Université de Montréal

Département d'Informatique et de Recherche Opérationnelle

Jad Karam Hiver 2007

Présentation du Projet IFT3051

- Introduction sur les environnements:
  - SAD (Ptidej)
    - Framework en Java pour analyse de code
  - SOUL
    - Declarative Meta Programming Language

  Objectifs du projet
  - Comparaison SAD (Ptidej) et SOUL
  - Proposition d'algorithmes d'anti-patrons

Comparaisons

Anti-Patrons présents dans SAD:

- Algorithmes déjà implémentés dans SAD
  - Grandes Classes
    - Nombreuses méthodes
  - Longue liste de paramètres
    - Nombreux Paramètres
  - Classes de Données

Comparaisons

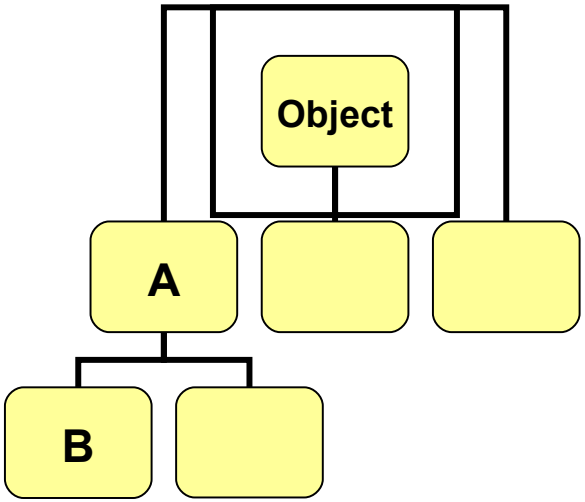
- Algorithmes proposés à rajouter:
  - Grandes Classes
    - Nombreuses Instances
  - Longues Méthodes
    - Nombreuses Instructions
    - Nombreuses variables locales
  - Liste de Paramètres
    - Paramètres Redondants
  - Blocs de Données (Data Clumps)

Comparaisons

- Algorithmes proposés à rajouter:
  - Interfaces
    - Interfaces Inappropriées
    - Méthodes abstraites non implémentées
  - Responsabilité
    - Feature Envy
    - Classe Déléguée
  - Code Inutile
    - Classes Paresseuses

Anti-Patrons

- Grandes Classes
  - Nombreuses Instances
    - Calculer le nombre dans chaque branche
      - `public void visit(VariableDeclerator n)`
    - Trouver les similitudes dans la branche



Anti-Patrons

- Longues Méthodes
  - Nombreuses Instructions
    - `Public void visit(Statement n)`
  - Nombreuses variables locales
    - `Public void visit(BlockStatement n)`
    - `Public void visit (LocalVariableDeclerator n)`
      - `n.f2.accpet(this)`

        `Equivalent à variableDeclerator.accept(this)`

Anti-Patrons

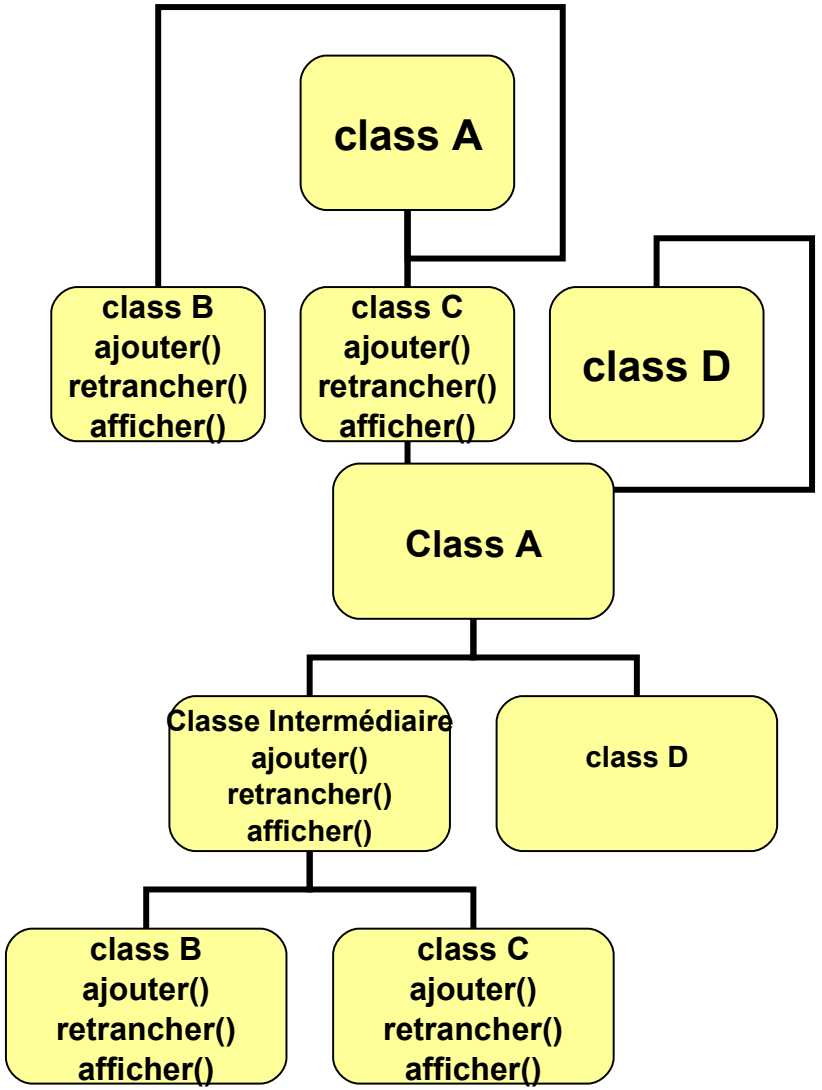
- Paramètres
  - Paramètres Redondants
    - `Public void visit(MethodDeclerator n)`
      - `n.f1.accept(this) //FormalParameters()`
    - `public void visit(FormalParameters n)`
    - `public void visit(FormalParameter n)`
      - On compare tous les paramètres de chaque méthode dans chaque classe pour trouver les redondances. (Appelants et appelés)

Anti-Patrons

- Blocs de Données (Data Clumps)
  - `public void visit(FormalParameter n)`
  - Adapter JWordNet a SAD.
  - Instancier le générateur de mots.
  - Générer les mots en rapports avec les paramètres.
  - Vérifier si tous les paramètres ont des sens en commun.

Anti-Patrons

- Interfaces
  - Interfaces Inappropriées
    - Produire des sous-ensembles de classes
    - Produire l'intersection des méthodes
    - Retourner la liste de méthodes en commun
    - Comparer avec les interfaces présentes

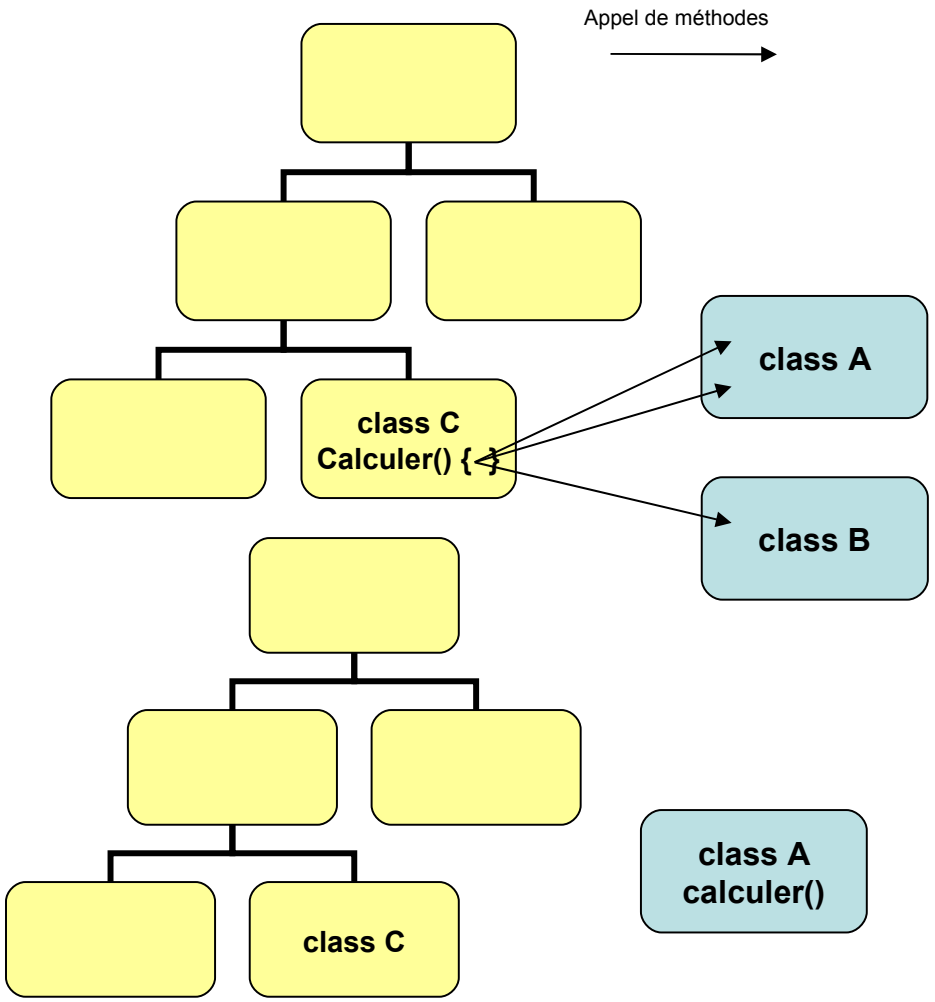


Anti-Patrons

- Interfaces
  - Méthodes abstraites non implémentées
    - `public void visit(UnmodifiedClassDeclaration n)`
      - `n.f3.accept(this) // « implements » NameList()`
      - `Pour chacun des noms recueillis on parcours l'interface par récurrence.`
    - `public void visit(UnmodifiedInterfaceDecleration n)`
      - `n.f4.accept(this) // InterfaceMemberDeclarator()`
      - `Retourne la liste des méthodes abstraites.`
    - `public void visit(MethodDeclaration n)`
      - `n.f4.accept(this)//On s'assure que Block n'est pas vide.`

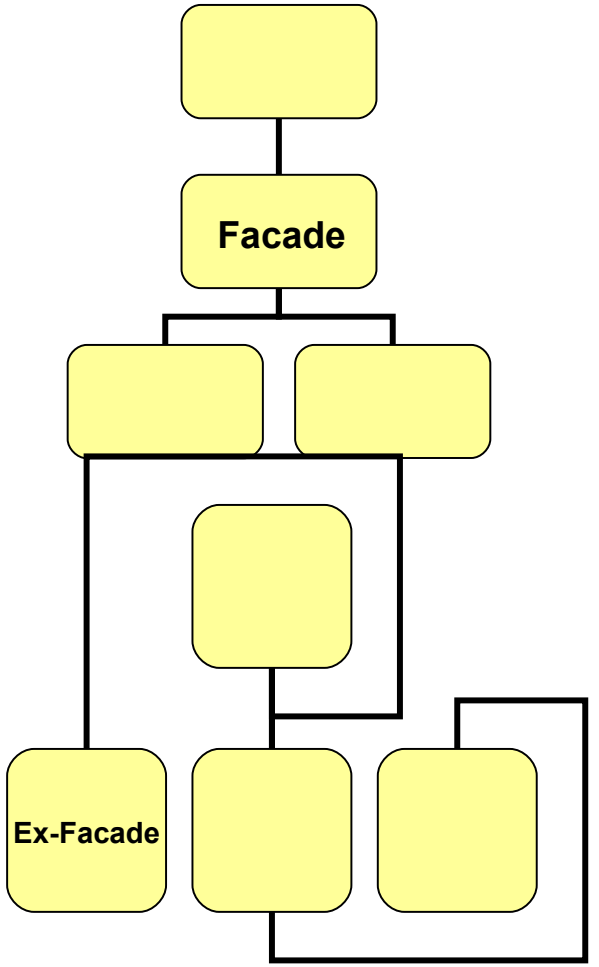
Anti-Patrons

- Responsabilités
  - Feature Envy
    - `public void visit(PrimaryPrefix n)`
    - On recueille le nombre d'appels de méthodes et nous déciderons si une migration sera nécessaire



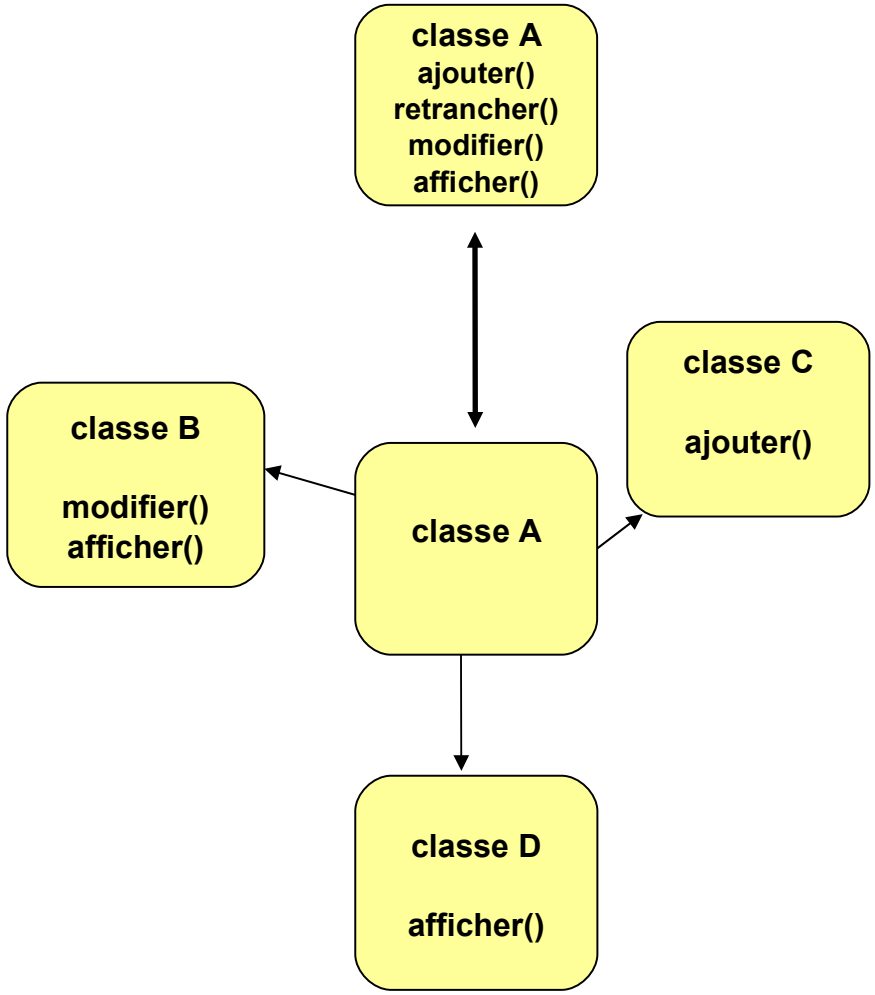
Anti-Patrons

- Responsabilité
  - Classe Déléguée (Middle Man)
    - En analysant la façade, nous pouvons déterminer l'importance de cette classe. Si elle ne traite pas de données, nous pouvons alors la baisser de niveau.
    - Nous pouvons appliquer des métriques par méthode pour déterminer le nombre d'opérations et analyser le contenu de cette classe.



Anti-Patrons

- Code Inutile
  - Classes Paresseuses
    - Éliminer les classes qui ne sont plus intéressantes pour le système
    - Appliquer des métriques (nombre d'instances, nombre de méthodes, nombre d'instructions) déjà implémentées.



Travaux Futurs

- À la suite de la détection:
  - Trouver des solutions réalisables
  - Appliquer les restructurations
- Utiliser des grammaires d'autres langages pour élargir le domaine de travail
- Vérifier la présence de commentaires et analyser leur cohérence par rapport au code.

Discussion

- Algorithmes ne pouvant être codés
- Smalltalk et Prolog vs. Java
- Personnes supplémentaire pour accélérer le travail.
- Ironie : Anti-Patrons dans SAD? (Commentaires)