

Architecture Logicielle



Les patrons de responsabilités

Florent Nicart

Université de Rouen

2016–2017

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le patron Singleton

Fournir une (hiérarchie de) classe(s) admettant au plus une instance.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

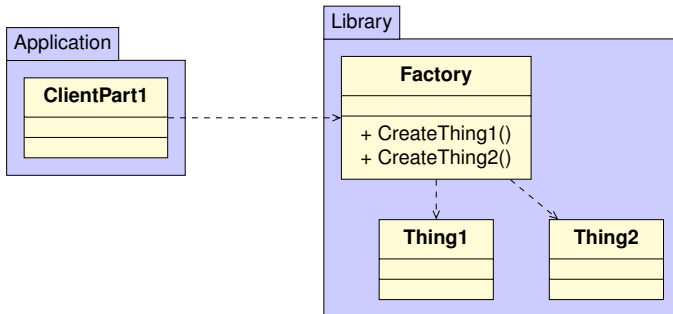
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Situation Initiale



- Une application utilise un service d'une bibliothèque, ici pour produire des objets¹.
- Ce service pourrait-être rendu par des méthodes statiques...

1. Nous verrons les usines au chapitre suivant.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

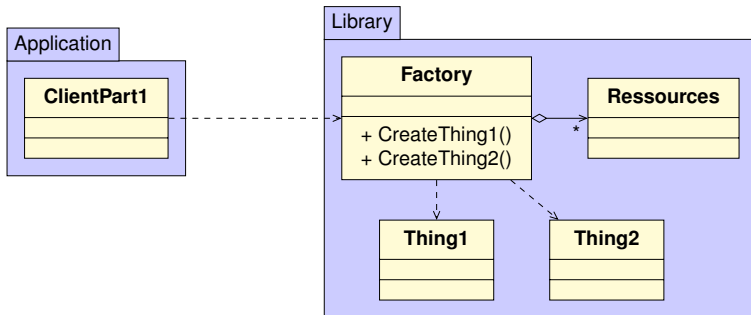
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

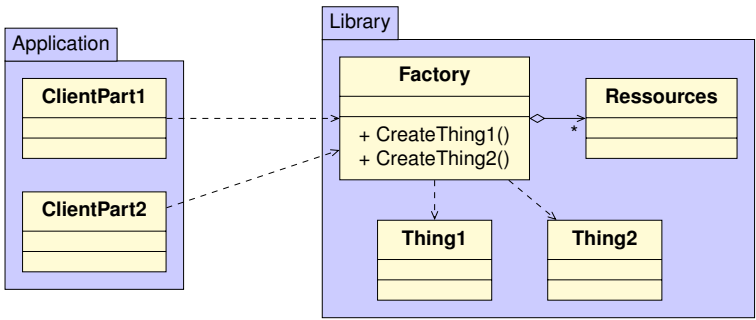
Motivation

Situation Initiale



- ... sauf que ce service requiert des ressources allouées dynamiquement.
- C'est donc bien sur une instance que nous souhaitons travailler.
- Ou bien la nature de cette instance peut varier dynamiquement (voir plus loin).

Situation Initiale



- En général, ces services sont utilisés en plusieurs endroits du code client.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

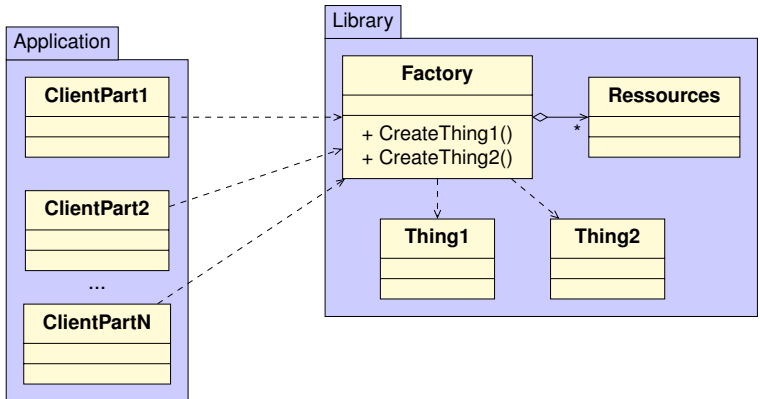
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Situation Initiale



- En général, ces services sont utilisés en plusieurs endroits du code client.
- En général, à beaucoup d'endroits.
- Parfois répartis sur plusieurs autres bibliothèques

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

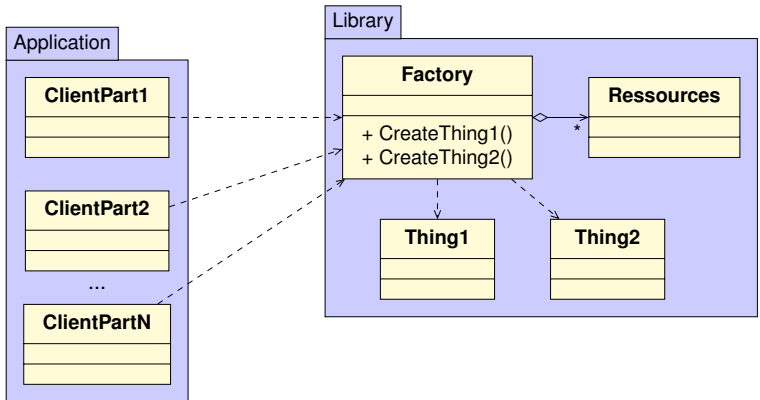
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Situation Initiale



- Des instantiations multiples sont inutiles (gaspillage en espace et en temps),
- conduire à travailler sur des espaces de ressources différents (disfonctionnements)

Une mauvaise solution

Gérer le problème au niveau du client

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

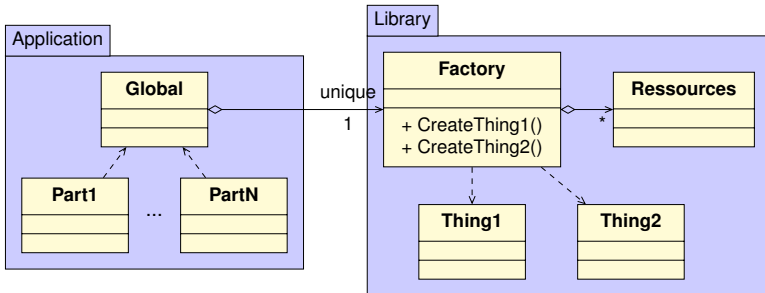
Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



- Problème : le client implémente une solution à un problème imposé par la bibliothèque !
- Solution : faire en sorte que la bibliothèque garantisse elle-même l'unicité grâce au patron **singleton**.

Une mauvaise solution

Gérer le problème au niveau du client

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

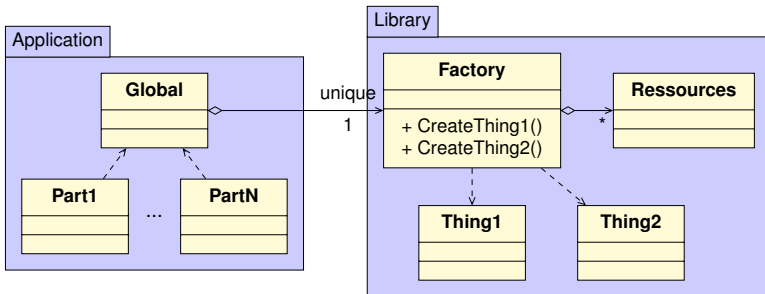
Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



- **Problème** : le client implémente une solution à un problème imposé par la bibliothèque !
- **Solution** : faire en sorte que la bibliothèque garantisse elle-même l'unicité grâce au patron **singleton**.

Une mauvaise solution

Gérer le problème au niveau du client

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

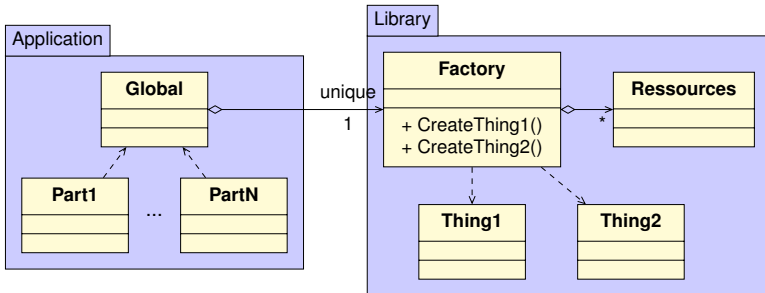
Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



- Problème : le client implémente une solution à un problème imposé par la bibliothèque !
- Solution : faire en sorte que la bibliothèque garantisse elle-même l'unicité grâce au patron **singleton**.

Le patron Singleton

Aussi connu comme Singleton

Intention

- Garantir l'unicité de l'instance d'une classe (ou à l'intérieur d'une arborescence de classes).
- Fournir au client un moyen d'accès simple et fiable à cette instance.

Motivation

- Parfois un service doit être rendu par une instance, mais utiliser Réutilisation d'une boîte à outils dont l'interface n'est pas compatible avec celle conçue pour l'application.
- Parfois, il est nécessaire que cette instance soit unique.

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Participants du patron Singleton

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

- **Singleton** : Classe (ou arborescence de classes) dont on souhaite avoir une unique instance.

Singleton

Motivations

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Ce que l'on veut :

- Disposer d'une instance maximum d'une classe donnée,
- ET donner un accès global à cet objet dans l'application.

Raisons de ne vouloir qu'une instance au plus d'une classe :

- Éviter à tout prix de travailler sur des instances séparées. Ex : contextes (graphiques, bases de données, ...), variables d'environnement/de session, .. ;
- variables globales,

Singleton

Schéma de principe

Singleton

- Motivation
- Structure**
- Conclusion

Observateur

- Motivation
- Structure
- Exemples
- Conclusion

Proxy

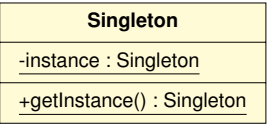
- Motivation
- Structure
- Exemples
 - P.Synchro
 - Copy-On-Write
 - P. Virtuel
- Proxy Dynamique
- Conclusion

Médiateur

- Motivation
- Structure
- Exemples
- Conclusion

Médiateur

- Motivation



Différent types de singleton

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

On distingue deux familles de singletons :

- non dérivables (une seule instance d'une seule classe),
- dérivable (une seule instance d'une arborescence de classes.

pour lesquelles on peut envisager deux grands types d'implémentations :

- instanciation agressive,
- instanciation paresseuse (à la demande).

Singleton non dérivable

Instanciation agressive

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1  /** Class Singleton is an implementation of a class
2  * that only allows one instantiation. */
3  public final class Singleton {
4      // The private reference to the one and only instance.
5      private static Singleton uniqueInstance = new Singleton();
6
7      // An instance attribute.
8      private int data = 0;
9
10     /**
11     * Returns a reference to the single instance.
12     * Creates the instance if it does not yet exist. (This is called lazy
13     instantiation.)
14     */
15     public static Singleton instance() {
16         return uniqueInstance;
17     }
18
19     /** The Singleton Constructor.
20     * Note that it is private! No client can instantiate a Singleton object! */
21     private Singleton() {}
22
23     // Accessors and mutators here!
24     ...
25 }
```


Singleton non dérivable

Instanciation agressive - test

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1 public static void main(String[] args) {
2     // Get a reference to the single instance of Singleton.
3     Singleton s = Singleton.instance();
4
5     // Set the data value.
6     s.setData(34);
7     System.out.println("First_\u005Creference :_\u005C" + s);
8     System.out.println("Singleton_\u005Cdata_\u005Cvalue_\u005Cis :_\u005C" + s.getData());
9
10    // Get another reference to the Singleton.
11    Singleton s1 = Singleton.instance();
12    System.out.println("\nSecond_\u005Creference :_\u005C" + s1);
13    System.out.println("Singleton_\u005Cdata_\u005Cvalue_\u005Cis :_\u005C" + s1.getData());
14    System.out.println("Is_\u005Cit_\u005Cthe_\u005Csame_\u005Cobject?_\u005C" + (s==s1));
15 }
```

First reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Second reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Is it the same object? true

Singleton non dérivable

Instanciation agressive - test

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1 public static void main(String[] args) {
2     // Get a reference to the single instance of Singleton.
3     Singleton s = Singleton.instance();
4
5     // Set the data value.
6     s.setData(34);
7     System.out.println("First_\u00a0reference :_\u00a0" + s);
8     System.out.println("Singleton_\u00a0data_\u00a0value_\u00a0is :_\u00a0" + s.getData());
9
10    // Get another reference to the Singleton.
11    Singleton s1 = Singleton.instance();
12    System.out.println("\nSecond_\u00a0reference :_\u00a0" + s1);
13    System.out.println("Singleton_\u00a0data_\u00a0value_\u00a0is :_\u00a0" + s1.getData());
14    System.out.println("Is_\u00a0it_\u00a0the_\u00a0same_\u00a0object?_\u00a0" + (s==s1));
15 }
```

First reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Second reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Is it the same object? true

Singleton non dérivable

Instanciation paresseuse

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

- Dans certains cas d'utilisation de l'application, le singleton ne sera jamais utilisé.
- Si l'initialisation du singleton est coûteuse (temps/mémoire), on souhaitera créer le singleton uniquement à sa première utilisation.
- On qualifie ce genre de traitements de *à la volé/à la demande (on the fly)* ou de *paresseux/laxiste²(lazy)*.
- Note : ces chargements/calculs à la demande se généralisent et ne sont pas propres au singleton.

Allez, sors-leur ta tirade sur les logiciels mal conçus !

2. Ce qui n'est pas du tout péjoratif, bien au contraire !

Singleton non dérivable

Instanciation paresseuse

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1  /** Class Singleton is an implementation of a class that
2  * only allows one instantiation.
3  */
4  public final class Singleton {
5      // The private reference to the one and only instance.
6      private static Singleton uniqueInstance = null;
7      // The private reference to the current class
8      private static Object lock = Singleton.class;
9      // An instance attribute.
10     private int data = 0;
11     /** The Singleton Constructor. Note that it is private !
12     * No client can instantiate a Singleton object !
13     */
14     private Singleton() {}
15     /**
16     * Returns a reference to the single instance.
17     * Creates the instance if it does not yet exist.
18     * (This is called lazy instantiation.)
19     */
20     public static Singleton instance() {
21         synchronized(lock){
22             if (uniqueInstance == null) {
23                 uniqueInstance = new Singleton();
24             }
25             return uniqueInstance;
26         }
27     }
28     // Accessors and mutators here !
29 }
```

Singleton non dérivable

Instanciation laxiste - test

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1 public static void main(String[] args) {
2     // Get a reference to the single instance of Singleton.
3     Singleton s = Singleton.instance();
4
5     // Set the data value.
6     s.setData(34);
7     System.out.println("First_\u005Creference :_\u005C" + s);
8     System.out.println("Singleton_\u005Cdata_\u005Cvalue_\u005Cis :_\u005C" + s.getData());
9
10    // Get another reference to the Singleton.
11    Singleton s1 = Singleton.instance();
12    System.out.println("\nSecond_\u005Creference :_\u005C" + s1);
13    System.out.println("Singleton_\u005Cdata_\u005Cvalue_\u005Cis :_\u005C" + s1.getData());
14    System.out.println("Is_\u005Cit_\u005Cthe_\u005Csame_\u005Cobject?_\u005C" + (s==s1));
15 }
```

First reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Second reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Is it the same object? true

Singleton non dérivable

Instanciation laxiste - test

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

```
1 public static void main(String[] args) {
2     // Get a reference to the single instance of Singleton.
3     Singleton s = Singleton.instance();
4
5     // Set the data value.
6     s.setData(34);
7     System.out.println("First_\u00a0reference :_\u00a0" + s);
8     System.out.println("Singleton_\u00a0data_\u00a0value_\u00a0is :_\u00a0" + s.getData());
9
10    // Get another reference to the Singleton.
11    Singleton s1 = Singleton.instance();
12    System.out.println("\nSecond_\u00a0reference :_\u00a0" + s1);
13    System.out.println("Singleton_\u00a0data_\u00a0value_\u00a0is :_\u00a0" + s1.getData());
14    System.out.println("Is_\u00a0it_\u00a0the_\u00a0same_\u00a0object?_\u00a0" + (s==s1));
15 }
```

First reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Second reference:

single.eager.Singleton@19821f

Singleton data value is: 34

Is it the same object? true

Singleton non dérivable

Note d'implémentation

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Il semble que l'instanciation paresseuse est

- préférable dans les cas à coût élevés,
- un peu plus complexe à implémenter,
- difficile à garantir dans son fonctionnement,

et que l'instanciation agressive

- convient dans la plupart des cas,
- et est très simple à implémenter,
- sans soucis de concurrence.

Sauf que ... depuis Java 5³ ...

Singleton non dérivable

Note d'implémentation

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Il semble que l'instanciation paresseuse est

- préférable dans les cas à coût élevés,
- un peu plus complexe à implémenter,
- difficile à garantir dans son fonctionnement,

et que l'instanciation agressive

- convient dans la plupart des cas,
- et est très simple à implémenter,
- sans soucis de concurrence.

Sauf que ... depuis Java 5³ ...

Note d'implémentation en Java

Initialization of a class consists of executing its static initializers and the initializers for static fields declared in the class. [...]

A class or interface type T will be initialized immediately before the first occurrence of any one of the following :

- T is a class and an instance of T is created.
- T is a class and a static method declared by T is invoked.
- A static field declared by T is assigned.
- A static field declared by T is used and the field is not a constant variable (§4.12.4).
- T is a top-level class, and an assert statement (§14.10) lexically nested within T is executed.

Invocation of certain reflective methods in class Class and in package java.lang.reflect also causes class or interface initialization. **A class or interface will not be initialized under any other circumstance.**

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Singleton dérivable

il ne peut y en avoir qu'un ?

Que se passe-t-il si l'on dérive un singleton ? (On enlève `final`)

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P. Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

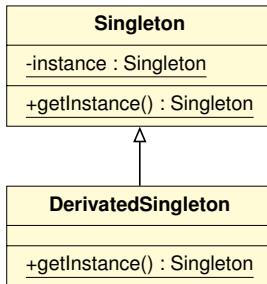
Structure

Exemples

Conclusion

Médiateur

Motivation



Le principe de singleton porte-t-il dans ce cas sur :

- chacune des classes ?
- sur l'arborescence entière ?

Singleton dérivable

il ne peut y en avoir qu'un ?

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P. Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Remarque : si nous voulions deux singletons, aurions pu en écrire un second sans dériver :

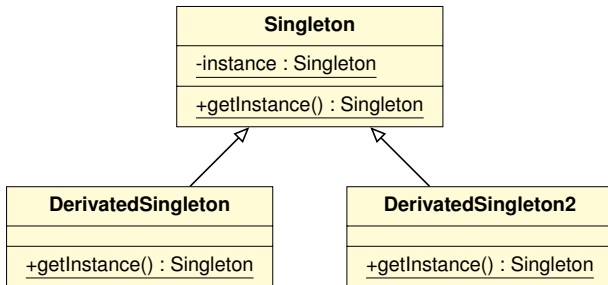
| Singleton |
|-----------------------------------|
| <u>-instance : Singleton</u> |
| <u>+getInstance() : Singleton</u> |

| Singleton2 |
|-----------------------------------|
| <u>-instance : Singleton</u> |
| <u>+getInstance() : Singleton</u> |

Singleton dérivable

il ne peut y en avoir qu'un ?

Le singleton porte sur toute l'arborescence d'héritage :



en particulier parce que l'attribut `instance` est partagé par toutes les classes.

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P. Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

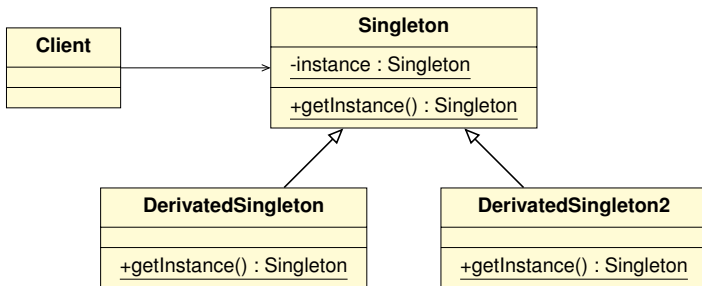
Conclusion

Médiateur

Motivation

Singleton dérivable

Objectif ?



- Le but est ici d'obtenir un singleton polymorphe, dont le type sera choisi dynamiquement,
- pour cela, le client continue de travailler avec la classe de base comme type statique.

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P. Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Singleton dérivable

Instanciation

- Où et comment instancier ? Certainement pas comme ceci :

```
1  class Singleton {  
2      private static instance=new DerivatedSingletonX () ;  
3      ...  
4  }
```

- On évitera également :

```
1  class Singleton {  
2      ...  
3      synchronized public Singleton getInstance(int type) {  
4          switch(type) {  
5              0: instance=new Singleton() ; break ;  
6              1: instance=new DerivatedSingleton1 () ; break ;  
7              ...  
8          }  
9      }  
10 }
```

car ...

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Singleton dérivable

Instanciation 2

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

- Il semble plus raisonnable de s'adresser à la classe en question pour instancier le singleton désiré :

```
1  class DerivatedSingleton1 {  
2      ...  
3      synchronized public Singleton getInstance() {  
4          instance=new DerivatedSingleton1 (); return instance ;  
5      }  
6  }  
7      ...  
8  class DerivatedSingleton2 {  
9      ...  
10     synchronized public Singleton getInstance() {  
11         instance=new DerivatedSingleton2 (); return instance ;  
12     }  
13 }
```

Singleton dérivable

Utilisation

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

- Toutefois cette approche peut être confusante pour l'utilisateur :

```
1  public class Client {  
2      public void test() {  
3          DerivatedSingleton2.getInstance().doSomething();  
4              // DerivatedSingleton2  
5          Singleton.getInstance().doSomething();  
6              // DerivatedSingleton2  
7          DerivatedSingleton1.getInstance().doSomething();  
8              // DerivatedSingleton2  
9      }  
10 }
```

- Le premier usage détermine le type pour toutes les demandes suivantes,
- ... ce qui est normale !
- une telle architecture enfreint le ...

Singleton dérivable

Utilisation

- Une meilleure approche consisterait à considérer la classe de base comme unique point d'accès au singleton et d'équiper les classes dérivées d'une méthode avec une sémantique de définition uniquement. Par exemple :

```
1  class Singleton {
2      synchronized public static Singleton getInstance() {
3          if (instance==null) throw new UndefinedSingletonException();
4          return instance;
5      }
6      synchronized protected static void setInstance(Singleton ins) {
7          instance=ins;
8      }
9  }
10 class DerivatedSingleton1()
11     public static void setInstance() {
12         if (instance==null) { instance=new DerivatedSingleton1(); }
13         else
14             if (!instance instanceof DerivatedSingleton1) {
15                 throw new AlreadyDefinedWithOtherTypeException();
16             }
17     }
18
19 class DerivatedSingleton2() { idem ...
```

Singleton

Motivation

Structure

Conclusion

Observateur

Motivation

Structure

Exemples

Conclusion

Proxy

Motivation

Structure

Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation

Structure

Exemples

Conclusion

Médiateur

Motivation

Principes respectés

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- **S.R.P.** et **I.S.P.** : dépend des autres fonctionnalités, si l'on considère que le singleton n'est pas une responsabilité.
- **O.C.P.** : il faudra être vigilant sur certains choix d'implémentation. Par exemple, on pourra d'abord choisir un singleton non dérivable puis changer d'avis sans violer l'OCP, le contraire n'est pas vrai.
- **L.S.P.** : ok si on se repose uniquement sur la classe de base pour l'accès.
- **D.I.P.** : dépend du reste.

Liens avec les autres patrons ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le singleton est souvent utilisé avec le patron *Abstract Factory* lorsqu'il n'est pas nécessaire de multiplier les instances d'une usine abstraite.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

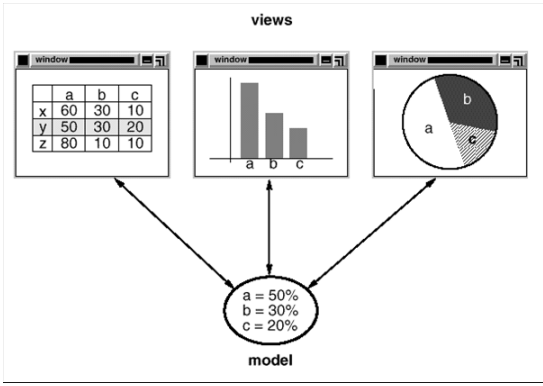
Motivation

Le patron Observateur

Permet de synchroniser l'état de plusieurs objets dans une relation un-à-plusieurs.

Motivation

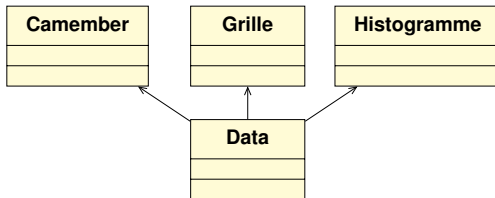
Usage classique : modèle-vue



- La notion d'observateur est fréquemment utilisé pour synchronizer des vues avec l'état d'un modèle.

Motivation

À ne pas faire !



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

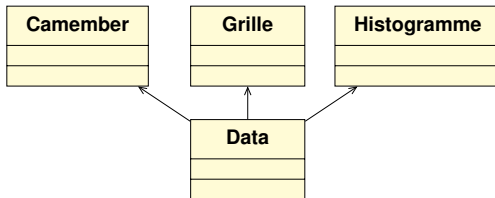
Motivation

```
1 public class Data {
2     private Camembert cam;
3     private Grille gr;
4     private Histogramme hist;
5     ...
6
7     public setA(float a) {
8         this.a = a;
9         if (cam!=null) cam.updateA(a)
10            ;
11         if (gr!=null) gr.updateA(a) ;
12         if (hist!=null) hist.updateA(
13             a) ;
14     }
15 }
```

- Couplage du code des classe,
- Data devrait être un TDA,
- Configuration dynamique rigide,

Motivation

À ne pas faire !



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

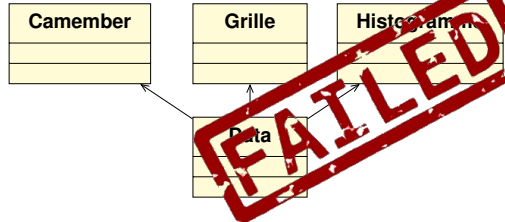
Motivation

```
1 public class Data {
2     private Camembert cam;
3     private Grille gr;
4     private Histogramme hist;
5     ...
6
7     public setA(float a) {
8         this.a = a;
9         if (cam != null) cam.updateA(a)
10            ;
11         if (gr != null) gr.updateA(a) ;
12         if (hist != null) hist.updateA(
13             a) ;
14     }
15 }
```

- Couplage du code des classe,
- Data devrait être un TDA,
- Configuration dynamique rigide,

Motivation

À ne pas faire !



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 public class Data {
2     private Camembert cam;
3     private Grille gr;
4     private Histogramme hist;
5     ...
6
7     public setA(float a) {
8         this.a = a;
9         if (cam != null) cam.updateA(a)
10            ;
11         if (gr != null) gr.updateA(a);
12         if (hist != null) hist.updateA(
13             a);
14     }
15 }
```

- Couplage du code des classe,
- Data devrait être un TDA,
- Configuration dynamique rigide,

Le patron **Observateur**

Aussi connu comme

Dependents, Publish-Subscribe, Model-View

Intention

- Définir une dépendance « un à plusieurs » dynamiquement entre un objet (l'observé) et plusieurs autres objets (observateurs).
- lorsque l'objet observé (le modèle) change d'état, tous les objets dépendants/observateurs (des vues/autres modèles) sont notifiés et mis à jour automatiquement.

Motivation

- Maintenir la relation de manière consistante tout en couplant les classes le plus faiblement possible.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Participants du patron **Observateur**

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

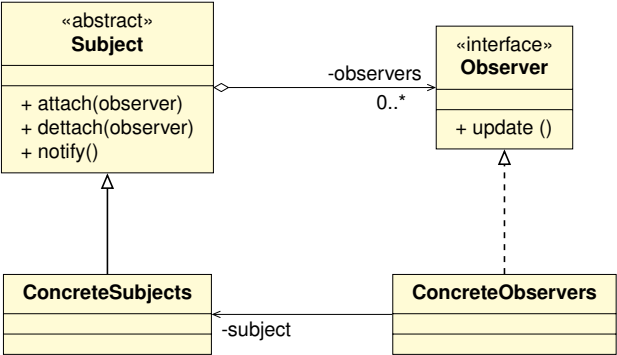
Médiateur

Motivation

- **Subject** : Souche de l'observé, maintient le lien avec les observateurs et définit l'interface de notification.
- **ConcreteSubjects** : Sujet(s) réel(s) notifiant les observateurs lors d'un changement d'état.
- **Observer** : Définit l'interface de notification des objets observateurs.
- **ConcreteObservers** : Classes pouvant être observatrices de l'état d'un autre objet.

Structure

Diagramme de principe (original)



Singleton

- Motivation
- Structure
- Conclusion

Observateur

- Motivation
- Structure**
- Exemples
- Conclusion

Proxy

- Motivation
- Structure
- Exemples
 - P.Synchro
 - Copy-On-Write
 - P. Virtuel
- Proxy Dynamique
- Conclusion

Médiateur

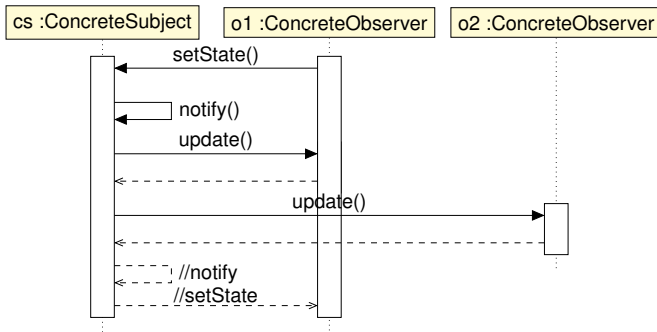
- Motivation
- Structure
- Exemples
- Conclusion

Médiateur

- Motivation

Structure

Fonctionnement



- Toute modification (souvent provoquée par un observateur) est notifiée à tous les observateurs enregistrés.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Avantages

- Couplage minimal entre le modèle Observable et ses Observers :
 - Les modèles sont réutilisables indépendamment des observateurs.
 - Ajout de nouveaux type d'observateurs sans modifier le modèle.
 - Le seul couplage du modèle concerne l'interface avec la méthode `update`.
 - Modèles et observateurs peuvent appartenir à des couches d'abstraction distinctes.
- Support de la diffusion événements :
 - Le modèle envoie des notifications aux observateurs abonnés.
 - Les observateurs peuvent ajoutés/supprimés dynamiquement.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Inconvénients

- Effet possible d'embrication en cascade des notifications :
 - Les observateurs n'ont pas à être conscient des autres aussi doivent-ils déclencher avec soin les mises à jour, surtout si l'un d'eux sert également de contrôleur.
 - Une interface de mise à jour trop simple nécessite que les observateurs repèrent les objets modifiés.
- Éviter de construire des cycles : si un observateur est également observable, il est possible de déclencher une récursion infinie.
- Les sources de mises à jour peuvent être de plusieurs natures :
 - le modèle lui-même,
 - les observateurs,
 - un acteur tiers.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Réalisation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Implémentation de la liste des observateurs (ArrayList, LinkedList, BTree) : les opérations d'ajout/suppression doivent elles être plus performantes que la notification ?
- Comment distinguer l'émetteur d'une notification lorsqu'un observateur observe plusieurs modèles ?
- S'assurer que le modèle met à jour son état et le stabilise avant de déclencher la notification.
- Combien d'information le modèle envoie-t-il sur les changements qu'il a subit :
 - *push model* : le modèle transmet ce qui a été modifié,
 - *pull model* : l'observateur vient chercher le nouvel état depuis le modèle.

Réalisation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Les observateurs peuvent souscrire aux seuls événements qui les intéressent. Dans ce cas le modèle filtre au moment de la notification.
- Il peut arriver qu'un observateur qui observe plusieurs modèles simultanément ne puisse entreprendre sa tâche qu'une fois que tous les observés ont changé d'état :
 - dans ce cas on utilisera un intermédiaire qui jouera le rôle de médiateur (voir le patron du même nom),
 - les modèles notifient à tour de rôle le médiateur de leurs changements d'état et celui-ci réalise les traitements nécessaires avant d'invoquer l'observateur final.

Réalisation

l'API Java

- L'API Java (`java.util`) fournit une souche pour implémenter le patron observateur/observé :

```
1 package java.util;
2
3 public class Observable {
4     // Construct an Observable with zero Observers :
5     public Observable();
6     // Adds an observer to the set of observers of this object :
7     public synchronized void addObserver(Observer o);
8     // Deletes an observer from the set of observers of this object :
9     public synchronized void deleteObserver(Observer o);
10
11     // Indicates that this object has changed since last notification :
12     protected synchronized void setChanged();
13     // Indicates that this object no longer has changed :
14     protected void clearChanged();
15     // Tests if this object has changed :
16     public synchronized boolean hasChanged();
17
18     // Notify all the observers if this object has changed :
19     public void notifyObservers(Object arg);
20     public void notifyObservers();
21     ...
22 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- De même pour créer une classe observateur :

```
1 package java.util ;  
2  
3 public interface Observer {  
4     public abstract void update(Observable o, Object arg) ;  
5 }
```

- This method is called whenever the observed object is changed. An application calls an observable object's notifyObservers method to have all the object's observers notified of the change.
- Parameters :
 - o : the observable object that triggered the notification,
 - arg : the optional parameter passed by the observable.

Exemple 1

L'observable

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure

Exemples

Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 package myobserver;
2 import java.util.Observable;
3
4 public class ConcreteSubject extends Observable {
5     private String name;
6     private float price;
7
8     public ConcreteSubject(String name, float price) {
9         this.name = name;
10        this.price = price;
11        System.out.println("ConcreteSubject created : " + name + " at " + price);
12    }
13
14    public String getName() { return name; }
15
16    public float getPrice() { return price; }
17
18    public void setName(String name) {
19        this.name = name;
20        setChanged();
21        notifyObservers(name);
22    }
23
24    public void setPrice(float price) {
25        this.price = price;
26        setChanged();
27        notifyObservers(new Float(price));
28    }
29 }
```

Exemple 1

Observateur de changement nom

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples

Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 package myobserver;
2 import java.util.Observable;
3 import java.util.Observer;
4
5 public class NameObserver implements Observer {
6     private String name;
7
8     public NameObserver() {
9         name = null;
10        System.out.println("NameObserver created : Name is "+name);
11    }
12
13    public void update(Observable obj, Object arg) {
14        if (arg instanceof String) { // Beurk !!!!
15            name = (String) arg;
16            System.out.println("NameObserver : Name changed to "+name);
17        } else {
18            System.out.println("NameObserver : Some other change to subject !");
19        }
20    }
21 }
```

Exemple 1

Observateur de changement prix

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples

Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 package myobserver;
2 import java.util.Observable;
3 import java.util.Observer;
4
5 public class PriceObserver implements Observer {
6     private float price;
7
8     public PriceObserver() {
9         price = 0;
10        System.out.println("PriceObserver created : price is " + price);
11    }
12
13    public void update(Observable obj, Object arg) {
14        if (arg instanceof Float) { // Beurk !!!!
15            price = ((Float) arg).floatValue();
16            System.out.println("PriceObserver : price changed to " + price);
17        } else {
18            System.out.println("PriceObserver : Some other change to subject");
19        }
20    }
21 }
```

Exemple 1

Programme de test

```
1 public class Test {  
2     public static void main(String args[]) {  
3         // Create the Subject and Observers.  
4         ConcreteSubject s = new ConcreteSubject("Corn_Pops", 1.29f);  
5         NameObserver nameObs = new NameObserver();  
6         PriceObserver priceObs = new PriceObserver();  
7         // Add those Observers!  
8         s.addObserver(nameObs);  
9         s.addObserver(priceObs);  
10        // Make changes to the Subject.  
11        s.setName("Frosted_Flakes");  
12        s.setPrice(4.57f);  
13        s.setPrice(9.22f);  
14        s.setName("Sugar_Crispies");  
15    }  
16 }
```

Produit :

```
1 ConcreteSubject created : Corn Pops at 1.29  
2 NameObserver created : Name is null  
3 PriceObserver created : Price is 0.0  
4 PriceObserver : Some other change to subject!  
5 NameObserver : Name changed to Frosted Flakes  
6 PriceObserver : Price changed to 4.57  
7 NameObserver : Some other change to subject!  
8 PriceObserver : Price changed to 9.22  
9 NameObserver : Some other change to subject!  
10 PriceObserver : Some other change to subject!  
11 NameObserver : Name changed to Sugar Crispies
```

Exemple 1

Problème

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Supposons que la classe qui doit devenir observable soit déjà incluse dans une hiérarchie :

```
1 public classe ConcreteSubject extends ParentClass {...
```

- Java ne supportant pas l'héritage multiple de classe⁴, `ConcreteSubject` ne peut étendre à la fois les classes `Observable` et `ParentClass` !
- De plus, cela viole la règle de codage 4 : « ne pas hériter d'une classe utilitaire » !
- Solution : utiliser la composition⁵ pour envelopper un objet observable.

4. Et c'est tant mieux !

5. Quelle surprise !

Exemple 1

Problème No 2

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples

Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- On note justement que la classe `Observable` de l'API Java a le bon goût de ne pas être abstraite,
- malheureusement ses concepteurs ont donné la visibilité *protégée* aux méthodes de gestion du drapeau de modification :

```
1 package java.util ;  
2 public class Observable {  
3     ...  
4     // Indicates that this object has changed since last notification :  
5     protected synchronized void setChanged() ;  
6     // Indicates that this object no longer has changed :  
7     protected void clearChanged ()  
8     ...  
9 }
```

- forçant à dériver cette classe de toute façon ! :-/
- Nous utiliserons une classe supplémentaire qui sera notre observateur délégué.

Exemple 2

Observable délégué

```
1 public class ConcreteSubject extends ParentClass {
2     private String name;     private float price;
3     private Observable obs;
4
5     public ConcreteSubject(String name, float price) {
6         this.name = name;     this.price = price;
7         System.out.println("ConcreteSubject created : "+name+" at "+price);
8         obs = new Observable() { // Utilisation d'une classe anonyme
9             public void setChanged() { super.setChanged(); }
10            public void clearChanged() { super.clearChanged(); }
11        }
12    }
13
14    public String getName() { return name; }
15    public float getPrice() { return price; }
16
17    public void setName(String name) {
18        this.name = name;
19        obs.setChanged();
20        obs.notifyObservers(name);
21    }
22    public void setPrice(float price) {
23        this.price = price;
24        obs.setChanged();
25        obs.notifyObservers(new Float(price));
26    }
27
28    public void addObserver(Observer o) { obs.addObserver(o); }
29    public void deleteObserver(Observer o) { obs.deleteObserver(o); }
30 }
```

Principes respectés

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- **S.R.P.** : ?
- **O.C.P.** : ?
- **L.S.P.** : ?
- **I.S.P.** : XXX.
- **D.I.P.** : ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

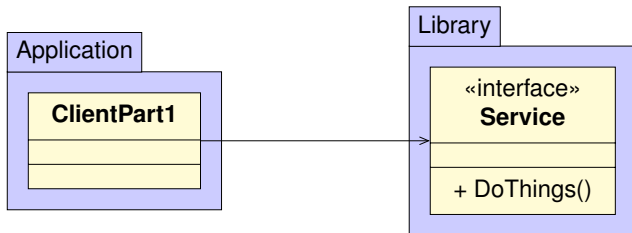
Médiateur

Motivation

Le patron Proxy

Contrôle l'accès à un objet au moyen d'un intermédiaire.

situation initiale



- Une application (existante) utilise les services d'une bibliothèque (existante).
- On souhaite ajouter des contrôles ou des comportements dans la relation entre l'application et le ou les composante de la bibliothèque.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

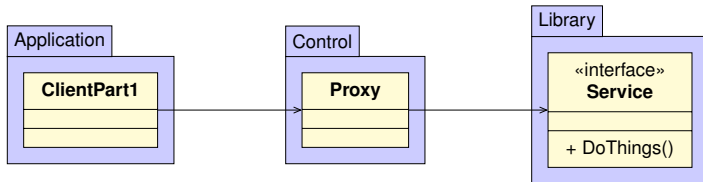
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

situation initiale

- On ne peut modifier ni l'un ni l'autre, mais l'on peut toutefois modifier le lien associatif entre-eux.
- Solution : intercaler un objet intermédiaire qui ajoutera les nouveaux comportements sans que le client "s'en rende compte" ⁶ et délèguera à la bibliothèque.



6. Sans que le client ait besoin d'être modifié.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le patron Proxy/Mandataire

Aussi connu comme
Surrogate (subrogé)

Intention

Fournir un intermédiaire qui permette de contrôler l'accès à un objet.

Motivation

- Ajouter un contrôle ou des traitements lorsqu'un client accède à un objet.
- Le client ne "sait" pas qu'il s'adresse à un proxy.
- le proxy délègue les traitements à l'objet réel.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy/mandataire

Différents types/motivations

- **Remote Proxy** : référence un objet situé dans un espace d'adresse différent (voir java.rmi)
- **Virtual Proxy** : permet la création laxiste d'une copie en mémoire rapide d'un objet.
- **Copy-On-Write Proxy** : diffère le clonage des objets tant qu'aucune opération de modification ne les différencie. C'est une forme de virtual proxy.
- **Protection (Access) Proxy** : fournit aux clients des niveaux d'accès différents à l'objet cible.
- **Cache Proxy** : permet de mémoriser et partager les résultats d'opérations coûteuses .
- **Firewall Proxy** : protège la cible des mauvais clients (ou vice versa)
- **Synchronization Proxy** : gère les accès multiples à la cible.
- **Smart Reference Proxy** : fournit des opérations complémentaires sur la référence. (voir java.lang.ref)

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Participants du patron Proxy

- **Subject** : L'interface définissant le comportement des classes dont les objets seront contrôlés
- **RealSubject** : Les classes dont les instances pourront être contrôlées par le Proxy.
- **Proxy** :
 - implémente l'interface Sujet afin de pouvoir se substituer au sujet réel ;
 - délègue les actions au sujet réel (maintien une référence vers celui-ci) ;
 - implémente le contrôle à effectuer ou les comportements à ajouter.
- **Client** : Code utilisateur qui manipulera indifféremment le proxy ou le sujet réel

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

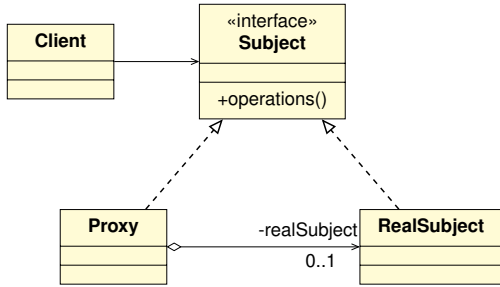
Motivation
Structure
Exemples
Conclusion

Médiateur

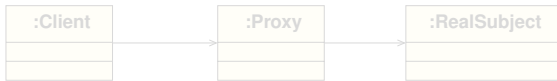
Motivation

Patron Proxy

Schéma de principe



- Le client manipule le sujet indirectement à travers le proxy :



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation

Structure

Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

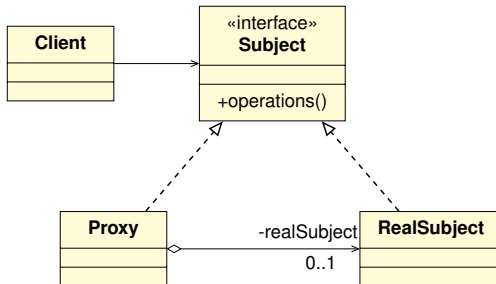
Motivation
Structure
Exemples
Conclusion

Médiateur

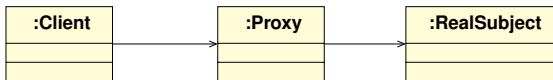
Motivation

Patron Proxy

Schéma de principe



- Le client manipule le sujet indirectement à travers le proxy :



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation

Structure

Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

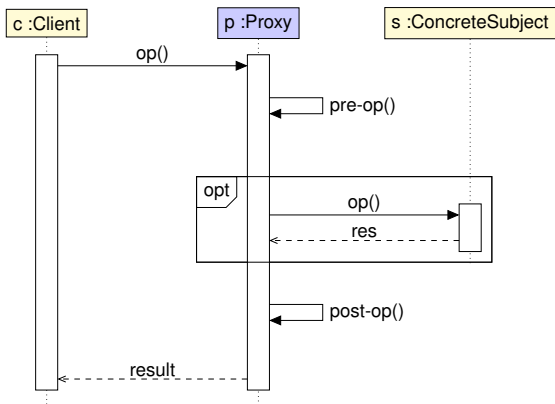
Médiateur

Motivation

Patron Proxy

Schéma de principe

- Le proxy ajoute des opérations des comportements ou des contrôles :



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Ex1 : Proxy de synchronisation

Situation de départ

- Une bibliothèque fournit une implémentation d'une table :

```
1  public interface ITable<T> {  
2      public T getElementAt(int row, int column);  
3      public void setElementAt(T element, int row, int column);  
4      public int getNumberOfRows();  
5  }  
6  ...  
7  public class Table<T> implements ITable<T> {  
8      // ...  
9      int numRows;  
10     public T getElementAt(int row, int column) {  
11         // Get the element.  
12         return ...;  
13     }  
14     public void setElementAt(T element, int row, int column) {  
15         // Set the element.  
16     }  
17     public int getNumberOfRows() {  
18         return numRows;  
19     }  
20 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Ex1 : Proxy de synchronisation

Problème

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- L'auteur de la bibliothèque originale n'a pas prévu un usage en environnement concurrent (*thread safety*).
- Notre application comporte plusieurs fils d'exécution et nécessite des garanties de synchronisation.
- Peut-on modifier la bibliothèque originale ?
- Techniquement, l'ajout de la synchronisation ne change pas le contrat, toutefois :
 - changerions nous le comportement ? ⁷
 - Sommes nous les auteurs de la bibliothèque ? (mise-à-jours/patches)
- Nous allons ici préférer ajouter le nouveau comportement au dessus de l'implémentation existante ⁸ à l'aide d'un proxy.

7. Ici ajout d'un coût pour les autres clients.

8. Solution proposée par Roger Whitney

Ex1 : Proxy de synchronisation

Problème

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- L'auteur de la bibliothèque originale n'a pas prévu un usage en environnement concurrent (*thread safety*).
- Notre application comporte plusieurs fils d'exécution et nécessite des garanties de synchronisation.
- Peut-on modifier la bibliothèque originale ?
- Techniquement, l'ajout de la synchronisation ne change pas le contrat, toutefois :
 - changerions nous le comportement ? ⁷
 - Sommes nous les auteurs de la bibliothèque ? (mise-à-jours/patches)
- Nous allons ici préférer ajouter le nouveau comportement au dessus de l'implémentation existante ⁸ à l'aide d'un proxy.

7. Ici ajout d'un coût pour les autres clients.

8. Solution proposée par Roger Whitney

Ex1 : Proxy de synchronisation

Problème

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

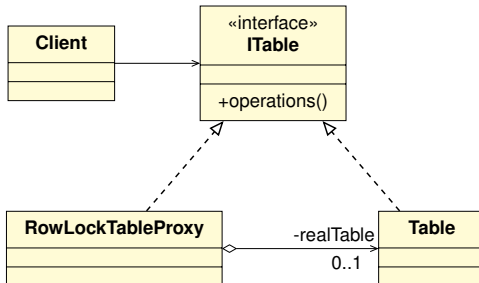
- L'auteur de la bibliothèque originale n'a pas prévu un usage en environnement concurrent (*thread safety*).
- Notre application comporte plusieurs fils d'exécution et nécessite des garanties de synchronisation.
- Peut-on modifier la bibliothèque originale ?
- Techniquement, l'ajout de la synchronisation ne change pas le contrat, toutefois :
 - changerions nous le comportement ? ⁷
 - Sommes nous les auteurs de la bibliothèque ? (mise-à-jours/patches)
- Nous allons ici préférer ajouter le nouveau comportement au dessus de l'implémentation existante ⁸ à l'aide d'un proxy.

7. Ici ajout d'un coût pour les autres clients.

8. Solution proposée par Roger Whitney

Ex1 : Proxy de synchronisation

Solution



- À l'utilisation, on écrira plutôt :

```
1 public class Client {
2     public ITable getTable() {
3         return new RowLockTableProxy<Integer>(new Table<Integer>(...));
4     }
5 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples

P.Synchro

Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Ex1 : Proxy de synchronisation

Solution

- On propose un verrou par ligne :

```
1 public class RowLockTableProxy<T> implements ITable<T> {
2     Table<T> realTable ;
3     Integer[] locks ;
4
5     public RowLockTableProxy(Table<T> toLock) {
6         realTable = toLock ;
7         locks = new Integer[toLock.getNumberOfRows()];
8         for (int row = 0; row < toLock.getNumberOfRows(); row++)
9             locks[row] = new Integer(row);
10    }
11
12    public T getElementAt(int row, int column) {
13        synchronized (locks[row]) {
14            return realTable.getElementAt(row, column);
15        }
16    }
17    public void setElementAt(T element, int row, int column) {
18        synchronized (locks[row]) {
19            realTable.setElementAt(element, row, column);
20        }
21    }
22    public int getNumberOfRows() {
23        return realTable.getNumberOfRows();
24    }
25 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtual
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Situation initiale

- Un programme opère sur des copies partagées et/ou séparées d'une même structure de données :

```
1  ...
2  Integer algorithm(HashTable<Integer> h) {
3      // Sous-algorithmes :
4      subAlgo1(h); // modifications globale.
5      int r1=subAlgo2(h.clone()); // Modif. locales (ou pas !)
6      int r2=subAlgo4(h.clone()); // Modif. locales (ou pas !)
7
8      int theBigResult=subAlgo3(h, r1, r2); // pas de modification.
9
10     return theBigResult;
11 }
12 ...
```

- certaines sous programmes travaillant sur des copies peuvent ne pas modifier ces dernières.
- On considère que la table est assez grosse et que les appels sans modification sont nombreux.
- Dans ce cas la copie induit un coût inutile.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples

P.Synchro

Copy-On-Write

P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Solution

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

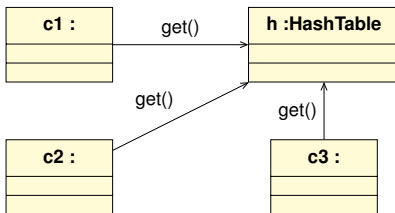
Médiateur

Motivation

- On se propose d'utiliser la technique de *copie à l'écriture* (*copy-on-write*).
- Cette technique est largement utilisée, notamment dans les implémentations de systèmes de fichiers (ZFS, Btrfs, LVM, etc.)
- Là aussi, nous allons souhaiter conserver l'implémentation initiale intacte et ajouter une nouvelle version, sans toutefois tout réécrire...

Exemple 2 : Copy-on-Write

Fonctionnement



- Tous les clients travaillent sur la même copie,
- le premier client qui effectue une modification,
- provoque une duplication de la structure, il sera le seul concerné par sa modification et continuera de travailler sur sa propre copie...

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

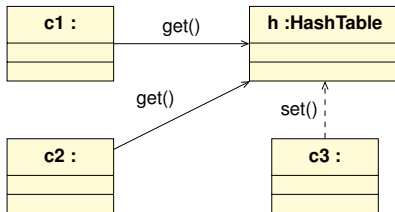
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Fonctionnement



- Tous les clients travaillent sur la même copie,
- le premier client qui effectue une modification,
- provoque une duplication de la structure, il sera le seul concerné par sa modification et continuera de travailler sur sa propre copie...

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

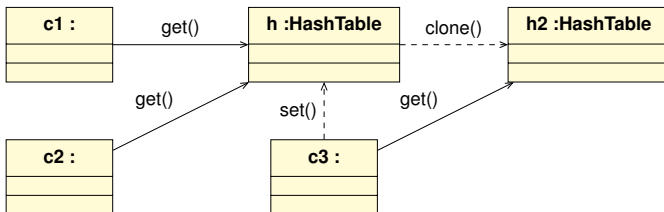
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Fonctionnement



- Tous les clients travaillent sur la même copie,
- le premier client qui effectue une modification,
- provoque une duplication de la structure, il sera le seul concerné par sa modification et continuera de travailler sur sa propre copie...

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Raffinement

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

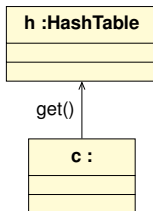
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- On souhaite bien sur éviter ce cas :



- Un seul client manipule la structure,
- s'il opère une modification,
- alors la duplication n'est pas nécessaire...

→ il faut donc procéder au clonage seulement lorsque plus d'un client utilise la table.

Exemple 2 : Copy-on-Write

Raffinement

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

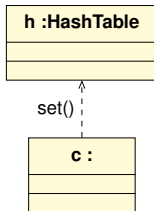
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- On souhaite bien sur éviter ce cas :



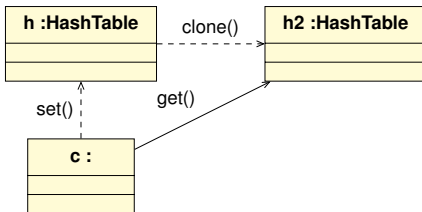
- Un seul client manipule la structure,
- s'il opère une modification,
- alors la duplication n'est pas nécessaire...

→ il faut donc procéder au clonage seulement lorsque plus d'un client utilise la table.

Exemple 2 : Copy-on-Write

Raffinement

- On souhaite bien sur éviter ce cas :



- Un seul client manipule la structure,
- s'il opère une modification,
- alors la duplication n'est pas nécessaire...

→ il faut donc procéder au clonage seulement lorsque plus d'un client utilise la table.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

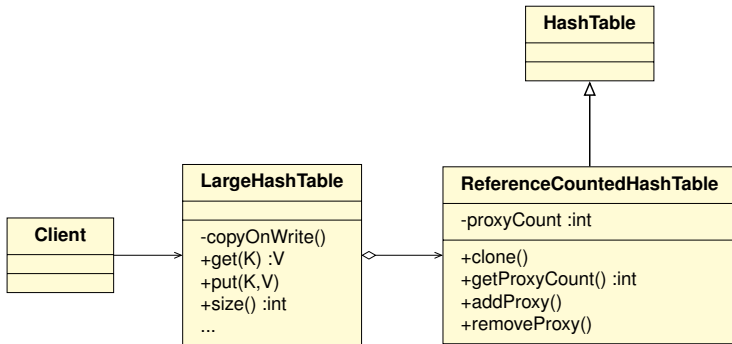
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Solution



- **LargeHashTable** est le proxy,
- **ReferenceCountedHashTable** est une classe équipant la classe *HashTable* d'un comptage de référence.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 2 : Copy-on-Write

Implémentation 1/2

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1  private class ReferenceCountedHashTable<K,V> extends Hashtable<K,V> {
2      private int proxyCount = 1;
3      // Constructor
4      public ReferenceCountedHashTable() {
5          super();
6      }
7      // Return a copy of this object with proxyCount set back to 1.
8      public synchronized Object clone() {
9          ReferenceCountedHashTable<K,V> copy;
10         copy = (ReferenceCountedHashTable<K,V>) super.clone();
11         copy.proxyCount = 1;
12         return copy;
13     }
14     // Return the number of proxies using this object.
15     synchronized int getProxyCount() {
16         return proxyCount;
17     }
18     // Increment the number of proxies using this object by one.
19     synchronized void addProxy() {
20         proxyCount++;
21     }
22     // Decrement the number of proxies using this object by one.
23     synchronized void removeProxy() {
24         proxyCount--;
25     }
26 }
```

Exemple 2 : Copy-on-Write

Implémentation 2/2

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

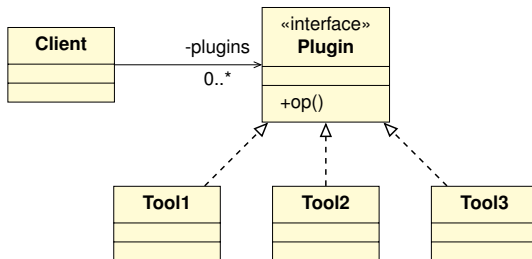
Motivation

```
1 public class LargeHashtable<K,V> implements Map<K,V> {
2     private ReferenceCountedHashTable<K,V> theHashTable ;
3
4     public LargeHashtable() {
5         theHashTable = new ReferenceCountedHashTable<K,V>();
6     }
7     public int size() { return theHashTable.size(); }
8     public synchronized V get(K key) { return theHashTable.get(key); }
9     public synchronized V put(K key, V value) {
10         copyOnWrite();
11         return theHashTable.put(key, value);
12     }
13     public synchronized Object clone() {
14         Object copy = super.clone();
15         theHashTable.addProxy();
16         return copy;
17     }
18     private void copyOnWrite() {
19         if (theHashTable.getProxyCount() > 1) {
20             synchronized (theHashTable) {
21                 theHashTable.removeProxy();
22                 theHashTable =(ReferenceCountedHashTable) theHashTable.clone();
23             }
24         }
25     }
26     ...
27 }
```

Exemple 3 : Proxy virtuel

Situation initiale

- Un programme instancie un certain nombre d'objets lors de son démarrage :



- Ces objets représentent des fonctionnalités qui ne sont pas nécessairement utiles par la suite,
- cependant le coût de cette solution au départ semble acceptable.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write

P. Virtuel

Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

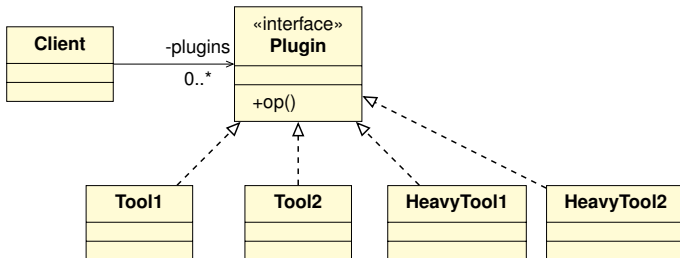
Médiateur

Motivation

Exemple 3 : Proxy virtuel

Et le prévisible arriva ...

- Cependant, l'application reçoit au fil du temps de nombreuses contributions,
- dont des modules assez lourds devenus standards :



- Conséquence : l'application devient très longue à démarrer, même pour les utilisateurs souhaitant en faire l'usage le plus basique.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 3 : Proxy virtuel

Situation initiale

- Bien-sur, cette situation aurait pu être évitée en anticipant le problème et en concevant une architecture adaptée, utilisant par exemple une initialisation paresseuse.
- Mais ce n'est pas le cas et nous devons faire avec l'existant :
- L'architecture impose que l'application, et tous ses modules, considère que les modules sont disponibles à tout moment après l'initialisation :

```
1 void initApp() {...  
2     Map<Plugin> plugins=Plugin.load("dossier/plugins"); // Builder  
3     ...  
4 }  
5  
6 ...  
7 Y calcul(X x) {  
8     return plugin.get("calcul").op(x); // Utilisation du plugin "calcul".  
9 }
```

Exemple 3 : Proxy virtuel

Solutions ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

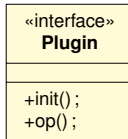
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Modifier l'application pour initialiser les objets de manière paresseuse : il faut alors déplacer du code du constructeur vers une méthode d'initialisation qui elle ne sera pas appelée systématiquement :



- ... mais nous changeons un contrat déjà établi publiquement !
- et le client doit être modifié pour prendre en charge le mécanisme paresseux !

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

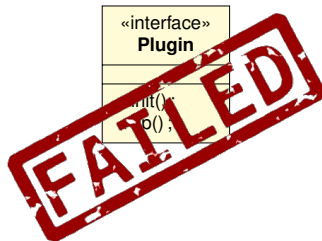
Médiateur

Motivation

Exemple 3 : Proxy virtuel

Solutions ?

- Modifier l'application pour initialiser les objets de manière paresseuse : il faut alors déplacer du code du constructeur vers une méthode d'initialisation qui elle ne sera pas appelée systématiquement :



- ... mais nous changeons un contrat déjà établi publiquement !
- et le client doit être modifié pour prendre en charge le mécanisme paresseux !

Exemple 3 : Proxy virtuel

Solutions ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write

P. Virtuel

Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Spécifier dans le manuel de programmation des greffons que l'initialisation doit se faire de manière paresseuse ?
- Le comportement dépendra de la bonne volonté des développeurs de greffons.
- impossible de garantir ce comportement pour les greffons qui ne sont pas empaquetés avec l'application (relecture de code).

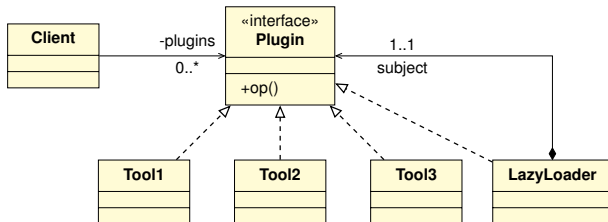
La version *proxy virtuel* du patron proxy nous permet d'ajouter de manière uniforme un comportement à des composants sans modifier l'application⁹.

9. À l'exception de l'initialisation bien-sur, c'est-à-dire ici du builder.

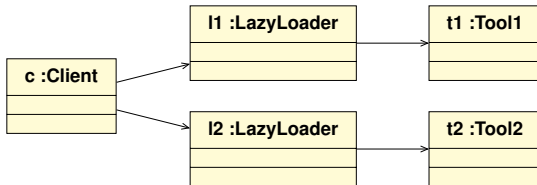
Exemple 3 : Proxy virtuel

Ajout du proxy dans l'infrastructure

- L'ajout n'induit aucune modification de code :



- Chaque greffon est accédé via son objet proxy :



Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Exemple 3 : Proxy virtuel

Réalisation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Dans le builder, on remplace :

```
1 plugins [ "pluginXName" ] = new PluginXName ( ) ;
```

par

```
1 plugins [ "pluginXName" ] = new LazyLoader ( "pluginXName" ) ;
```

- On ne modifie rien d'autre dans l'application.

Exemple 3 : Proxy virtuel

Réalisation

- Le proxy instancie dynamiquement le greffon à sa première utilisation :

```
1 public class LazyLoader() implements Plugin {
2     private String pluginName;
3     private Plugin plugin;
4
5     public LazyLoader(String PluginName) {
6         this.PluginName=PluginName;
7     }
8
9     private synchronized Plugin getPlugin()
10    throws ClassNotFoundException, NoSuchMethodException, ... {
11        if (plugin==null) {
12            Class c=Class.forName(PluginName);
13            Constructor cons=c.getConstructor(new Class []{});
14            plugin=(Plugin)cons.newInstance(new Object []{});
15        }
16        return plugin;
17    }
18
19    public String operation(String par) {
20        return getPlugin().operation(par);
21    }
22 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

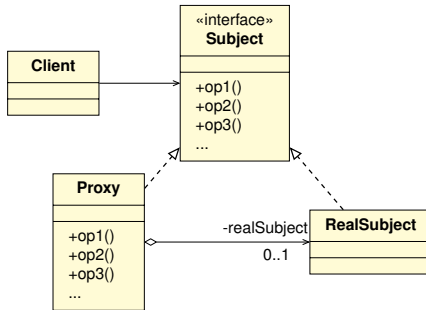
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

Situation initiale



- La délégation (par composition) induit l'écriture exhaustive de toutes les méthodes de l'interface **subject** dans la classe du proxy,
- Le proxy est générique si le type utilisé pour la composition est une interface,

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

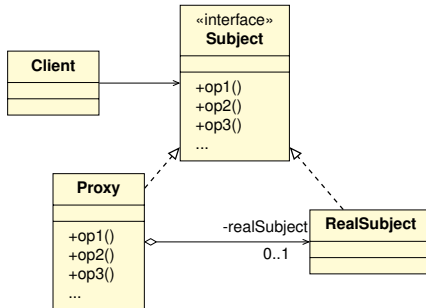
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

Situation initiale



- La délégation (par composition) induit l'écriture exhaustive de toutes les méthodes de l'interface **subject** dans la classe du proxy,
- Le proxy est générique si le type utilisé pour la composition est une interface,
- le proxy peut-il être indépendant de toute interface ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples

P.Synchro
Copy-On-Write
P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

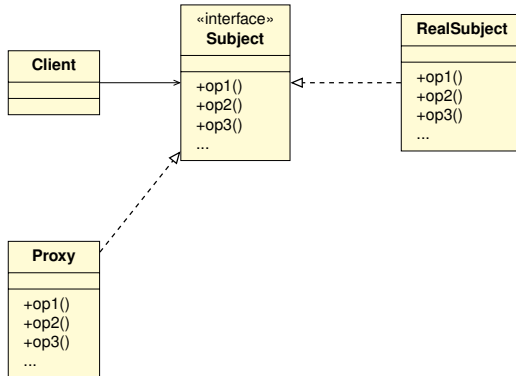
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

principe



- L'API `reflection` de Java génère un proxy implémentant l'interface `Subject`.
- Quel code dans les méthodes ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

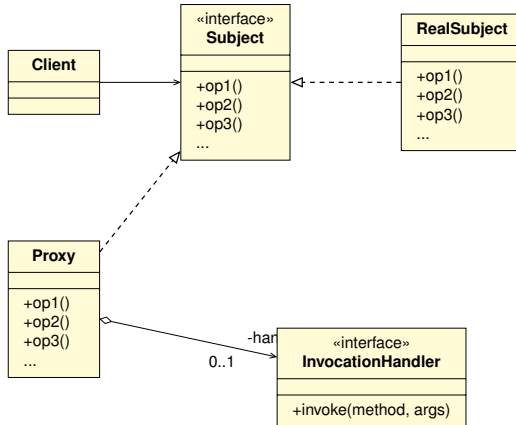
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

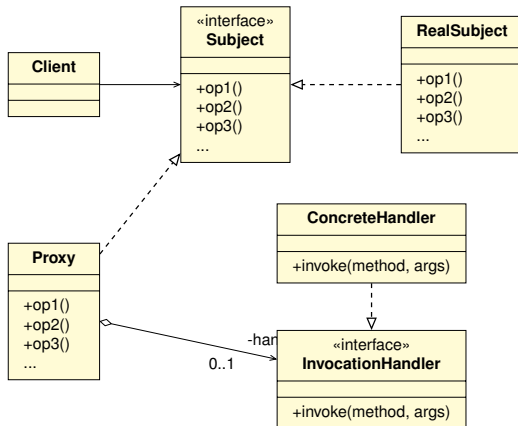
principe



- Le code généré est générique et appelle la méthode `invoke` d'un gestionnaire d'invocation associé au proxy.

Proxy Dynamique

principe



- **ConcreteHandler** est la classe que nous devons écrire.
- Mais comment appeler les méthodes du sujet réel ?

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

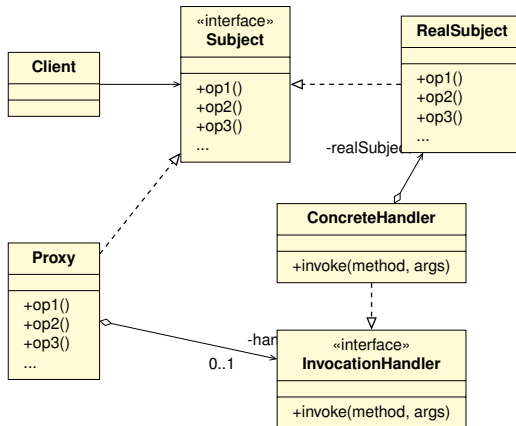
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

principe



- Composer avec le sujet réel ? Pas générique du tout !

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchron
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

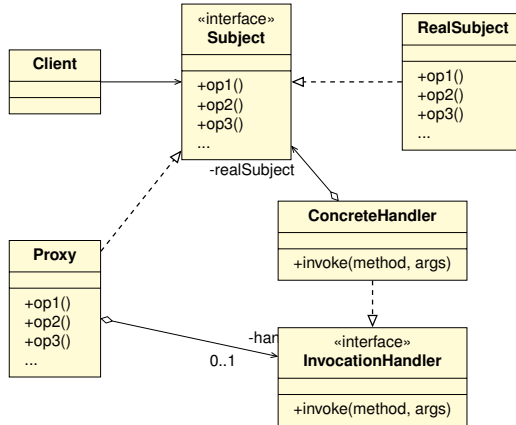
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

principe



- Composer avec le sujet ? Mieux, mais nous aurions pu nous en sortir avec un simple paramètre générique T !

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

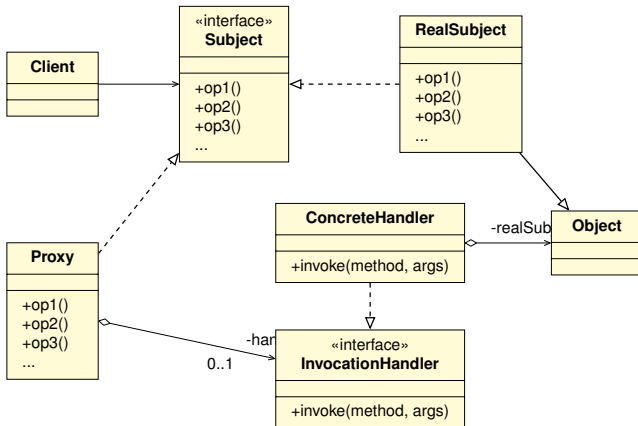
Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

principe



- Il ne nous reste que le type `Object`.
- L'appel se fera à travers un objet de la classe `Method...`

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Proxy Dynamique

Éléments de l'API Reflection

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchron
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Les classes proxy sont créées grâce à la classe `java.lang.reflect.Proxy`.
- Les classes proxy sont des classes concrètes publiques et finales, dérivées de `java.lang.reflect.Proxy`.
- Le nom d'une classe proxy n'est pas précisé. Toutefois les noms commençant par "`$Proxy`" sont réservés.
- Une classe proxy réalise exactement les interfaces spécifiées lors de sa création.

Proxy Dynamique

Éléments de l'API Reflection

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Chaque classe proxy possède un constructeur public qui admet pour argument une réalisation de l'interface `InvocationHandler`.
- S'il est possible d'utiliser l'API `reflection` pour accéder à ce constructeur, il est plus simple d'utiliser la méthode statique `Proxy.newInstance()` qui combine la création dynamique de la classe et de l'instance .

Proxy Dynamique

Éléments de l'API Reflection

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Quelques méthodes de la classe

`java.lang.reflect.Proxy :`

- Création d'une classe proxy :

```
1 public static Class getProxyClass(ClassLoader loader, Class[] interfaces  
2     )  
    throws IllegalArgumentException
```

- Constructeur de la classe générée :

```
1 protected Proxy(InvocationHandler ih)
```

- Tester si une classe est un proxy :

```
1 public static boolean isProxyClass(Class c)
```


Proxy Dynamique

Éléments de l'API Reflection

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel

Proxy Dynamique

Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Raccourci pour la création d'un objet proxy :

- La méthode :

```
1 public static Object newProxyInstance(ClassLoader loader ,  
2     Class[] interfaces ,  
3     InvocationHandler ih)  
4     throws IllegalArgumentException
```

- Construit une classe proxy et retourne une instance.
- `Proxy.newProxyInstance(cl, interfaces, ih) ;` est équivalent à :

```
1 Proxy.getProxyClass(cl , interfaces).getConstructor(  
2     new Class[] { InvocationHandler.class }  
3 ).newInstance(new Object[] { ih });
```

Proxy Dynamique

Éléments de l'API Reflection

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P. Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 package java.lang.reflect ;  
2  
3 public Interface InvocationHandler {  
4     Object invoke(Object proxy ,  
5         Method method ,  
6         Object [] args  
7     ) throws Throwable  
8 }
```

- **proxy** : une référence à l'objet proxy responsable de l'invocation.
- **method** : la classe `Method` décrit une méthode avec une signature donnée et permet d'effectuer un appel.
- **args** : un tableau d'objets représentant les paramètres effectifs.

Exemple Proxy Dynamique

Situation initiale

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- Dans le cadre de profilage de programmes on souhaite pouvoir mesurer les temps d'exécution des méthodes de certaines classes pour détecter où le programme consomme le plus de temps.
- On ne souhaite évidemment pas modifier le code des classes existantes,
- On souhaite pouvoir réutiliser notre instrument de mesure pour toute classe existante ou à venir.
- On se propose de créer un proxy dynamique capable de s'intercaler entre un client et une classe quelconque.

Exemple Proxy Dynamique

Réalisation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

```
1 package proxy.dynamic;
2 import java.lang.reflect.*; // {InvocationHandler, Method, Proxy}
3
4 public class ImpatientHandler implements InvocationHandler {
5     private Object target;
6     private ImpatientHandler(Object target) {
7         this.target = target;
8     }
9
10    public static Object newInstance(Object target) {
11        ClassLoader loader = target.getClass().getClassLoader();
12        Class[] interfaces = target.getClass().getInterfaces();
13        return Proxy.newProxyInstance(loader, interfaces, new Impatient
14            Handler (target));
15    }
16
17    public Object invoke(Object proxy, Method m, Object[] args) throws
18        Throwable {
19        Object result;
20        long t1 = System.currentTimeMillis();
21        result = m.invoke(target, args);
22        long t2 = System.currentTimeMillis();
23        if (t2 - t1 > 10) {
24            System.out.println(">It takes " + (t2 - t1) + " millis to invoke "
25                + m.getName()+"() with");
26            for (int i = 0; i < args.length; i++)
27                System.out.println(">arg[" + i + "] : " + args[i]);
28        }
29        return result;
30    }
31 }
```

Exemple Proxy Dynamique

Réalisation

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Utilisation :

```
1 package proxy.dynamic ;
2 import java.util.HashSet;
3 import java.util.Set;
4
5 public class TestDynamicProxy {
6     public static void main(String[] args) {
7         // Le sujet :
8         Set<Apple> set = new HashSet<Apple>();
9         // Le proxy+gest. d'invocation :
10        Set<Apple> proxy = (Set<Apple>) ImpatientHandler.newInstance(set);
11
12        proxy.add(new GoodApple("Cox_Orange"));
13        proxy.add(new BadApple("Lemon"));
14        proxy.add(new GoodApple("Pears"));
15        System.out.println("The_set_contains_ " + set.size() + "_things.");
16    }
17 }
```

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Conclusion

- Les implémentations du pattern PROXY produisent un objet intermédiaire qui gère l'accès à un objet cible.
- Un objet proxy peut dissimuler aux clients les changements d'état d'un objet cible, comme dans le cas d'une image qui nécessite un certain temps pour se charger.
- Le problème est que ce pattern s'appuie habituellement sur un couplage étroit entre l'intermédiaire et l'objet cible.
- Dans certains cas, la solution consiste à utiliser un proxy dynamique : lorsque la classe d'un objet implémente des interfaces pour les méthodes que vous voulez intercepter, vous pouvez envelopper l'objet dans un proxy dynamique et faire en sorte que votre code s'exécute avant/après le code de l'objet enveloppé ou à sa place.

Principes respectés

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- **S.R.P. ; ?**
- **O.C.P. : ?**
- **L.S.P. : ?**
- **I.S.P. : ?**
- **D.I.P. : ?**

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le patron Médiateur

Encapsule la façon dont un groupe d'objets coopèrent en leur évitant de se connaître.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



Switch ...

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



Switch ...

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



Switch ...

Principes respectés

XXX

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

- **O.C.P.** : le code réutilisé n'est pas modifié, les bibliothèques sont conservées intactes et continueront de fonctionner avec fiabilité avec le reste du système.
- **D.I.P.** : `Target` est une abstraction issue de l'univers « métier » du client, `Adaptee` une abstraction de bas niveau fournie par la boîte à outils. Conserver ces deux interfaces intactes contribue à respecter le **D.I.P.**.
- **I.S.P.** : en conservant les interfaces `Target` et `Adaptee` séparées, on évite la pollution d'interface et l'on respecte l'**I.S.P.**.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le patron Médiateur

Encapsule la façon dont un groupe d'objets coopèrent en leur évitant de se connaître.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



Poursuivre avec ...

4-mediator-chain-responsabilité.pdf

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Le patron Chaîne de responsabilité

Découple l'émetteur d'une requête du récepteur et permet à plus d'un objet de participer à son traitement.

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation



Poursuivre avec ...

4-mediator-chain-responsabilité.pdf

F. Nicart

Singleton

Motivation
Structure
Conclusion

Observateur

Motivation
Structure
Exemples
Conclusion

Proxy

Motivation
Structure
Exemples
P.Synchro
Copy-On-Write
P. Virtuel
Proxy Dynamique
Conclusion

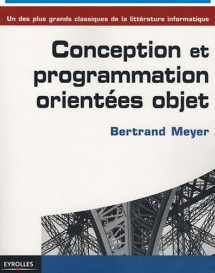
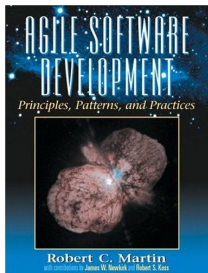
Médiateur

Motivation
Structure
Exemples
Conclusion

Médiateur

Motivation

Quelques références



Agile Software Development : Principles, Patterns, and Practices.,
Robert C. Martin, Prentice Hall (2002).

ISBN-13 : 978-0135974445.

Conception et programmation orientées objets,
Bertand Meyer, Eyrolles (3 janvier 2008).
ISBN-13 : 978-2212122701.