INF4067
2023-2024

Université de Yaoundé I

INF4067 : UML et Design Patterns

Patterns de Comportement

Décembre 2023

Valéry MONTHE

valery.monthe@facsciences-uy1.cm

Bureau R114, Bloc pédagogique 1

Présentation

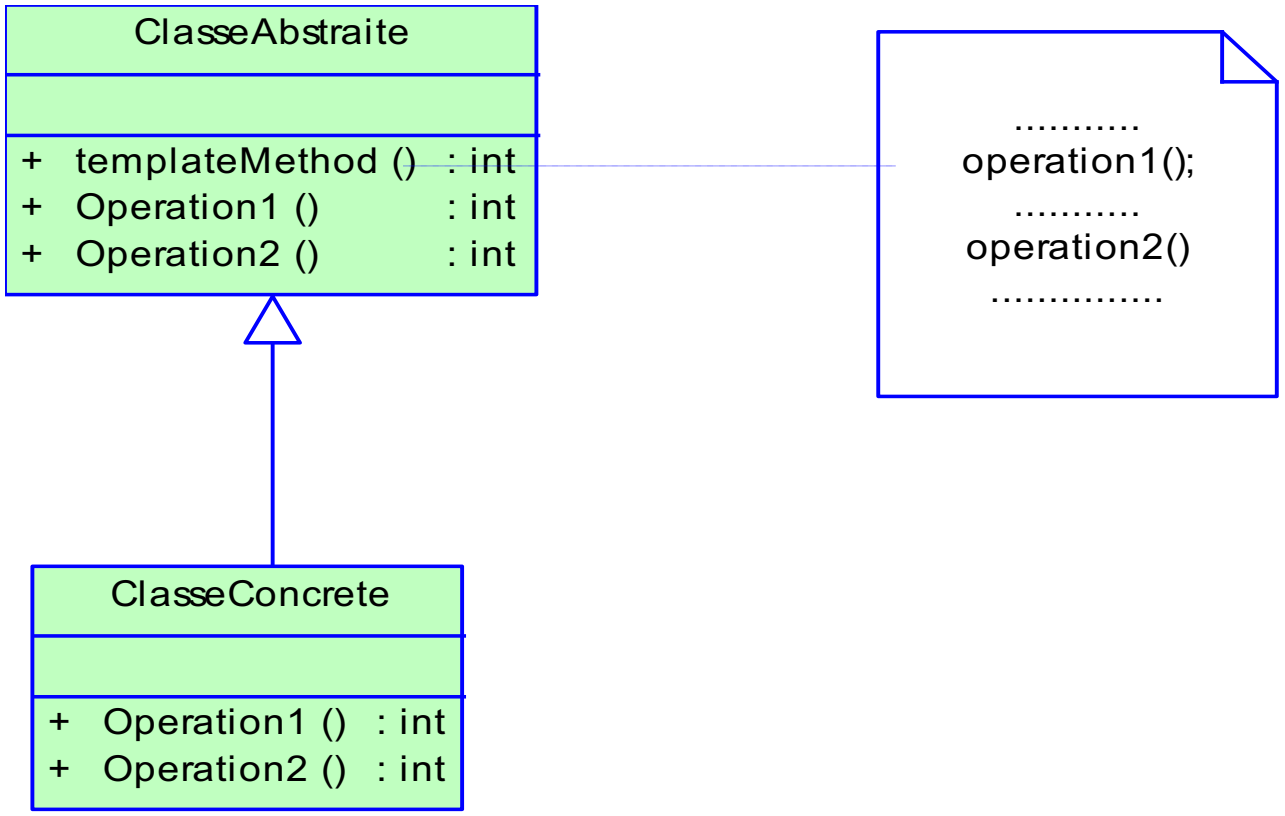
- Ils permettent de résoudre les problèmes liés aux comportements et à l'interaction entre les classes.
- Ils s'occupent des algorithmes et de la répartition des responsabilités entre les objets

Le patron Template Method

Template Method: Description

- Il permet de reporter dans des sous-classes certaines étapes de l'une des opérations d'un objet, ces étapes étant alors décrites dans les sous classes.

Template Method: Structure



Template Method: Participants

- La classe abstraite introduit la méthode patron ainsi que la signature des méthodes abstraites que cette méthode invoque.

- La sous-classe concrète implante les méthodes abstraites utilisées par la méthode patron de la classe abstraite.

- Il peut y avoir plusieurs classes concrètes.

Template Method: Exemple

- On considère un système de vente des véhicules en ligne. Ce système gère des commandes issues des clients au Cameroun et au Gabon. La différence entre ces deux commandes concerne notamment le calcul de la TVA. Au Cameroun le taux de TVA est toujours de 19,6%, mais au Gabon, il est variable et est de 12% pour les prestations et 15% pour le matériel.

Template Method: Exemple

1. Une solution avec 2 classes distinctes, inconvénient : duplication des codes identiques.

L'algorithme pour le Cameroun :

```
calculeMontantTtc()
{
 montantTva=montantHt * 0.196 ;
 montantTtc=montantHt + montantTva ;
}
```

L'algorithme pour le Gabon est donné par le pseudo-code suivant :

```
calculeMontantTtc()
{
 montantTva=(montantPrestationHt * 0.12) + (montantMatérielHt * 0.15) ;
 montantTtc=montantHt + montantTva ;
}
```

Nous voyons sur cet exemple que la dernière ligne de la méthode est commune aux deux pays

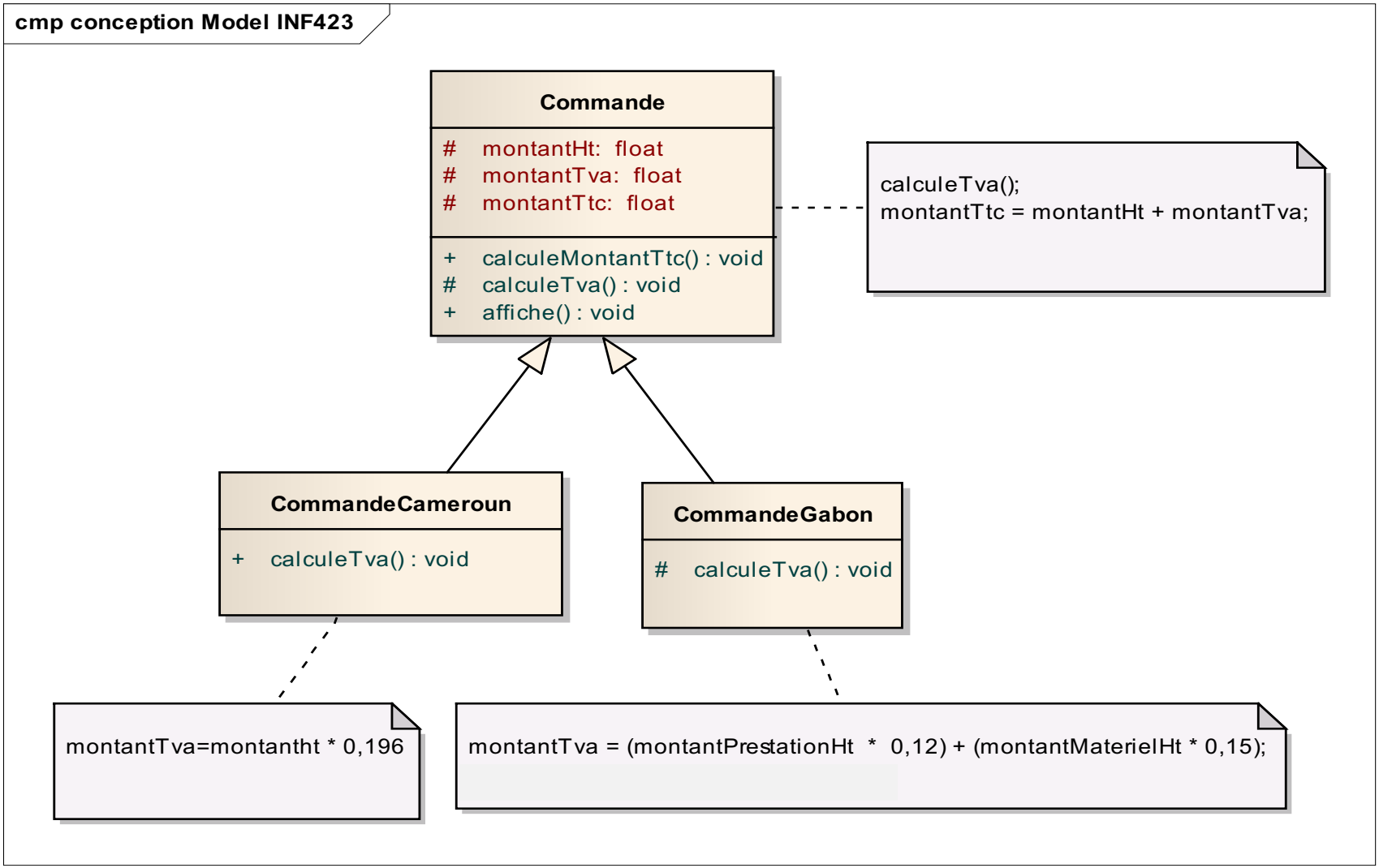
Exemple: Solution avec Template method

- Remplacer la première ligne par un appel d'une nouvelle méthode appelée ***calculeTva***.
- La méthode ***claculeMontantTtc*** est décrite dorénavant ainsi :

```
calculeMontantTtc()
{
 calculeTva () ;
 montantTtc=montantHt + montantTva ;
}
```

- La méthode ***calculeMontantTtc*** peut maintenant être factorisée. Le code spécifique a été déplacé dans la méthode ***calculeTva*** spécifique à chaque pays.
- La méthode ***calculeTva*** est introduite dans la classe Commande en tant que méthode abstraite.
- La méthode ***calculeMontantTtc*** est appelée méthode « patron » (template method).
- Une méthode patron introduit la partie commune d'un algorithme qui est ensuite complétée par des parties spécifiques.

Exemple: Solution avec Template method

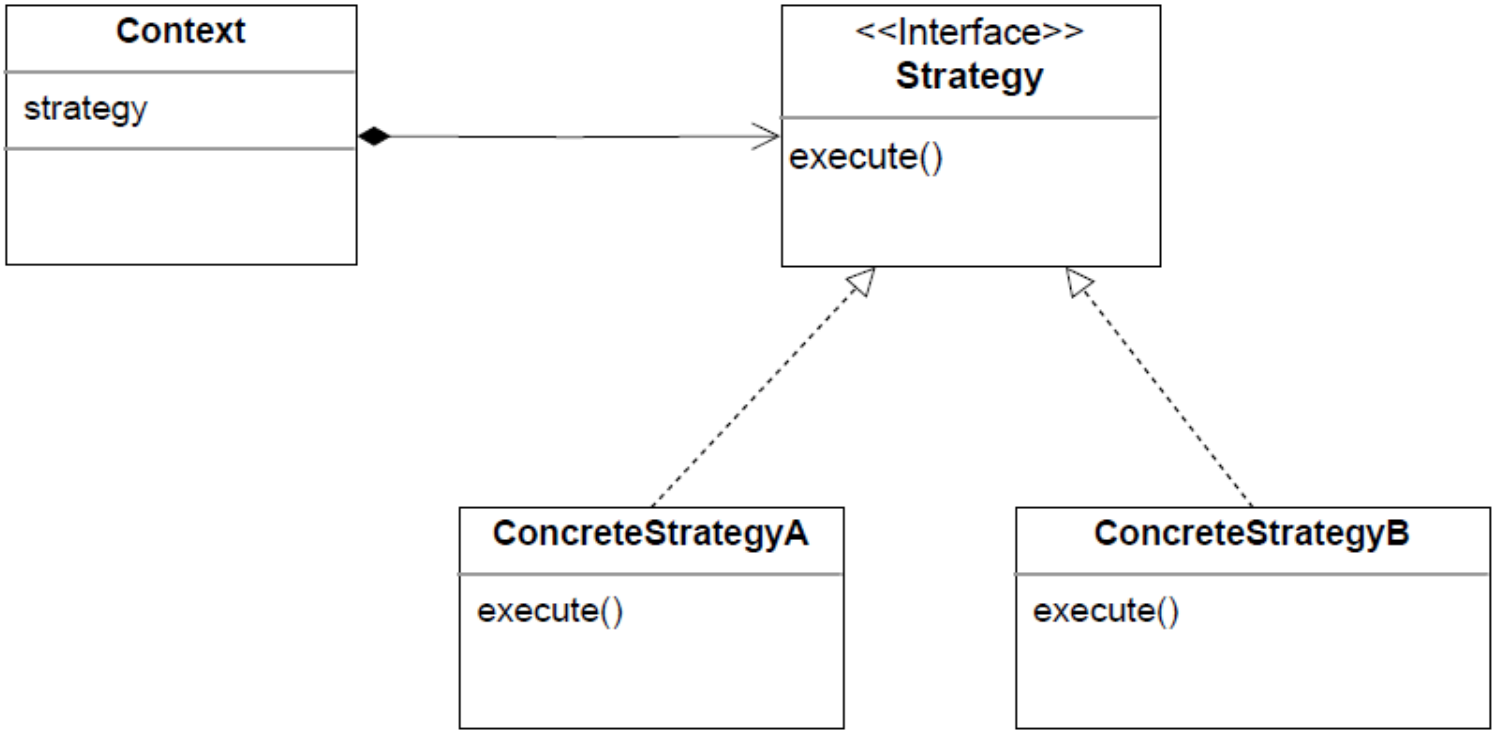


Le patron Strategy

Le pattern Strategy: Description

- Propose de définir un ensemble d'implémentations(des classes) d'un concept spécifique(interface).
- Les implémentations sont exécutées en fonction de l'appelant.
- Chaque implémentation représente une façon spécifique de résoudre le problème de l'appelant.
- C'est un pattern qui permet d'adapter la façon de faire en fonction de la situation
- Donc plus de flexibilité

Le pattern Strategy: Structure



Le pattern Strategy: Participants

- **Context** : c'est la classe qui définit l'objet dont le comportement doit être modifié dynamiquement.
- **Strategy** : l'interface ou classe abstraite qui définit les méthodes communes à tous les algorithmes pouvant être utilisés par l'objet Context.
- **ConcretStrategyA** et **ConcretStrategyB** sont les classes qui implémentent l'interface Strategy et fournit l'implémentation réelle de l'algorithme.

A l'exécution, un objet ***Context*** utilise un objet ***ConcretStrategy*** sélectionné lors de sa création pour effectuer ses opérations.

Le pattern Strategy: Exemple

On souhaite réaliser une application de sauvegarde d'images des utilisateurs. Pour cela, l'application doit :

- Compresser l'image en utilisant un algorithme de compression selon le format de l'image(JPEG, PNG, GIF, …)
- Appliquer différents filtres suivant un large choix de filtres

Exemple: solution 1

```
public class ImageStorage {

private String compressor;
private String filter;

public ImageStorage(String compressor, String filter) {
this.compressor = compressor;
this.filter = filter;
}

public void store(String fileName){
if("jpeg".equals(compressor))
// appeler la méthode de compression adaptée aux images JPEG.
System.out.println("Compression spécifique à JPEG");

else if("png".equals(compressor))
// appeler la méthode de compression adaptée aux images PNG.
System.out.println("Compression spécifique à PNG");

if("high-contrast".equals(filter))
System.out.println("Applying high contrast filter");
else if("BnW".equals(filter))
System.out.println("Applying black and white contrast filter");
}
}
```

Exemple: Solution avec Strategy