Université Constantine 2
جامعة قسنطينة 2

TML

Refactoring

Mauvaises odeurs

Manel Djenouhat, Sahar Smaali

NTIC/ TLSI

Refactoring

Bad smells

- Ballonnements
- Abuseurs de l'OO
- Empêcheurs de changement
- Dispensables
- Coupleurs

Techniques

- Méthodes de composition
- Déplacement d'entités entre objets
- Organisation des données
- Simplification des expressions conditionnelles
- Simplification des appels de méthodes
- Gestion de la généralisation

Méthodes de Composition

- **Extraire une méthode**
  - Permet de convertir un fragment de code en une méthode dont le nom indique son utilité

```
void printOwing(double amount) {
      printBanner();

      //print details
      System.out.println ("name:" + _name);
      System.out.println ("amount" + amount);
}
```

```
void printOwing(double amount) {
      printBanner();
      printDetails(amount);
}

void printDetails (double amount) {
      System.out.println ("name:" + _name);
      System.out.println ("amount" + amount);
}
```

Méthodes de Composition

- **Incorporer une méthode**
  - Déplacer le contenu d'une méthode vers ses appelants et supprimer la méthode
  - Utile lorsque le nom d'une méthode est aussi clair que son contenu

```
int getRating() {
    return (moreThanFiveLateDeliveries()) ? 2 : 1;
}

boolean moreThanFiveLateDeliveries() {
    return _numberOfLateDeliveries > 5;
}
```

```
int getRating() {
      return (_numberOfLateDeliveries > 5) ? 2 : 1;
}
```

Méthodes de Composition

- **Incorporer une variable temporaire**
  - Remplacer toutes les occurrences de cette variable par l'expression
  - Utile lorsqu'une variable temporaire empêche l'application d'un autre réusinage

```
double basePrice = anOrder.basePrice();
return (basePrice > 1000);
```

```
return (anOrder.basePrice() > 1000);
```

Méthodes de Composition

- **Remplacer une variable par une requête**
  - Extraire une expression dans une méthode. Remplacer la variable temporaire par un appel à la nouvelle méthode.

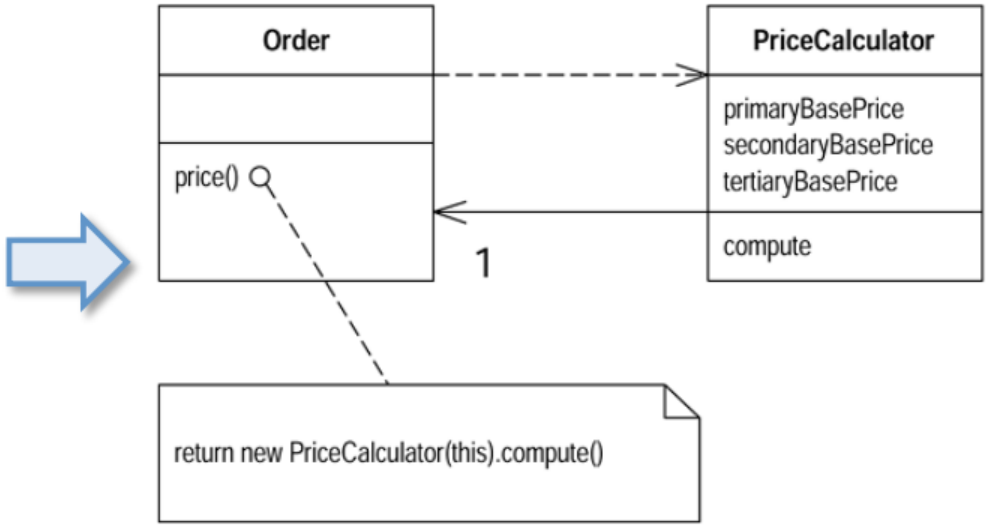
```
double basePrice = _quantity * _itemPrice;
if (basePrice > 1000)
    return basePrice * 0.95;
else
    return basePrice * 0.98;
```

```
if (basePrice() > 1000)
    return basePrice() * 0.95;
else
    return basePrice() * 0.98;
…
double basePrice() {
    return _quantity * _itemPrice;
}
```

Méthodes de Composition

- **Remplacer une méthode par un objet**
  - Extraire une expression dans une méthode. Remplacer la variable temporaire par un appel à la nouvelle méthode.

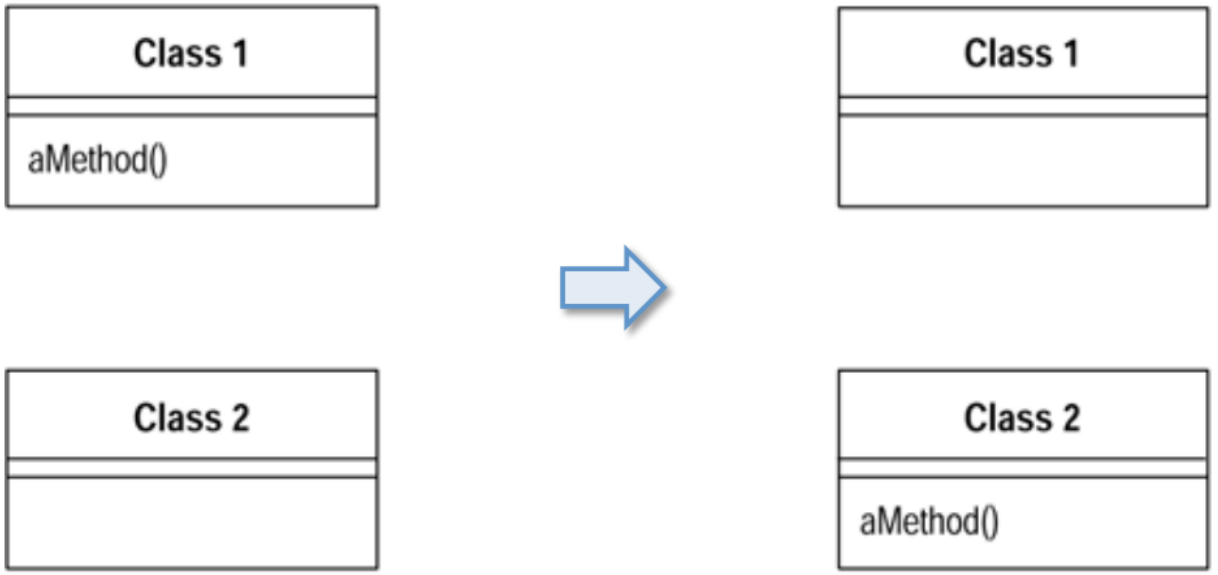
```
class Order {
   double price() {
       double primaryBasePrice;
       double secondaryBasePrice;
       double tertiaryBasePrice;
       // long computation;
       ...
   }
```



Deplacement

- **Déplacer une méthode**

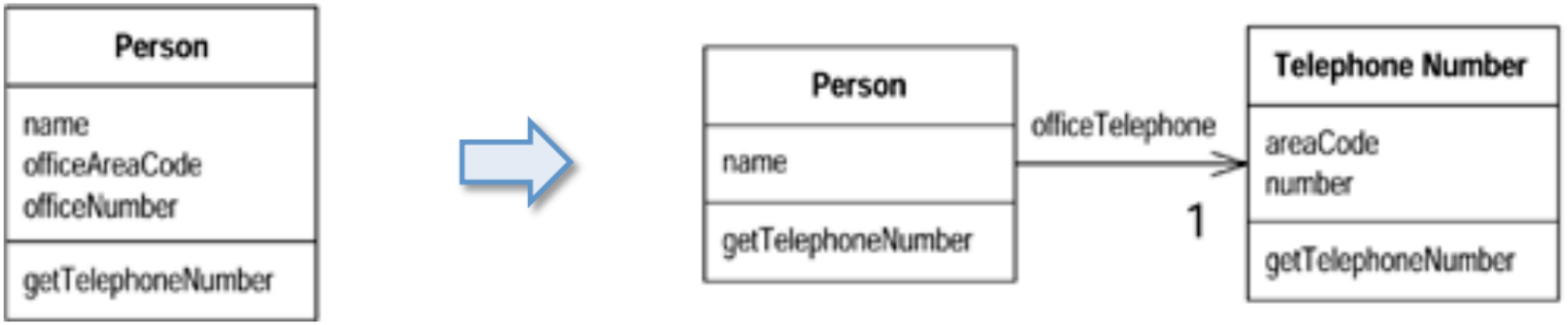
  Créer une méthode similaire dans la classe qu'elle utilise le plus. L'ancienne méthode pour être supprimée ou modifiée pour déléguer à la nouvelle.



Deplacement

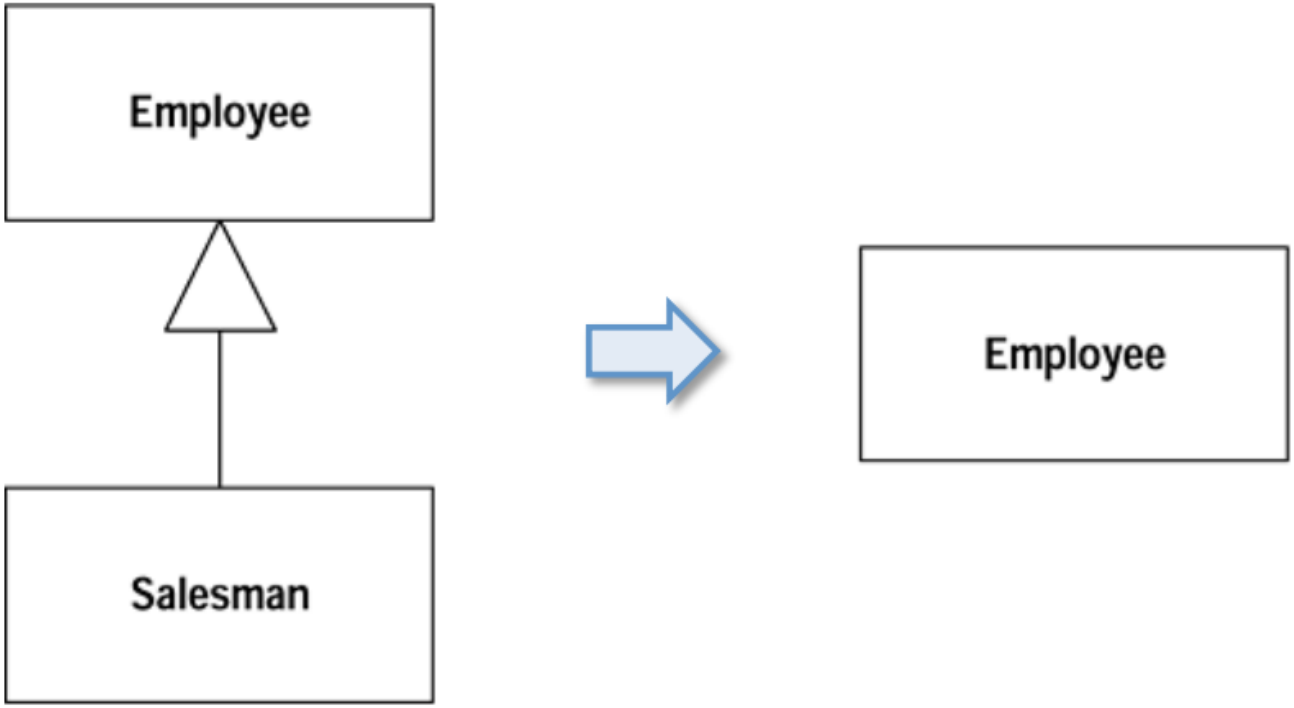
- **Extraire une classe**

  Créer une nouvelle classe et déplacer les champs et méthodes appropriés depuis l'ancienne classe vers la nouvelle.



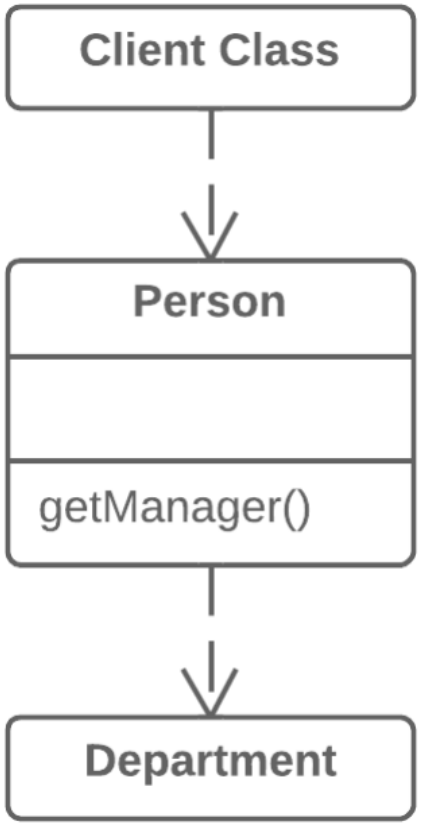
Deplacement

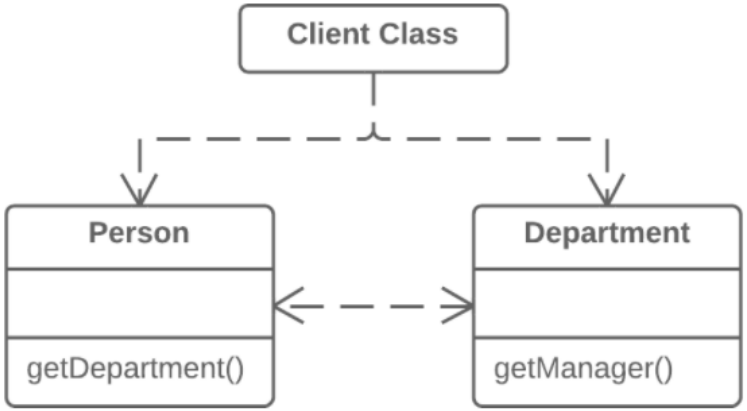
- **Aplatir la hiérarchie**



Deplacement

- **L'homme au milieu**





Simplification

- **Préserver l'objet entier**

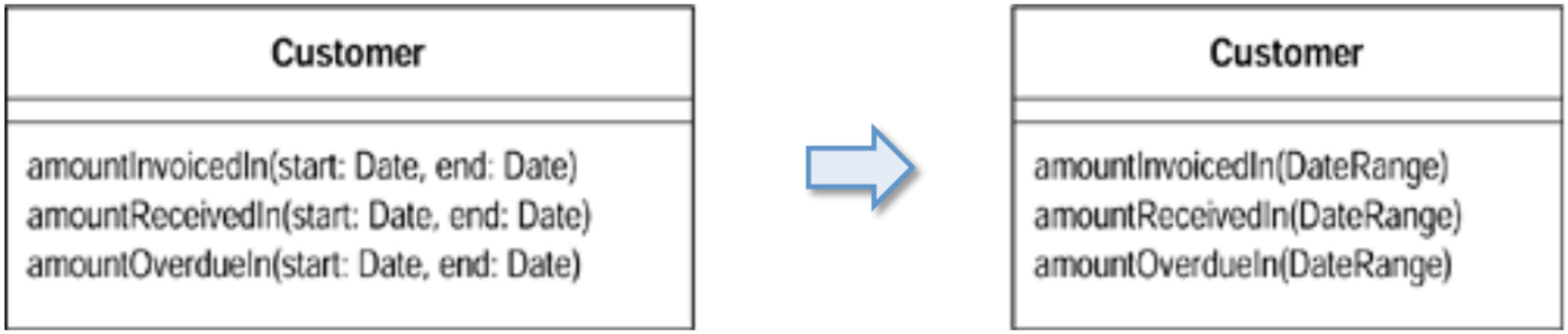
Passer l'objet entier plutôt que des valeurs extraites de ce dernier.

```
int low = daysTempRange().getLow();
int high = daysTempRange().getHigh();
withinPlan = plan.withinRange(low, high);
```

```
withinPlan = plan.withinRange(daysTempRange());
```

Simplification

- **Introduire un objet paramètre**



Mauvaises odeurs

Bad Smells

Dispensables

- **Duplication du code**
  - lorsque des parties spécifiques du code semblent différentes mais effectuent le même travail.
- **Lazy Class**
  - une classe ne fait pas assez pour attirer votre attention, elle doit être supprimée.
- **Dead Code**
  - Une variable, un paramètre, un champ, une méthode ou une classe n'est plus utilisé (généralement parce qu'il est obsolète).
- **Commentaires**
  - Une méthode est remplie de commentaires explicatifs.

Bad Smells

Ballonnements - Bloaters

- **Méthode longue**
  - **Signes et symptômes**
    - Une méthode contient trop de lignes de code : plus de 10 lignes.
  - **Techniques**
    - Extraire une méthode , Remplacer une variable par une requête, Introduire un objet parameter ou Preserver l'objet entier.
- **Classe Large**
  - **Signes et symptômes**
    - Une classe contient de nombreux champs / méthodes / lignes de code.
  - **Techniques**
    - Extraire une classe, Extraire une sous-classe, Extraire une interface

Ballonnements - Bloaters

- **Obsession des primitives**
  - **Signes et symptômes**
    - Utilisation de primitives au lieu de petits objets pour des tâches simples (telles que la devise, les plages, les chaînes spéciales pour les numéros de téléphone, etc.)
    - Utilisation de constantes pour coder les informations (comme une constante USER_ADMIN_ROLE = 1 pour faire référence aux utilisateurs avec des droits d'administrateur.)
    - Utilisation de constantes de chaîne comme noms de champ à utiliser dans des tableaux de données.
  - **Techniques**
    - Remplacer les valeurs de donnees par un objet, Introduire Objet parametre or Preserver l'objet entier

Bad Smells

Ballonnements - Bloaters

- **Longue Liste de paramètres**
  - **Signes et symptômes**
    - Plus de 3 ou 4 paramètres
  - **Techniques**
    - Remplcer parametre par un appel a une methode, Introduire un objet paramete.

Bad Smells

Abuseurs de l'OO – Object-Orientation Abusers

- **Déclaration d'un switch**
  - **Signes et symptômes**
    - Un switch complexe ou une séquence de Si
  - **Techniques**
    - Extraire une methode, Deplacer une methode , polymorphisme ou sous-classes
- **Champs temporaires**
  - **Signes et symptômes**
    - Les champs temporaires n'obtiennent leurs valeurs (et sont donc nécessaires aux objets) que dans certaines circonstances. En dehors de ces circonstances, ils sont vides.
  - **Techniques**
    - Extraire une classe, Remplacer une méthode avec une méthode Objet

Bad Smells

Abuseurs de l'OO – Object-Orientation Abusers

- **Classes alternatives avec différentes interfaces**
  - **Signes et symptômes**
    - Deux classes exécutent des fonctions identiques mais ont des noms de méthodes différents
  - **Techniques**
    - Extraire Superclass, Renommer une methode

Bad Smells

Empêcheurs de changement – Change Preventers

- **Changement divergent**
  - **Signes et symptômes**
    - Avec le temps, certaines classes accumulent trop de responsabilités.
    - Des changements divergents sont présents lorsqu'une classe doit changer de façon différente pour des raisons différentes
    - Indique souvent un manque de cohésion
  - **Techniques**
    - Extraire une classe, Extraire une SuperClasss.,
- **Shotgun Surgery**
  - **Signes et symptômes**
    - S'applique lorsqu'un changement requiert plusieurs petits changements répartis dans le programme
  - **Techniques**
    - Deplacer une Methode and Deplacer un attribut

Bad Smells

Empêcheurs de changement – Change Preventers

- **Jalousie fonctionnelle (feature envy)**
  - **Signes et symptômes**
    - Une méthode appelle beaucoup d'accesseurs d'une autre classe pour obtenir les données dont elle a besoin.
  - **Techniques**
    - Extraire une méthode et Déplacer une méthode pour déplacer la méthode vers la classe à laquelle elle devrait appartenir.
- **Homme du milieu**
  - **Signes et symptômes**
    - Si une classe n'effectue qu'une seule action, et délègue le travail à une autre classe
  - **Techniques**
    - Remove Middle Man

Reference

- https://refactoring.guru/