Design Patterns
Part 5

Mohamed Youssfi

Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)

ENSET, Université Hassan II Casablanca, Maroc

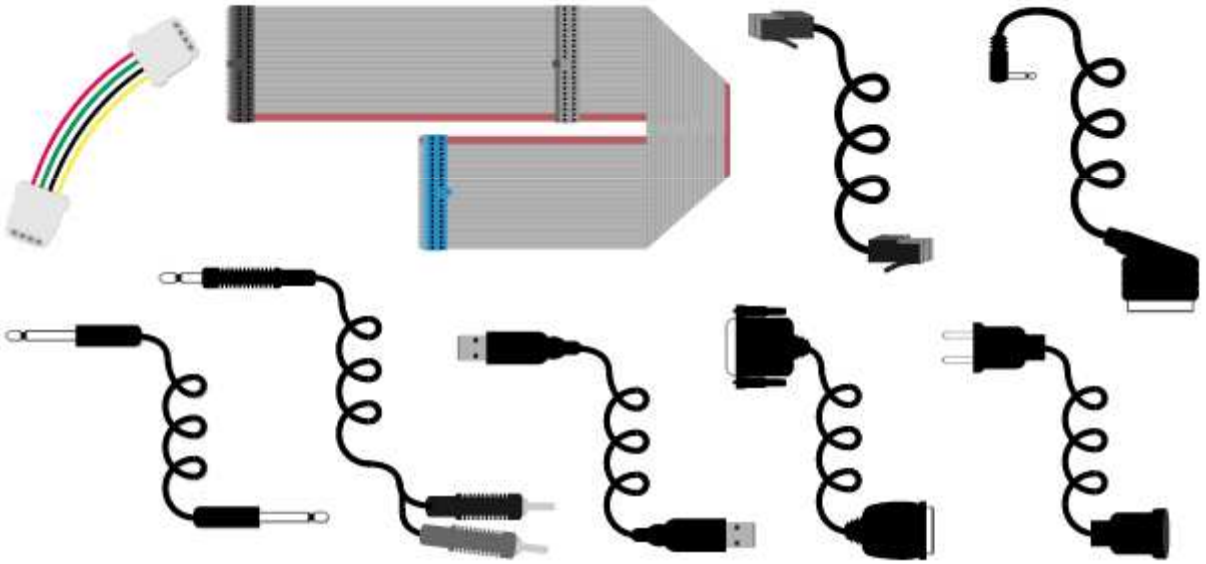
Email : med@youssfi.net

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

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

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

Pattern Adapter

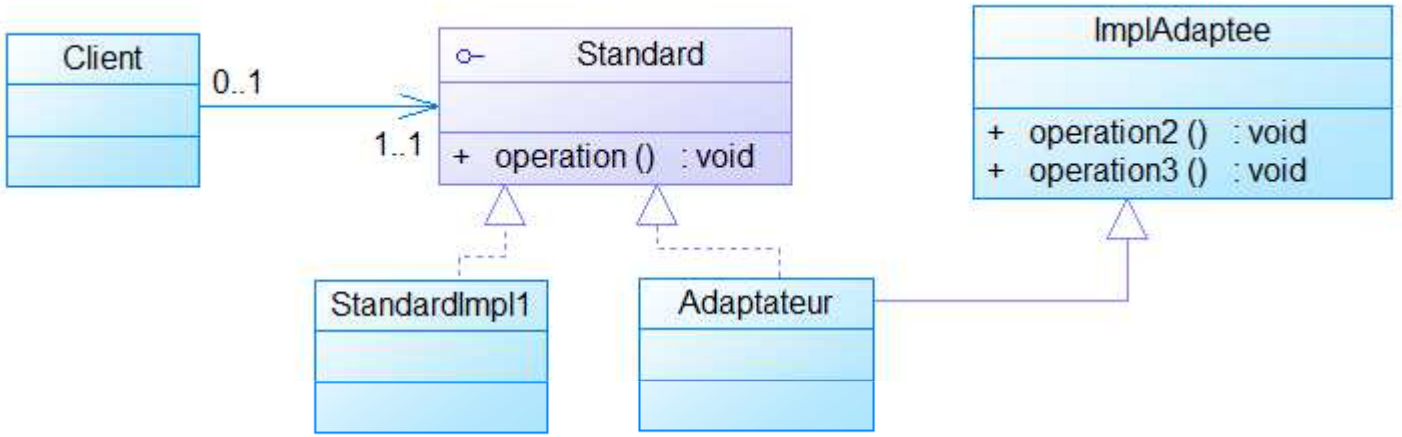


Pattern Adapter

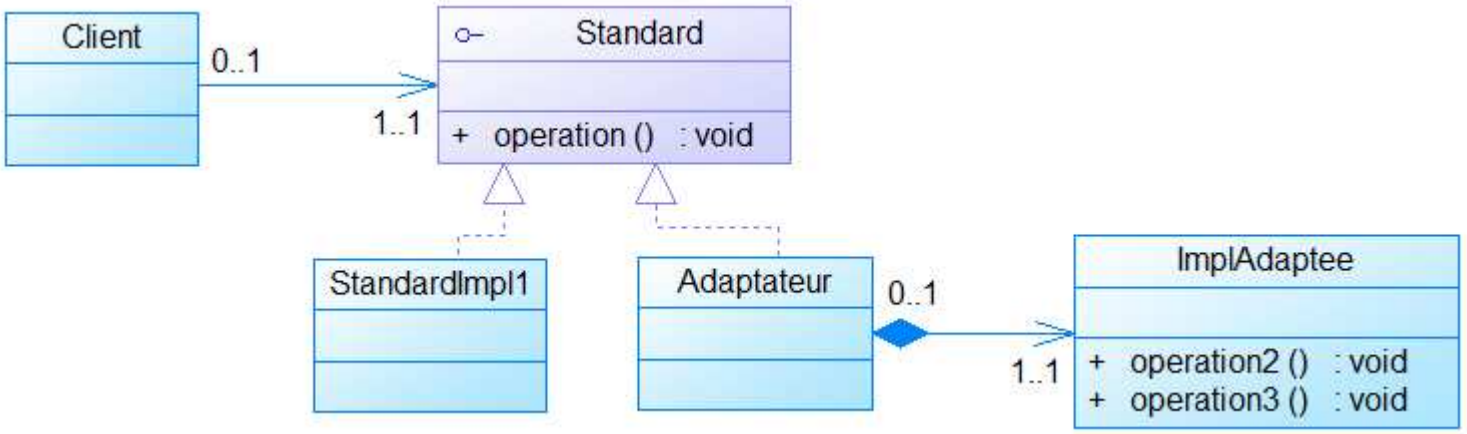
- Catégorie :
  - Structure
- Objectif du pattern
  - *Convertir l'interface d'une classe dans une autre interface comprise par la partie cliente.*
  - *Permettre à des classes de fonctionner ensemble, ce qui n'aurait pas été possible à cause de leurs interfaces incompatibles.*
- Résultat :
  - Le Design Pattern permet d'isoler l'adaptation d'un sous-système.

Diagramme de classes

Pattern Adapter par héritage



Pattern Adapter par Composition



Raison d'utilisation

- Le système doit intégrer un sous-système existant.
- Ce sous-système a une interface non standard par rapport au système.
- Cela peut être le cas d'un driver bas niveau pour de l'informatique embarquée.
- Le driver fournit par le fabricant ne correspond pas à l'interface utilisée par le système pour d'autres drivers.
- La solution est de masquer cette interface non stantard au système et de lui présenter une interface standard.
- La partie cliente utilise les méthodes de l'Adaptateur qui utilise les méthodes du sous-système pour réaliser les opérations correspondantes.

Responsabilités

- **Standard** : définit une interface qui est identifiée comme standard dans la partie cliente.
- **ImplStandard** : implémente l'interface Standard. Cette classe n'a pas besoin d'être adaptée.
- **ImplAdaptee** : permet de réaliser les fonctionnalités définies dans l'interface Standard, mais ne la respecte pas. Cette classe a besoin d'être adaptée.
- **Adaptateur** : adapte l'implémentation ImplAdaptee à l'interface Standard. Pour réaliser l'adaptation, l'Adaptateur peut utiliser une ou plusieurs méthodes différentes de l'implémentation ImplAdaptee pour réaliser l'implémentation de chaque méthode de l'interface Standard.
- **La partie cliente** : manipule des objets Standard. donc, l'adaptation est transparente pour la partie cliente.

Implémentation

/* Standard.java */

```
public interface Standard {
   public void operation(int nb1,int nb2);
}
```

/* ImplStandard.java */

```
public class ImplStandard implements Standard {
 @Override
 public void operation(int nb1, int nb2) {
  System.out.println("Standard, Résultat est :"+nb1*nb2);
 }
}
```

/* ImplStandard.java */

```
public class ImplAdaptee {
    public int operation2(int nb1,int nb2){
     return nb1*nb2;
    }
    public void operation3(int nb){
     System.out.println("Adaptée, Résultat="+nb);
    }
}
```

Implémentation

/* AdaptateurHeritage.java */

```
public class AdaptateurHeritage extends ImplAdaptee implements Standard {
   @Override
   public void operation(int nb1, int nb2) {
        int nb=operation2(nb1, nb2);
        operation3(nb);
   }
}
```

/* AdaptateurComposition.java */

```
public class AdaptateurComposition implements Standard {
  private ImplAdaptee adaptee=new ImplAdaptee();
  @Override
  public void operation(int nb1, int nb2) {
        int nb=adaptee.operation2(nb1, nb2);
        adaptee.operation3(nb);
  }
}
```

Implémentation

/* Application.java */

```
public class Application {
public static void main(String[] args) {
 Standard standard=new ImplStandard();
 standard.operation(7, 9);
 Standard adaptee1=new AdaptateurHeritage();
 adaptee1.operation(7, 9);
 Standard adaptee2=new AdaptateurComposition();
 adaptee2.operation(7, 9);
}
}
```

```
Standard, Résultat est :63
Adaptée, Résultat=63
Adaptée, Résultat=63
```