade Proxy intro: Section

titre=« Truc »
preambule= « Blablabla »

corps: Section

titre=« Machin »
preambule= « Blublu »

document: Section

titre=« Bidule »
preambule= « »

Nos objets sont organisés en arbre. Il faut pouvoir les

para1: Section

titre=« Chose »
preambule= « Blabli »

para2: **Section** 

titre=« Chouette » preambule= « Toto »

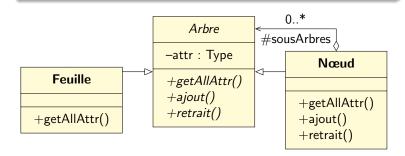


### Solution

Composite

### Idée

- On divise en classe feuille et classe nœud qui héritent d'une classe principale.
- Les accès se font récursivement dans les descendants.
- Il faut prévoir des accesseurs pour ajouter/enlever des composants.





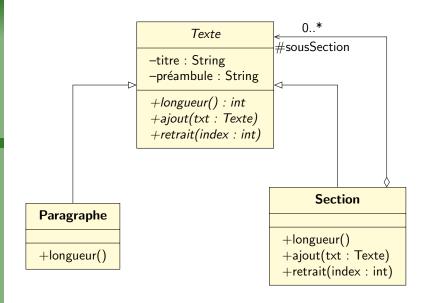
## Solution

```
Composite
```

```
public class Feuille extends Arbre {
Idée
       public Type getAllAttr(){return attr;} //
       public class Nœud extends Arbre{
   II få
                                               des
         public Type getAllAttr(){
    COI
           Type res = attr;
           for (Arbre desc: sousArbres) {
             res.append(desc.getAllAttr());
           };
           return res;
                                               ttr()
  +get/
```



Composite



```
FONTAINEBLEAU
```

Composite

```
public class Paragraphe extends Texte {
   public int longueur(){
     return (titre.length() + préambule.length());
public class Section extends Texte{
  public int longueur(){
```

```
int res = 0:
res = res + titre.length() + préambule.length();
for (Texte subsec: sousSections) {
  res= res + subsec.longueur();
};
return res;
```



#### Plan de la séance

Adaptateur

- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy

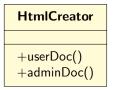


### Problème

Adaptateur

Les différentes implémentations d'une même fonctionnalité sont fournies avec des noms différents, voire en plusieurs parties.

# **PdfCreator** +docEntete() +simpleDocCorps() +fullDocCorps()



### C'est le cas lorsque...

- On travaille avec des classes/composants créés par quelqu'un d'autre (importées).
- On a mal conçu le système au premier abord (il aurait fallu utiliser une fabrique abstraite).

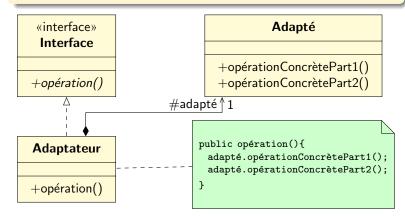


Design
Patterns
M. Sassolas
M3105
Cours 4

Adaptateur

ldée

Une classe Adaptateur s'occupe de la traduction entre les méthodes demandées par l'interface et les méthodes implémentées par l'Adapté.





Design Patterns M. Sassolas M3105 Cours 4 atrons de onception réation

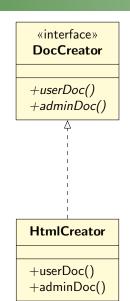
Adaptateur

Décorateur Façade Proxy

Comportemen

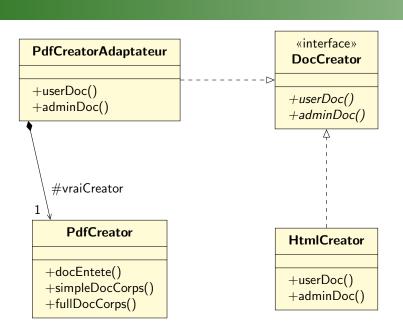
#### **PdfCreator**

+docEntete()
+simpleDocCorps()
+fullDocCorps()



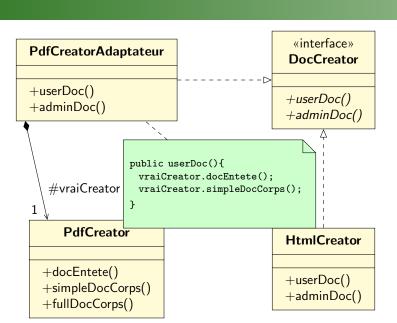








Adaptateur





#### Plan de la séance

Décorateur

- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy



#### Problème

Design Patterns

M3105 Cours 4

conception Création

Structuratior Composite

Décorateur

Façade Proxy

Comporteme

On veut étendre une (ou deux) opération de base dans certains cas autrement que par héritages.

#### Pourquoi pas l'héritage?

- ▶ Pas très joli : hériter juste pour modifier une fonction, c'est moche.
- ▶ Moins flexible : si l'on veut ajouter d'autres extensions, on change drastiquement le modèle.



# Solution Idée

Design Patterns

M3105 Cours 4

conception

Structuration
Composite
Adaptateur

Décorateur

Proxy

Comportemen

- ► On définit un <u>Décorateur</u> qui pointe vers l'objet de base et redéfinit seulement l'opération en question.
- ► On peut définir plusieurs versions (d'autres extensions) en héritant du décorateur.
- ► Le décorateur et la classe de base sont abstraits par une interface.

# Solution Diagramme de classes

Design Patterns

M3105

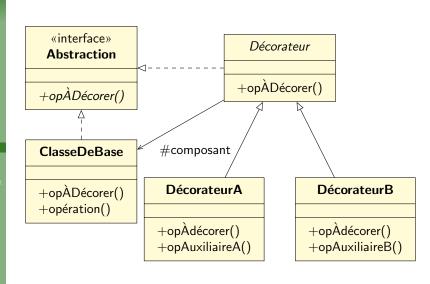
Patrons de conception

Structuratio

Décorateur

Façade

Comportemen



# Solution Diagramme de classes

Design

M3105

Cours 4

Patrons d

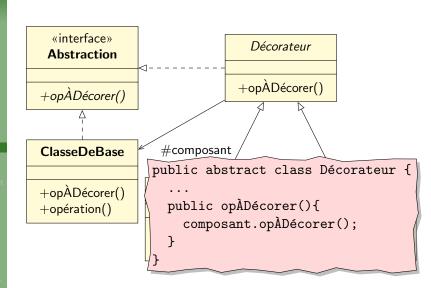
Créat

Structuration Composite

Décorateur

Façade Proxy

Comportemen



# Solution Diagramme de classes

Design Patterns

M3105

Patrons de conception Création

Structuration Composite

Decorateur

```
Proxy
Comportement
```

```
public class DécorateurA extends Décorateur {
  public opADécorer(){
    super.opADécorer();
     opAuxiliaireA();
  public opAuxiliaireA(){...}
ClasseDeBase
                 #composant
                  DécorateurA
                                         DécorateurB
+opÀDécorer()
+opération()
                 +opÀdécorer()
                                       +opÀdécorer()
                 +opAuxiliaireA()
                                       +opAuxiliaireB()
```



Design Patterns

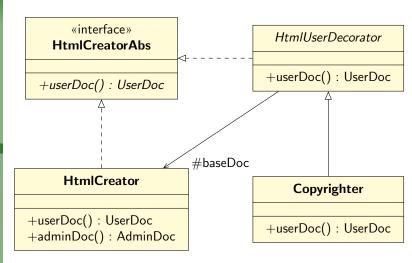
M. Sassolas

Patrons de conception

Structuration

Décorateur

Façade Proxy Comportement



```
public UserDoc userDoc(){
     return baseDoc.userDoc():
                        #baseDoc
     HtmlCreator
                                      Copyrighter
+userDoc(): UserDoc
                                  +userDoc(): UserDoc
```

public abstract class HtmlUserDecorator {

public HtmlUserDecorator(HtmlCreator theDoc){

protected HtmlCreator baseDoc;

this.baseDoc = theDoc;

+adminDoc(): AdminDoc

Design Patterns

M3105

Patrons de conception

Structuration Composite

Adaptateur Décorateur

Façade

```
public abstract class HtmlUserDecorator {
  protected HtmlCreator baseDoc;
  public HtmlUserDecorator(HtmlCreator theDoc){
    this.baseDoc = theDoc;
  }
  public UserDoc userDoc(){
    return baseDoc.userDoc();
  }
}
```

```
public UserDoc userDoc() {
   UserDoc res = super.userDoc();
   res.append("<!-- Copyright 2014 IUT S&M -->");
   return res;
}
```



#### Plan de la séance

Design Patterns

M3105 Cours 4

Patrons c

Créati

Composite

Façade

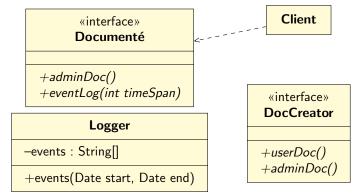
Comportemer

- Les patrons de conception
- Patrons de création
- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy
- 4 Patrons de comportement (semaine prochaine)



### Problème

- Besoin d'une interface qui implémente certaines fonctions.
- Ces fonctions sont implémentées mais par différentes classes/interfaces.
- ▶ Ou bien il ne manque pas grand chose pour les implémenter depuis ces fonctions.



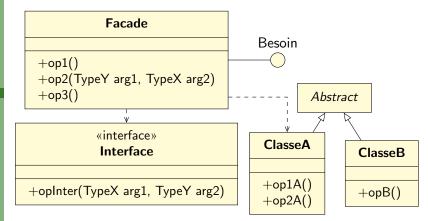
Facade



Facade

#### ldée

Regrouper dans une classe fournissant l'interface tout ce dont on a besoin en le « piochant » ailleurs.

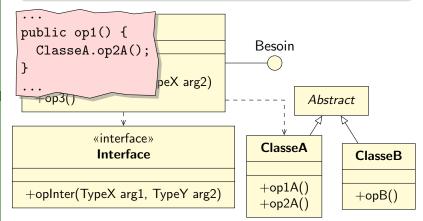




Facade

#### ldée

Regrouper dans une classe fournissant l'interface tout ce dont on a besoin en le « piochant » ailleurs.





Design Patterns M. Sassolas

onception

réation

tructuration

Composite

Adaptateur

Facade

#### Idée

Regrouper dans une classe fournissant l'interface tout ce dont on a besoin en le « piochant » ailleurs.

```
public op1() {
                                  Besoin
  ClasseA.op2A();
        public op2(TypeY arg1, TypeX arg2) {
           Interface.opInter(arg2,arg1);
                                                 ClasseB
            nterrace
                                   +op1A()
+opInter(TypeX arg1, TypeY arg2)
                                                 +opB()
                                   +op2A()
```

Design Patterns M. Sassolas

Patrons de conception Création

Composite
Adaptateur
Décorateur
Facade

#### ldée

Regrouper dans une classe fournissant l'interface tout ce dont on a besoin en le « piochant » ailleurs.

```
public op1() {
                                 Besoin
  ClasseA.op2A();
        public op2(TypeY arg1, TypeX arg2) {
           Interface.opInter(arg2,arg1);
                                      public op3() {
           nterrace
                                        ClasseA.op1A();
+opInter(TypeX arg1, TypeY arg2)
```



Design

M3105

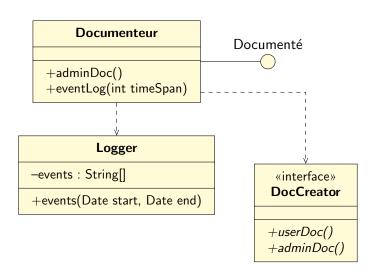
Patrons d conceptio

Creation

Composite Adaptateur

Façade

Comportemer





Design Patterns

M3105

conception

Structuration
Composite
Adaptateur

Façade

Comportement

```
Documenteur
                             Documenté
  +adminDoc()
      public eventLog(int timeSpan) {
        Date now= new Date();
        Date old= new Date();
        old.goBack(timeSpan);
-events
        Logger.events(old,now);
                                         ce»
                                         ator
+event }
                                  +userDoc()
                                  +adminDoc()
```



#### Plan de la séance

Design Patterns

M3105 Cours 4

Patrons d conceptio

Creati

Structuratio

Composite

Adaptateur

Décorateur

Facade

Comportemen

- Les patrons de conception
- Patrons de création
- Patrons de structuration
  - Composite
  - Adaptateur (adapter)
  - Décorateur (decorator)
  - Façade (facade)
  - Proxy
- 4 Patrons de comportement (semaine prochaine)



### Problème

Design Patterns

M3105

conception Création

Structuration
Composite
Adaptateur
Décorateur
Façade

Comportemer

Proxy

- ► Un objet est coûteux à invoquer, on ne veut pas le créer vraiment tant qu'on n'en a pas besoin.
- Ou un objet est sur une autre ressource, que l'on ne veut pas utiliser à chaque fois.
- Ou bien un objet nécessite des droits d'accès que l'on veut vérifier à chaque accès; on souhaite également séparer la vérification de l'accès.

#### Exemple

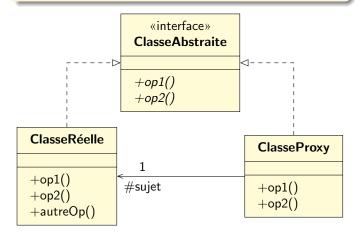
C'est le cas des Query vers les bases de donnée en Java : la requête n'est vraiment exécutée que sur demande expresse.



Proxy

#### ldée

La classe est abstraite par une interface implémentée à la fois par le proxy et par la « vraie » classe.



```
BIUT
       public class ClasseProxy implements ClasseAbstraite {
FONTAINEBLEA
         protected ClasseRéelle sujet = null;
         public op1(){
           if (sujet == null) {
              sujet = new ClasseRéelle();
             // Création
           sujet.op1(); // Délégation
         public op2(){
           if (sujet == null) {
             println("Erreur");
           }else{
           sujet.autreOp();
           sujet.op2(); // Délégation
```



### Plan de la séance

Patterns

M. Sassolas M3105

Patrons d

Créatio

Structuration

Comportement

- 1 Les patrons de conception
- Patrons de création
- Patrons de structuration
- 4 Patrons de comportement (semaine prochaine)