Design pattern Adapter

ANDRE Manon
DE BREM Lucas
PERRUCHAUD Lucie

Plan

I. Les designs patterns
II. Le design pattern Adapter
III. Exemples
IV. Le QCM

Les Design Patterns

Introduction:

- Pattern:
  - élément, événement qui se répète

- Design Pattern:
  - Bonne pratique de développement
  - 1970, Christopher Alexander
    - *A Pattern Language*
    - Rassemble Patterns architecturaux
  - 1995, Gang of Four
    - *Design Patterns - Elements of Reusable Object-Oriented Software*
    - Rassemble 23 Design Patterns

Les Design Patterns

Définition:

- 3 familles: Création, Structure, Comportement
- 4 caractéristiques par Design Pattern:
  - Nom
  - Problématique
  - Solution
  - Conséquences

Les Design Patterns

Single Responsabity Principle

Open/Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

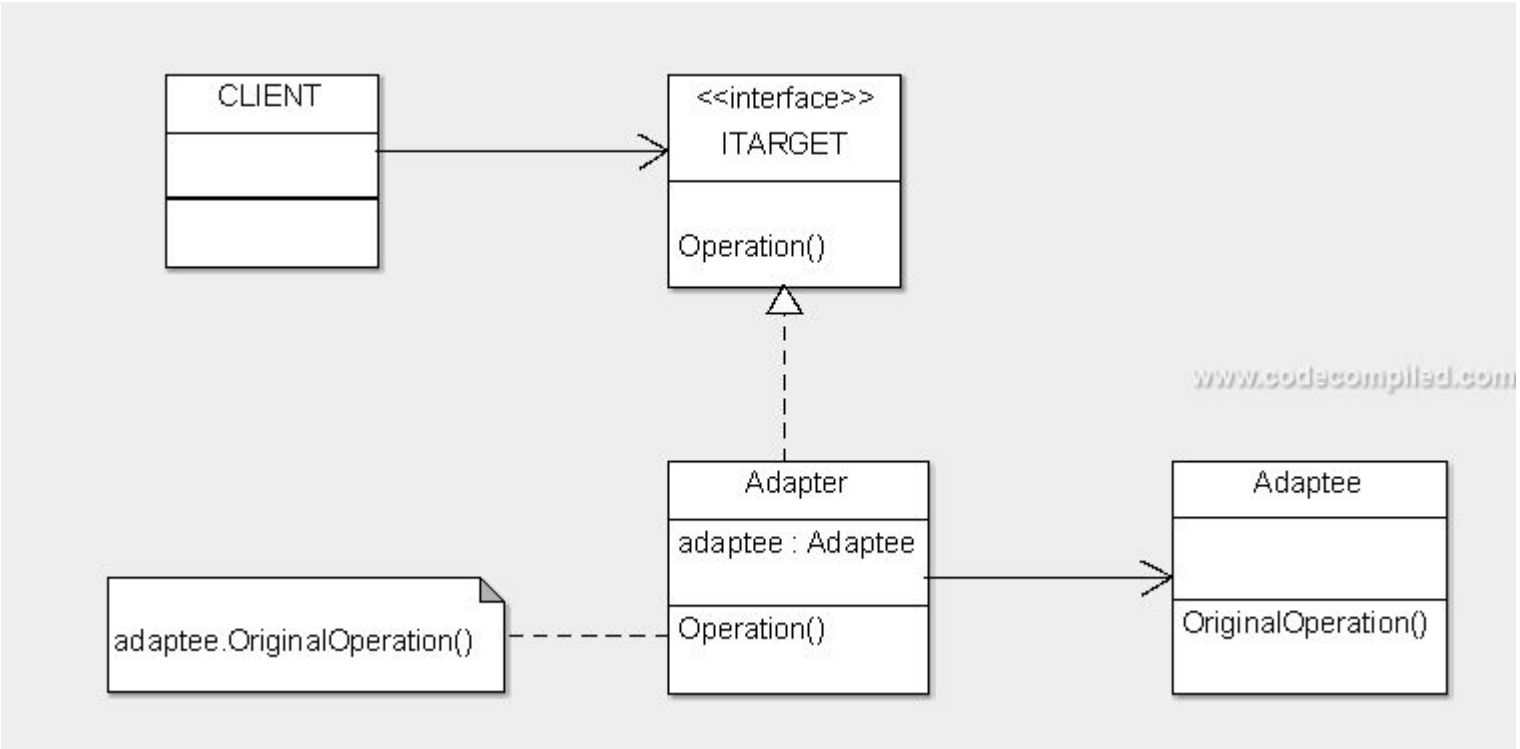
Le design pattern Adapter

- Structural pattern
- Les caractéristiques:
  - Adapter
  - Ajuster l’interface d’un objet à celle attendue par le code client
  - Conserver une instance de la classe adaptee et convertir les appels d’une interface existante vers l’interface implémentée
  - Permettre à des classes de collaborer

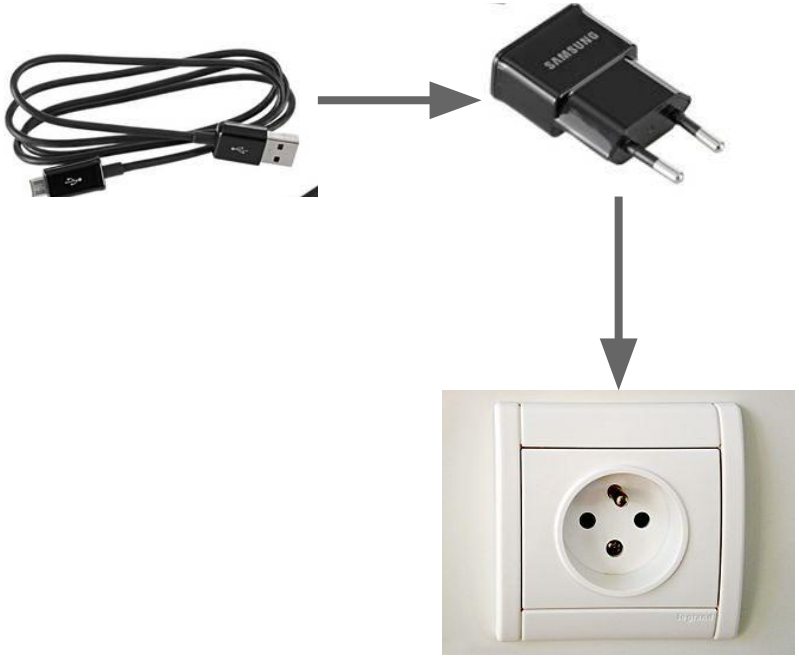
Le design pattern Adapter

- Avantages :
  - moins de méthodes vide
  - moins de méthodes contenant du code répétitif
- Inconvénients:
  - une classe ne peut hériter que d’un adapter à la fois

Le design pattern Adapter

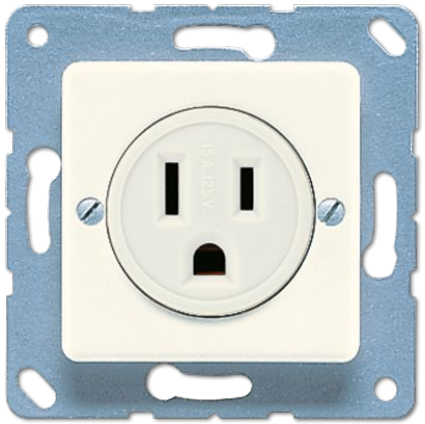


Le design pattern Adapter

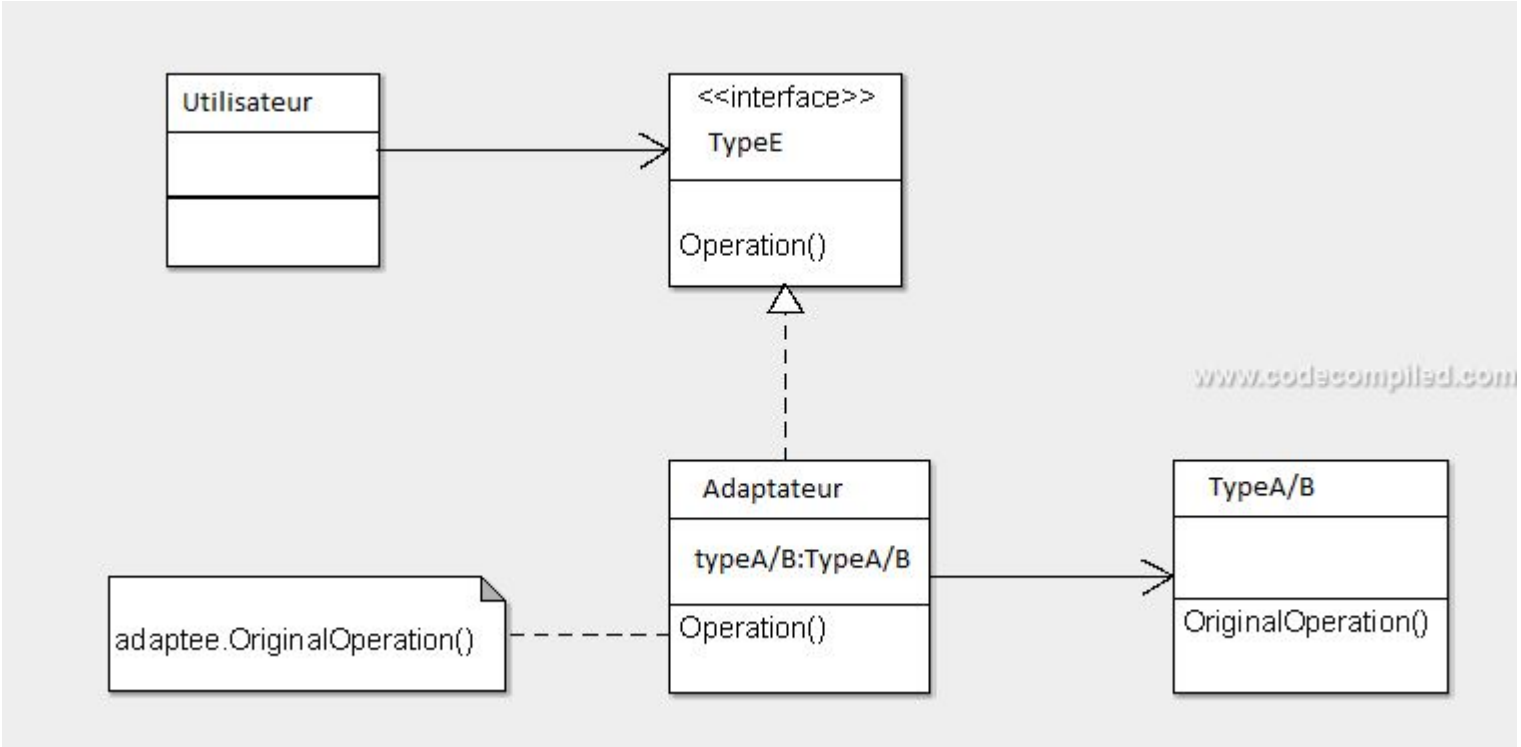


Le design pattern Adapter

Le design pattern Adapter



Le design pattern Adapter

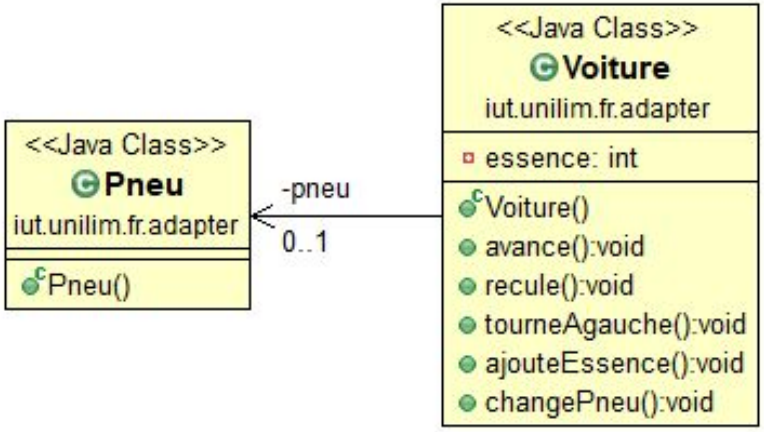


Le design pattern Adapter

Principe SOLID:

- LSP

Exemples


<<Java Class>>
Pneu
iut.unilim.fr.adapter
Pneu()
-pneu
0..1
<<Java Class>>
Voiture
iut.unilim.fr.adapter
essence: int
Voiture()
avance():void
recule():void
tourneAgauche():void
ajouteEssence():void
changePneu():void

Exemples

```java
public class Voiture {
    private int essence;
    private Pneu pneu;

    public void avance() {
        this.essence = this.essence-1;
        System.out.println("la voiture avance");
    }

    public void recule() {
        this.essence = this.essence-1;
        System.out.println("la voiture recule");
    }

    public void tourneAgauche() {
        this.essence = this.essence-1;
        System.out.println("la voiture tourne a gauche");
    }

    public void ajouteEssence() {
        this.essence = 100;
        System.out.println("le plein a été fait");
    }

    public void changePneu() {
        this.pneu = new Pneu();
        System.out.println("le pneu a été changé");
    }


}
```

```java
public class Pneu {
    public Pneu() {
    }
}
```

Exemples

```
public class VoitureVolante extends Voiture {
    private TrainAterissage trainAterissage;

    public void changePneu(){
    //Solution1: Ne rien faire

    //Solution2: faire Autre chose qui ressemble
    this.changeTrainAterissage();

    //Solution3: Lancer une exception
    throw new PasDePneuException();
    }

//Dois-je mettre ceci en public étant donné qu'on y accède via "changePneu" ?
private void changeTrainAterissage()
{
    this.trainAterissage = new TrainAterissage();
    System.out.println("TrainAterissage changé");
}

}
```

Exemples

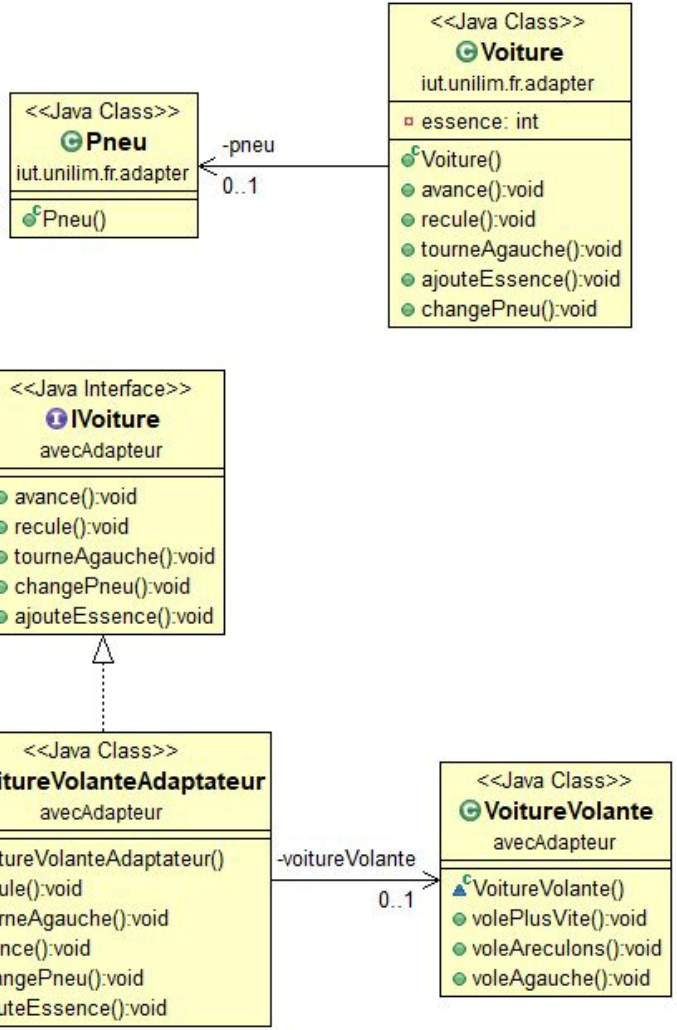
```java
interface IVoiture
{
    public void avance();
    public void recule();
    public void tourneAgauche();
    public void changePneu();
    public void ajouteEssence();
}
```

```java
class VoitureVolante
{
    //Similaire à "avance()" pour une voiture normale
    public void volePlusVite()
    {
        System.out.println("La voiture volante avance/vole plus vite!");
    }

    public void voleAreculons() {
        System.out.println("La voiture volante recule.");
    }

    public void voleAgauche() {
        System.out.println("La voiture volante tourne a gauche");
    }
}
```

Exemples

```
class VoitureVolanteAdaptateur implements IVoiture
{
    private VoitureVolante voitureVolante;

    public VoitureVolanteAdaptateur() {
        this.voitureVolante = new VoitureVolante();
    }

    public void recule() {
        this.voitureVolante.voleAreculons();
    }

    public void tourneAgauche() {
        this.voitureVolante.voleAgauche();

    }

    public void avance() {
        this.voitureVolante.volePlusVite();
    }

    public void changePneu()
    {
    }

    public void ajouteEssence() {
    }
}
```

Exemples


<<Java Class>>
Pneu
iut.unilim.fr.adapter
Pneu()
-pneu
0..1
<<Java Class>>
Voiture
iut.unilim.fr.adapter
essence: int
Voiture()
avance():void
recule():void
tourneAgauche():void
ajouteEssence():void
changePneu():void
<<Java Interface>>
IVoiture
avecAdapteur
avance():void
recule():void
tourneAgauche():void
changePneu():void
ajouteEssence():void
<<Java Class>>
VoitureVolanteAdaptateur
avecAdapteur
VoitureVolanteAdaptateur()
recule():void
tourneAgauche():void
avance():void
changePneu():void
ajouteEssence():void
-voitureVolante
0..1
<<Java Class>>
VoitureVolante
avecAdapteur
VoitureVolante()
volePlusVite():void
voleAreculons():void
voleAgauche():void

QCM

https://tech.io/playgrounds/f0b0a1a0fe3fdba4f2de4a65e8f1f09684283/java-maven-project