



ENCART CAMÉRA

CC BY NC SA

Patrons de conception
Structurels

Sébastien Mosser - INF5153
Chapitre 7 - Capsule 2
Automne 2020

UQAM | Département d'informatique

ace

ENCART CAMÉRA

CC BY NC SA

Adapter

ENCART CAMÉRA

CC BY NC SA

Problème

Comment **intégrer du code** réalisant une **interface X** alors que c'est **une instance d'un type différent** qui est attendue ?

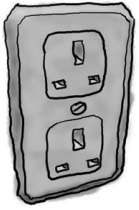
UQAM | Département d'informatique

Exemple

ENCART CAMÉRA



European Wall Outlet



Standard AC Plug



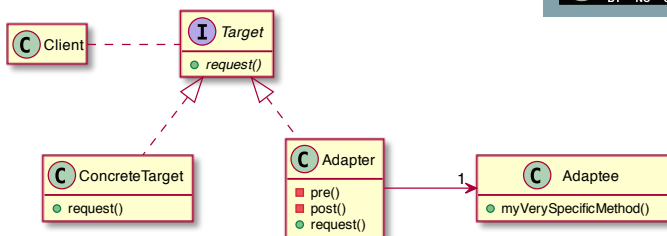
Intention

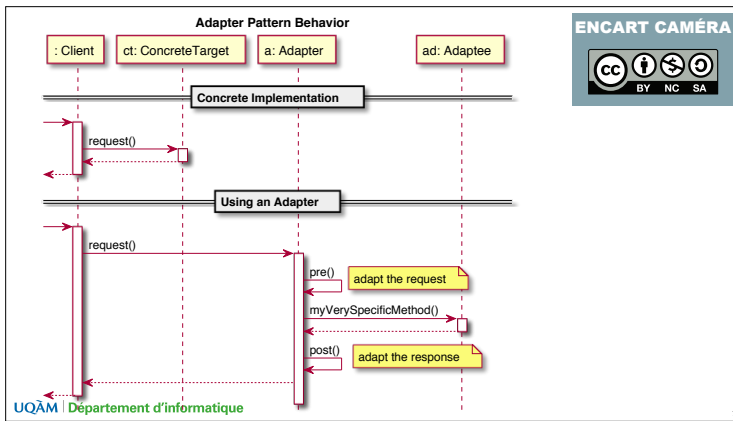
ENCART CAMÉRA



- **Convertir** une interface en une autre
- **Faire collaborer des objets** qui ne pourraient pas en l'état
- Garantir de l'**évolutivité** sur les éléments adaptés

ENCART CAMÉRA





Utilisation dans le code

```
class Line {
    public void draw(int x1, int y1, int x2, int y2) {
        System.out.println("Line from point A(" + x1 + "," +
            y1 + "), to point B(" + x2 + "," + y2 + ")");
    }
}

class Rectangle {
    public void draw(int x, int y, int width, int height) {
        System.out.println("Rectangle with coordinate left-down point (" +
            x + "," + y + "), width: " + width + ", height: " + height);
    }
}

interface Shape {
    void draw(int x, int y, int z, int j);
}
```

UQAM | Département d'informatique

<https://sourcemaking.com/>

8

```
class RectangleAdapter implements Shape {
    private Rectangle adaptee;

    public RectangleAdapter(Rectangle rectangle) {
        this.adaptee = rectangle;
    }

    @Override
    public void draw(int x1, int y1, int x2, int y2) {
        int x = Math.min(x1, x2);
        int y = Math.min(y1, y2);
        int width = Math.abs(x2 - x1);
        int height = Math.abs(y2 - y1);
        adaptee.draw(x, y, width, height);
    }
}
```

UQAM | Département d'informatique

9

Conséquences

ENCART CAMÉRA



- **Intégration aisée** d'interfaces "exotiques"
- Mais tout a un prix :
 - Le **code d'adaptation** est potentiellement "**sale**"
 - **Impossible** d'adapter une **famille de produits** facilement
 - Idem pour une **hiérarchie** de classes

Où le rencontrer dans la "vraie-vie"⁽⁴⁾ ?

ENCART CAMÉRA



- Transformation d'espaces de couleurs
 - RGB, CMYK
- En système distribué, quasiment partout
- Architectures hexagonales

Composite

ENCART CAMÉRA



Problème

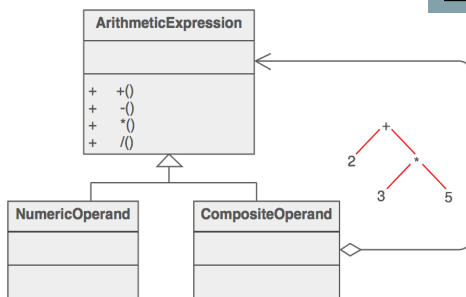
ENCART CAMÉRA



Comment représenter de manière
uniforme une hiérarchie d'objets ?

Exemple

ENCART CAMÉRA

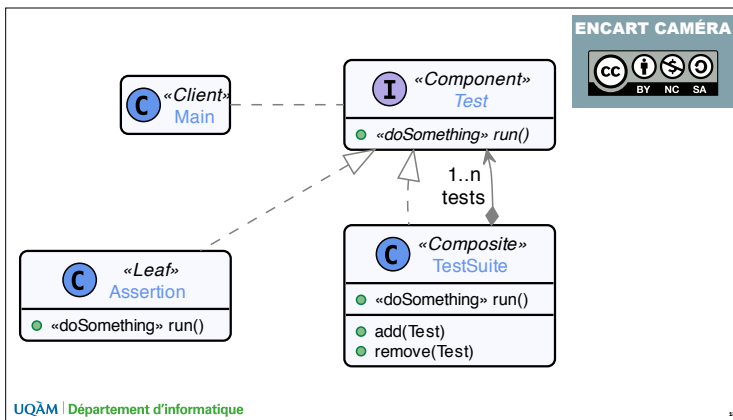
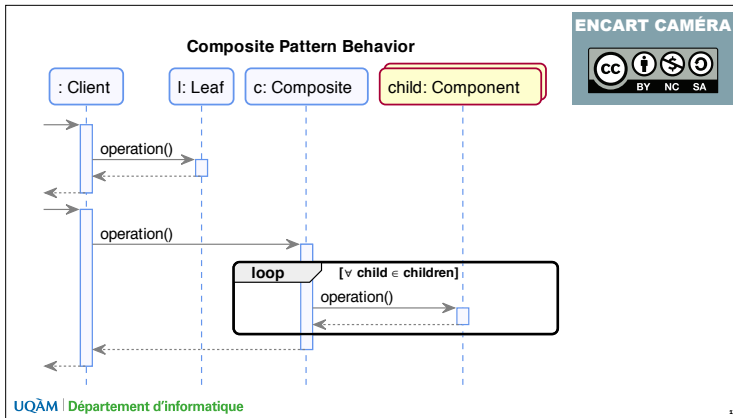
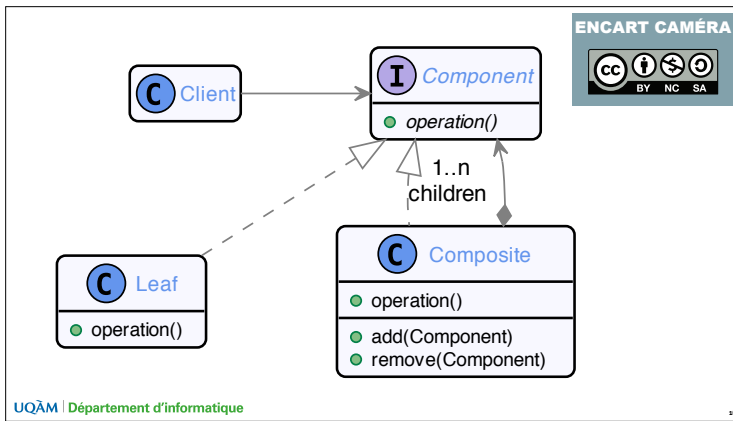


Intention

ENCART CAMÉRA



- Formaliser la structuration arborescente des objets
- Traiter feuilles et noeuds de manière uniforme
 - Propagation récursive des comportements



Utilisation dans le code

ENCART CAMÉRA



```
public static void main(String[] args) {  
  
    TestSuite unit = new TestSuite("Unit tests");  
    unit.add(new Assertion("1st unit test"));  
    unit.add(new Assertion("2nd unit test"));  
    unit.add(new Assertion("3rd unit test"));  
  
    TestSuite accept = new TestSuite("Acceptance tests");  
    accept.add(new Assertion("1st scenario"));  
    accept.add(new Assertion("2nd scenario"));  
  
    TestSuite ext1 = new TestSuite("Integration with EXT-1");  
    ext1.add(new Assertion("Test with external partner #1"));  
    TestSuite ext2 = new TestSuite("Integration with EXT-2");  
    ext2.add(new Assertion("Test with external partner #2"));  
    TestSuite integration = new TestSuite("Integration tests");  
    integration.add(ext2);  
    integration.add(ext1);  
  
    TestSuite complete = new TestSuite("Complete test suite");  
    complete.add(unit);  
    complete.add(accept);  
    complete.add(integration);  
  
    System.out.println("\n# Running a single assertion");  
    (new Assertion("single unit test")).run();  
  
    System.out.println("\n# Running unit tests only");  
    unit.run();  
  
    System.out.println("\n# Running the all product suite");  
    complete.run();  
  
}
```

Conséquences

ENCART CAMÉRA



- Structure **hiérarchique simple et uniforme**
- **Facile à étendre** pour de nouveaux objets
- **Comportement propagable** de manière automatique
- Mais ...
 - potentiellement **trop rigide**
 - Les composants fils sont-ils **ordonnés** ou non ?

Où le rencontrer dans la “vraie-vie”⁽⁴⁾ ?

ENCART CAMÉRA



- Dès qu'ont fait face à une structure arborescente
 - Système de fichiers, GUI, compilation, documents
- **Il se compose particulièrement bien avec la Commande :**
 - Une macro-commande est un composite de commandes



Decorator

22

Problème



- Un objet doit pouvoir **changer dynamiquement de comportement**
- Ces comportements ne sont **pas forcément connus à l'avance**
- L'approche par **héritage** génère une **explosion combinatoire**
- **Les changements sont au niveau de l'objet** et pas de la classe

23

Exemple



- Dans le système d'info de la marque "Van Horton" :
 - Comment calculer le prix d'un café ?
 - D'un café avec lait de soya ?
 - D'un café avec crème fouettée et une dose d'expresso ?
 - D'un café avec ...

24

ENCART CAMÉRA



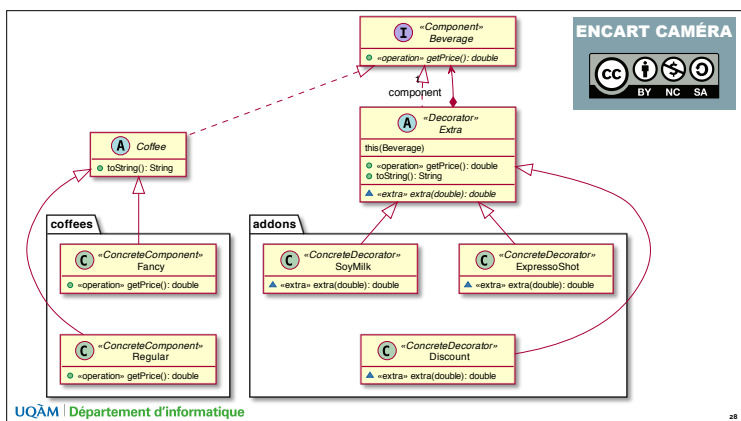
- **Attacher dynamiquement** au niveau de l'objet
- des **capacités additionnelles**
- En lien avec la capacité initialement présente
- Fournir une **alternative plus flexible** à l'héritage



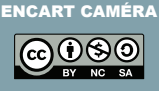
UQÀM | Département d'informatique



UQÀM | Département d'informatique



Utilisation dans le code



```

public static void main(String[] args) {

    System.out.println("# Selling plain coffees");
    displayBeverage(new Regular());
    displayBeverage(new Fancy());


    System.out.println("\n# Adding extras");
    displayBeverage(new SoyMilk(new Regular()));
    displayBeverage(new EspressoShot(new Fancy()));
    displayBeverage(new SoyMilk(new EspressoShot(new Fancy())));
    displayBeverage(new EspressoShot(new EspressoShot(new Regular())));

    System.out.println("\n# Messing up with discounts (aka order matters)");
    displayBeverage(new Discount(new Regular()));
    displayBeverage(new Discount(new EspressoShot(new Regular())));
    displayBeverage(new EspressoShot(new Discount(new Regular())));

}

private static void displayBeverage(Beverage b) {
    System.out.println(b + ": $" + String.format("%.2f", b.getPrice());
}

```



29

```

public abstract class Extra implements Beverage {

    private Beverage component;

    protected Extra(Beverage inner) { this.component = inner; }

    @Override
    public double getPrice() {
        return extra(component.getPrice());
    }

    protected abstract double extra(double price);
}

public class EspressoShot extends Extra {

    public EspressoShot(Beverage inner) { super(inner); }

    @Override
    protected double extra(double price) { return price + 0.80; }
}

public class SoyMilk extends Extra {

    public SoyMilk(Beverage inner) { super(inner); }

    @Override
    protected double extra(double price) { return price + 1.25; }
}

public class Regular extends Coffee {
    @Override
    public double getPrice() { return 2.0; }
}

public class Fancy extends Coffee {
    @Override
    public double getPrice() { return 3.25; }
}

```

Conséquences

ENCART CAMÉRA



- **Plus de flexibilité** qu'avec une approche par héritage
- **Evite la surcharge** initiale de classe
- Le décorateur et son composant sont **deux objets différents**
 - Pose des problèmes pour calculer des **égalités** (typage)
- **Plein de tous petits objets**, beaucoup d'appels inter objets

Où le rencontrer dans la “vraie-vie”⁽⁴⁾ ?

ENCART CAMÉRA



- Système de définition d'Interfaces Graphiques
- Mise en forme de documents
 - CSS en Web, Format de fichier Word / LibreOffice
- L'API de gestion des I/O dans Java
 - `BufferedReader br = new BufferedReader(new FileReader(FILENAME))`

Facade

ENCART CAMÉRA



Problème

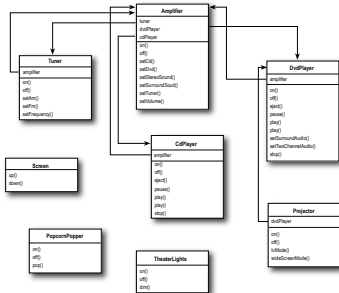
ENCART CAMÉRA



- L'application développée possède
 - un **ensemble d'interfaces complexes**,
 - qu'il faut **coordonner**
- Chaque client doit se **coupler avec des sous-éléments** du système
- Le sous-système peut difficilement **évoluer**

Exemple

ENCART CAMÉRA



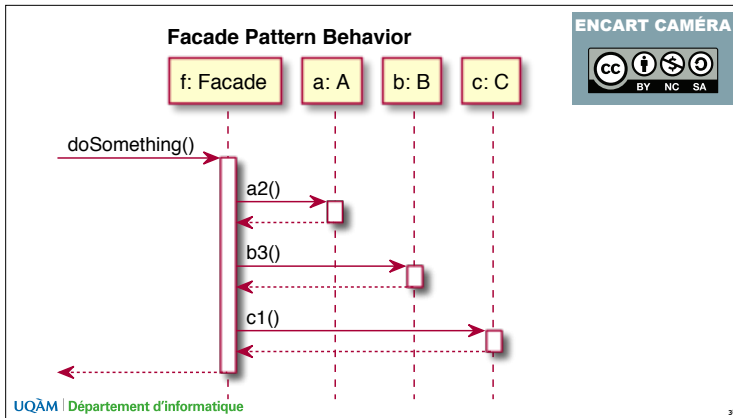
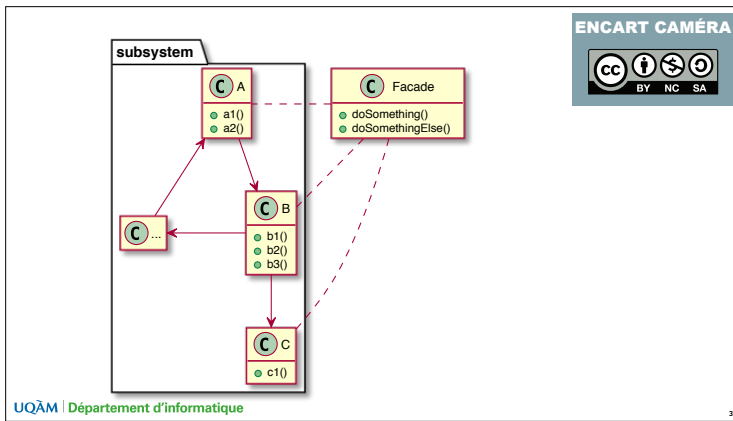
Systeme
HiFi

Intention

ENCART CAMÉRA



- Permettre de **continuer à faire évoluer le sous-système**
- **Découpler** client et sous-système
 - pour ne pas avoir de **dépendances** fortes
- Fournir une **interface unique et simplifiée**
 - qui servira à **isoler** le client du système



Conséquences

UQÀM | Département d'informatique

ENCART CAMÉRA

- Facilite l'utilisation du système
- Permet la diminution du couplage
- Plus flexible que de la visibilité (on peut outrepasser la façade)
- Permet de modifier le système sans modifier le client
- L'interface Facade peut-être trop restrictive

39

Où le rencontrer dans la “vraie-vie”^(c) ?

ENCART CAMÉRA



- Classique en application réparties
 - Interfaces de services / micro-services
 - Mot clé : “API Gateway”
- Développement d'application par composants
 - Promotion / exposition d'interfaces

Proxy

ENCART CAMÉRA



Problème

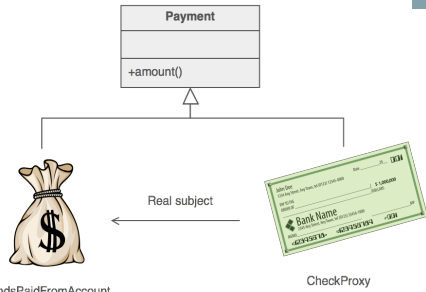
ENCART CAMÉRA



Comment **fournir** un **substitut**
à un objet qui n'est **pas**
accessible facilement ?

Exemple

ENCART CAMÉRA



FundsPaidFromAccount

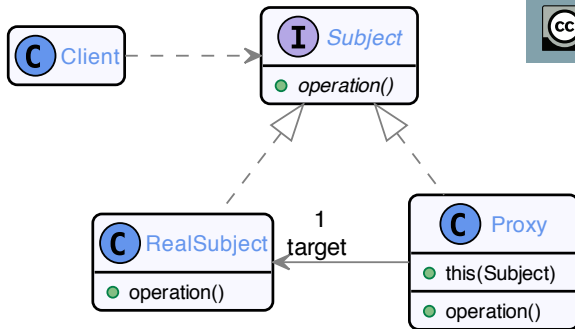
CheckProxy

Intention

ENCART CAMÉRA

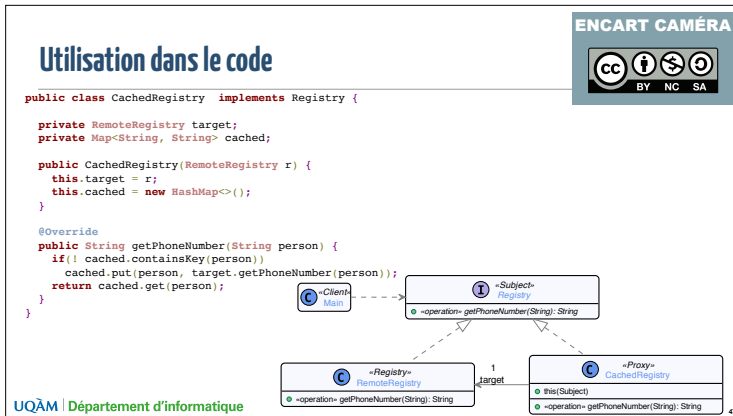
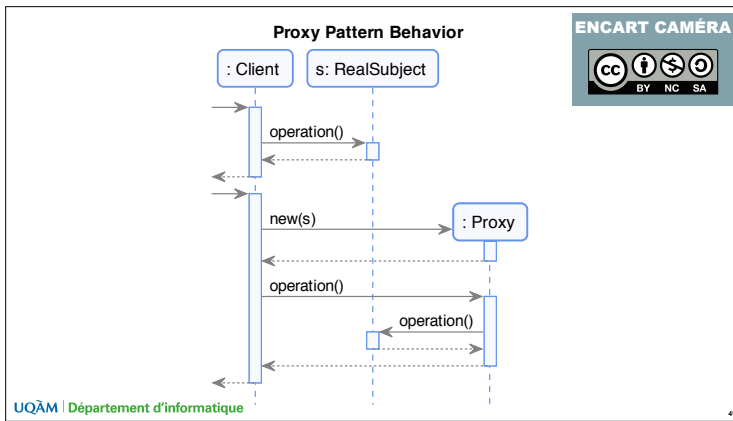


- **Séparer** l'interface de l'implémentation
- Réifier le concept de "**procuration**"
 - pour **substituer un objet inaccessible** par un autre,
 - qui lui est accessible.
- **Travailler avec le substitut** et le vrai objet de manière transparente



ENCART CAMÉRA





Conséquences

- L'inaccessibilité du sujet est **transparente**
- **Proxy et Sujet** peuvent être aisément **échangés**
- **Le proxy n'est pas une extension** (sous-classe) du sujet,
 - mais simplement une **réplique exacte**,
 - au **même niveau** dans la hiérarchie

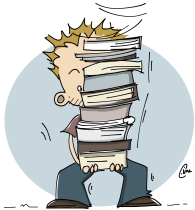
UQÀM | Département d'informatique

ENCART CAMÉRA
CC BY NC SA

48

Où le rencontrer dans la “vraie-vie”^(c) ?

- Principe d'échantillonnage de données
 - Image volumineuse à charger
- Gestion des droits d'accès
 - Le proxy gère les droits, l'objet le métier
- Objets distants
 - Java RMI, talons (stubs) d'accès à des WebServices
- Bouchons / Espions pour le test logiciel



<https://mosser.github.io/>



<https://ace-design.github.io/>

Abonne toi à la chaine,
et met un pouce bleu !

