Design Patterns
Part 6

Mohamed Youssfi

Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)

ENSET, Université Hassan II Casablanca, Maroc

Email : med@youssfi.net

Supports de cours : http://fr.slideshare.net/mohamedyoussfi9

Chaîne vidéo : http://youtube.com/mohamedYoussfi

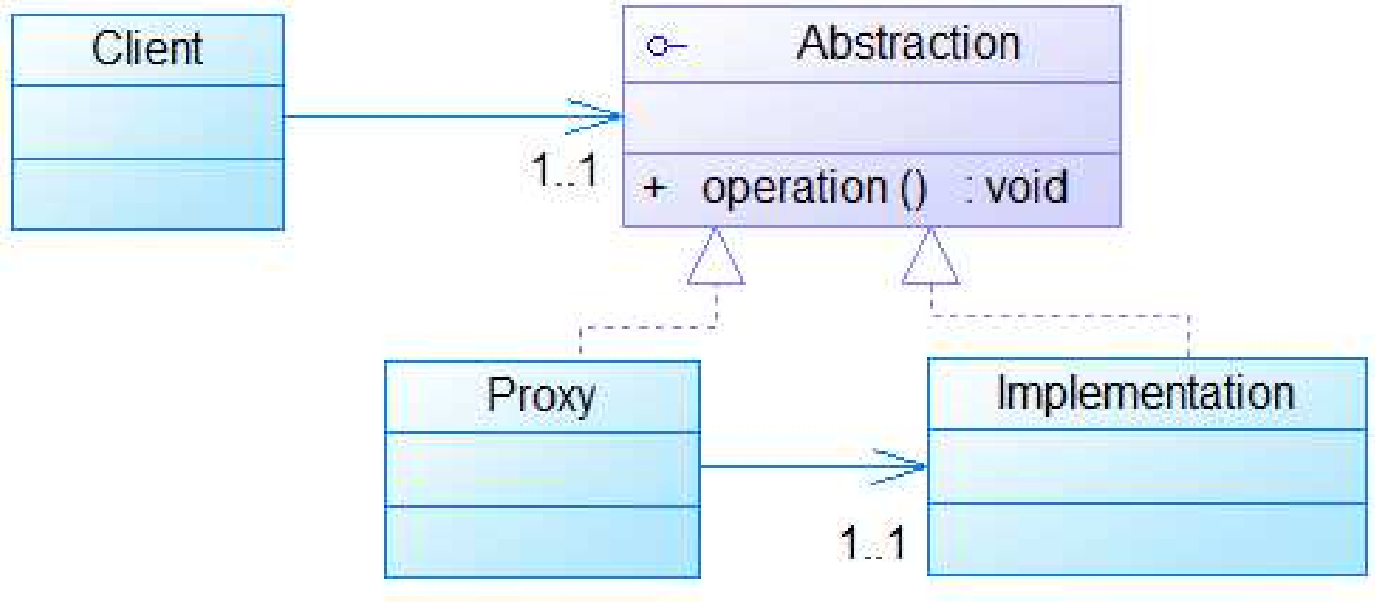
Recherche : http://www.researchgate.net/profile/Youssfi_Mohamed/publications

Pattern Proxy

Pattern Proxy

- Catégorie :
  - Structure
- Objectif du pattern
  - *Fournir un intermédiaire entre la partie cliente et un objet pour contrôler les accès à ce dernier.*
- Résultat :
  - Le Design Pattern permet d'isoler le comportement lors de l'accès à un objet.

Diagramme de classes du pattern Proxy



Raison d'utilisation

- Les opérations d'un objet sont coûteuses en temps ou sont soumises à une gestion de droits d'accès.
- Il est nécessaire de contrôler l'accès à un objet.
- Cela peut être un système de chargement d'un document. Le document est très lourd à charger en mémoire ou il faut certaines habilitations pour accéder à ce document.
- L'objet réel (système de chargement classique) est l'implémentation. L'intermédiaire entre l'implémentation et la partie cliente est le proxy.
- Le proxy fournit la même interface que l'implémentation. Mais il ne charge le document qu'en cas de réel besoin (pour l'affichage par exemple) ou n'autorise l'accès que si les conditions sont satisfaites.

Responsabilités

- **Abstraction** : définit l'interface des classes Implémentation et Proxy.
- **Implémentation** : implémente l'interface. Cette classe définit l'objet que l'objet Proxy représente.
- **Proxy** : fournit un intermédiaire entre la partie cliente et l'objet Implémentation. Cet intermédiaire peut avoir plusieurs buts (synchronisation, contrôle d'accès, cache, accès distant, ...). Dans l'exemple, la classe Proxy n'instancie un objet Implémentation qu'en cas de besoin pour appeler la méthode correspondante de la classe Implémentation.
- La **partie cliente** appelle la méthode operation() de l'objet Proxy.

Implémentation

/* Abstraction.java */

```
public interface Abstraction {
   public void operation();
}
```

/* Implemantation.java */

```
public class Implementation implements Abstraction {
 @Override
 public void operation() {
  System.out.println("Exécution de l'opération de
l'implémentation...");
}}
```

/* Proxy.java */

```
public class Proxy implements Abstraction {
  private Implementation implementation;
  @Override
  public void operation() {
        System.out.println("Vérification des conditions d'accès par le
proxy");
        implementation=new Implementation();
        implementation.operation();
}}
```

Implémentation

/* Application.java */

```
public class Client {
 public static void main(String[] args) {
   Abstraction proxy=new Proxy();
   proxy.operation();
 }}
```

```
Vérification des conditions d'accès par le proxy
Exécution de l'opération de l'implémentation...
```