Vers une traitification des collections

Guillaume Libersat (glibersat@sigill.org)

Sous la direction de Stéphane Ducasse
Équipe RMoD/Adam

15 juin 2008

1. Introduction
   - Contexte
   - Les traits
   - Tour d'horizon
   - Exemple
   - Questions
2. Analyse des collections
   - Introduction
   - Résultats
   - Souhaits
3. Proposition
   - Union
   - Hétérogénéité
   - Pollution/Ouverture
   - Affinage
4. Conclusion
   - Synthèse

Introduction

- Contexte
  - Équipe RMoD : « Comment aider les équipes de développement à maintenir et faire évoluer leurs logiciels ? »
    - Remodularisation
    - Constructions modulaires

Introduction

- Contexte
  - Équipe RMoD : « Comment aider les équipes de développement à maintenir et faire évoluer leurs logiciels ? »
    - Remodularisation
    - Constructions modulaires
- Les traits

Les Traits

Définition, Motivations

Traits : « Unités de comportement réutilisables et composables »

Motivations

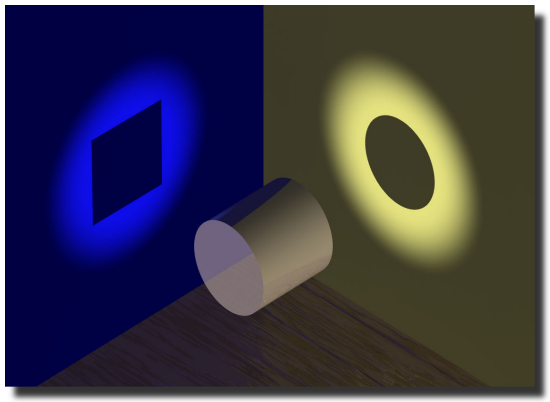
- Héritage, Mixins, ...
  - Duplication de code
  - Problème du diamand
  - Hiérarchies fragiles

Définition, Motivations

Traits : « Unités de comportement réutilisables et composables »

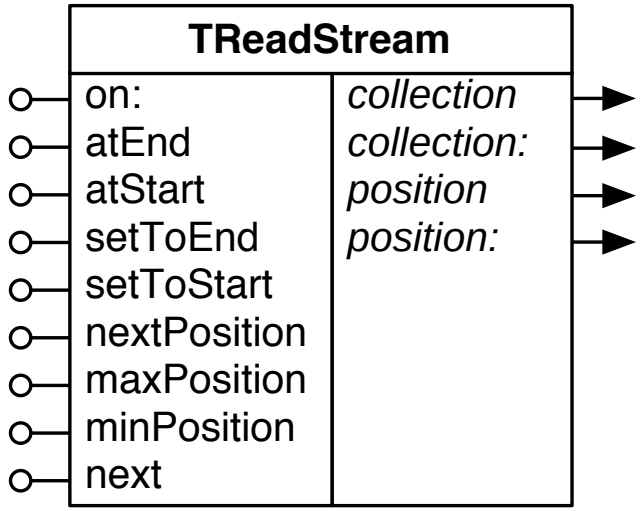
Motivations

- Héritage, Mixins, ...
  - Duplication de code
  - Problème du diamand
  - Hiérarchies fragiles
- Dualité des classes



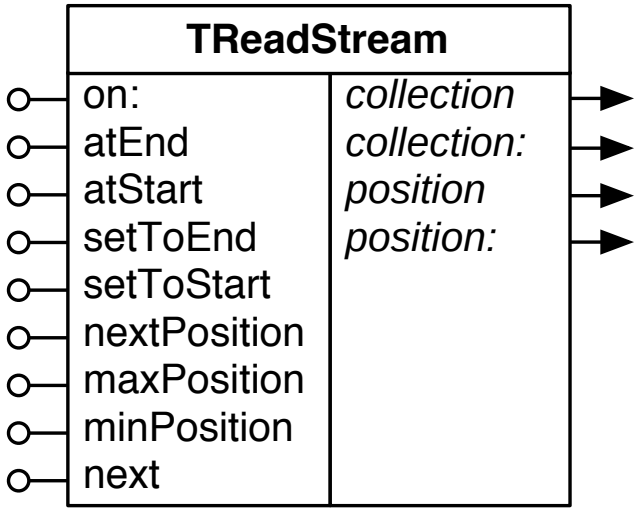
Les traits, concrètement

- Trait
  - Pas d'état
  - Méthodes fournies
  - Méthodes requises



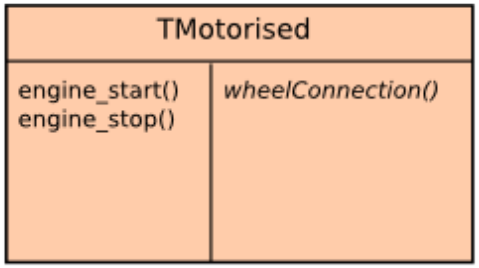
Les traits, concrètement

- Trait
  - Pas d'état
  - Méthodes fournies
  - Méthodes requises
- Notions clés
  - Simplicité
  - Complémentarité
  - Contrôle à la composition
  - Pas d'impact sur les performances



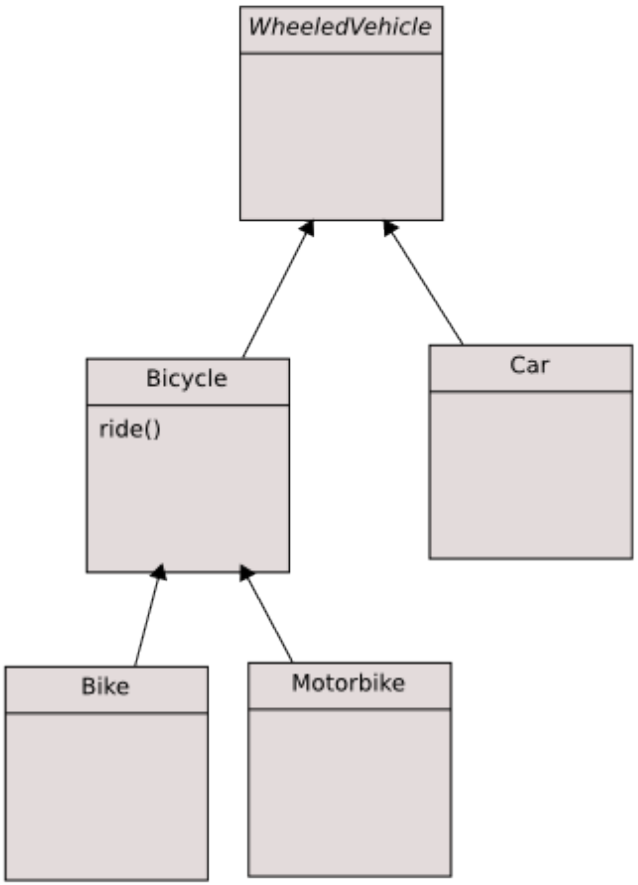
Traits : propriétés illustrées

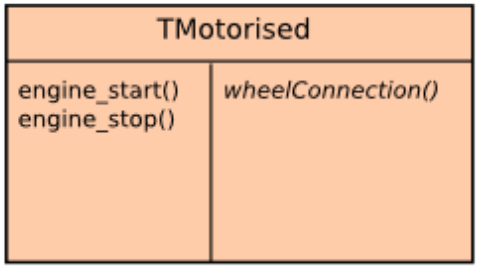
Composition : aplatissement



Traits : propriétés illustrées

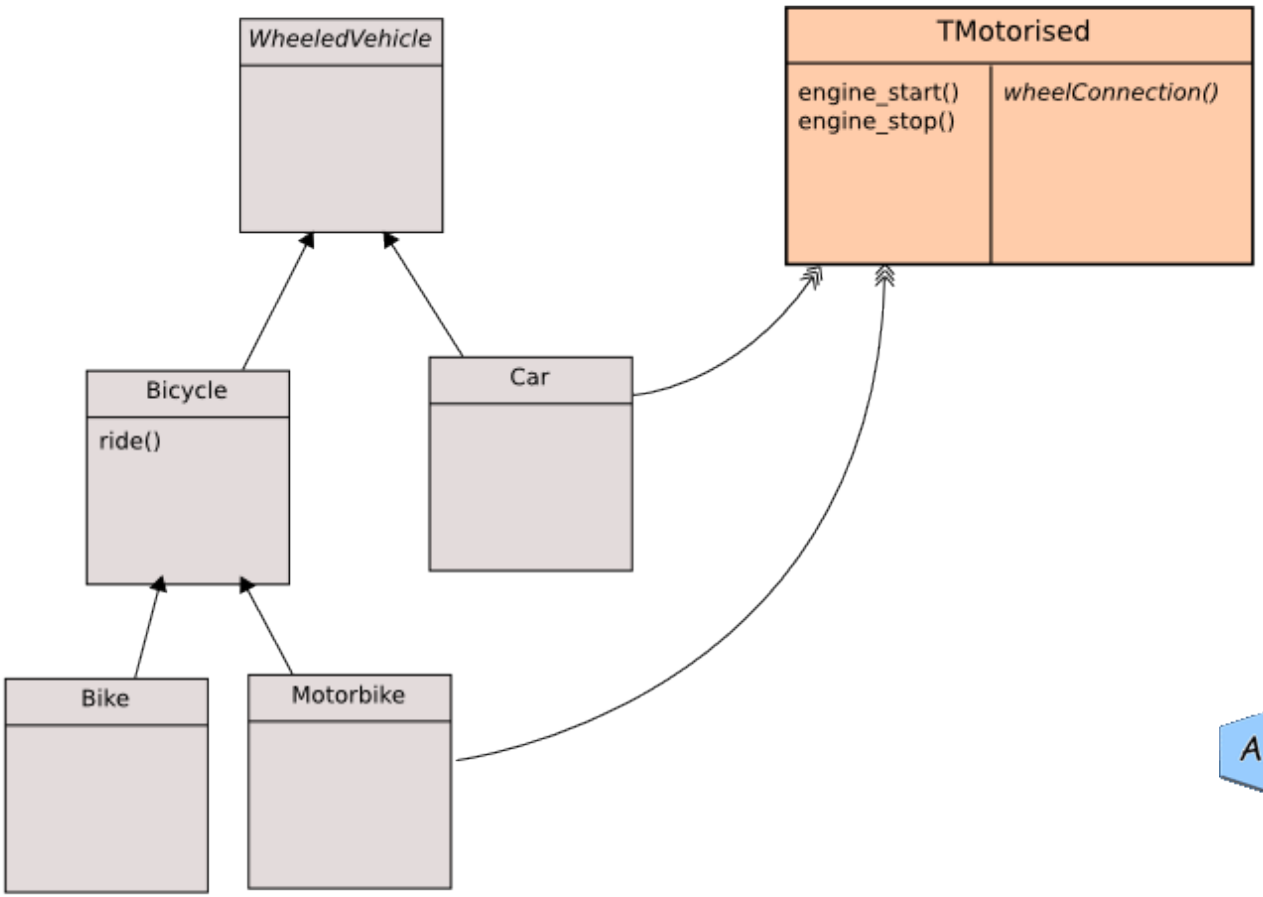
Composition : aplatissement





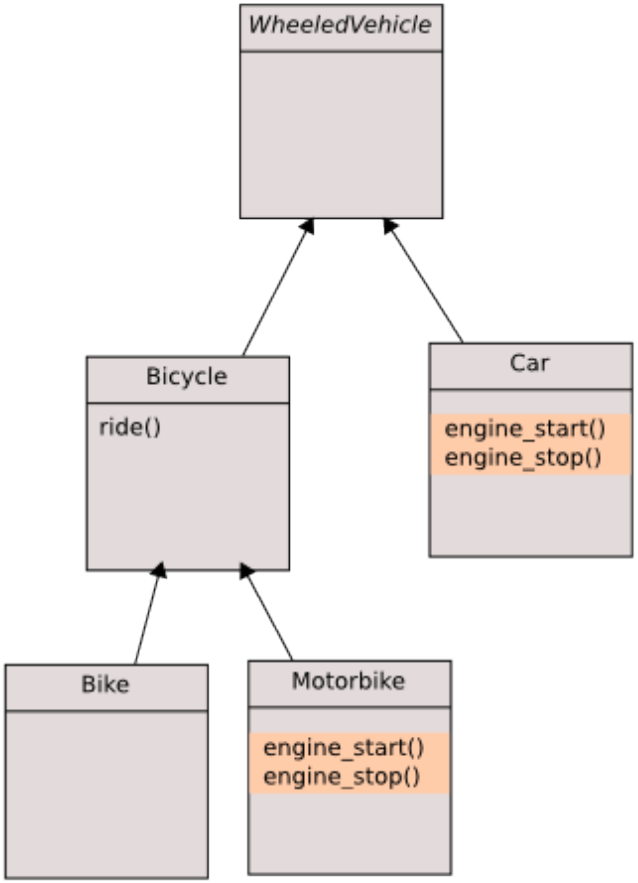
Traits : propriétés illustrées

Composition : aplatissement



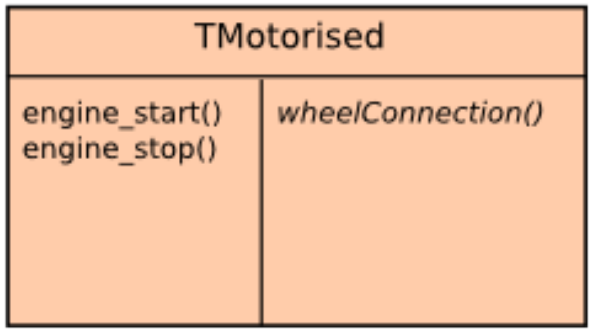
Traits : propriétés illustrées

Composition : aplatissement



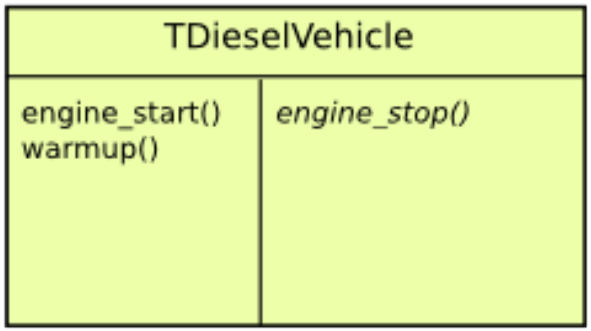
Traits : propriétés illustrées (2)

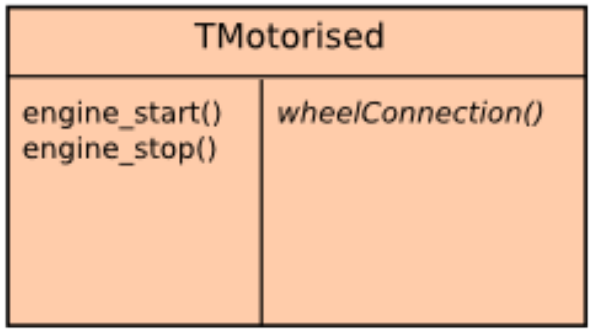
Résolution de conflits : exclusion



Traits : propriétés illustrées (2)

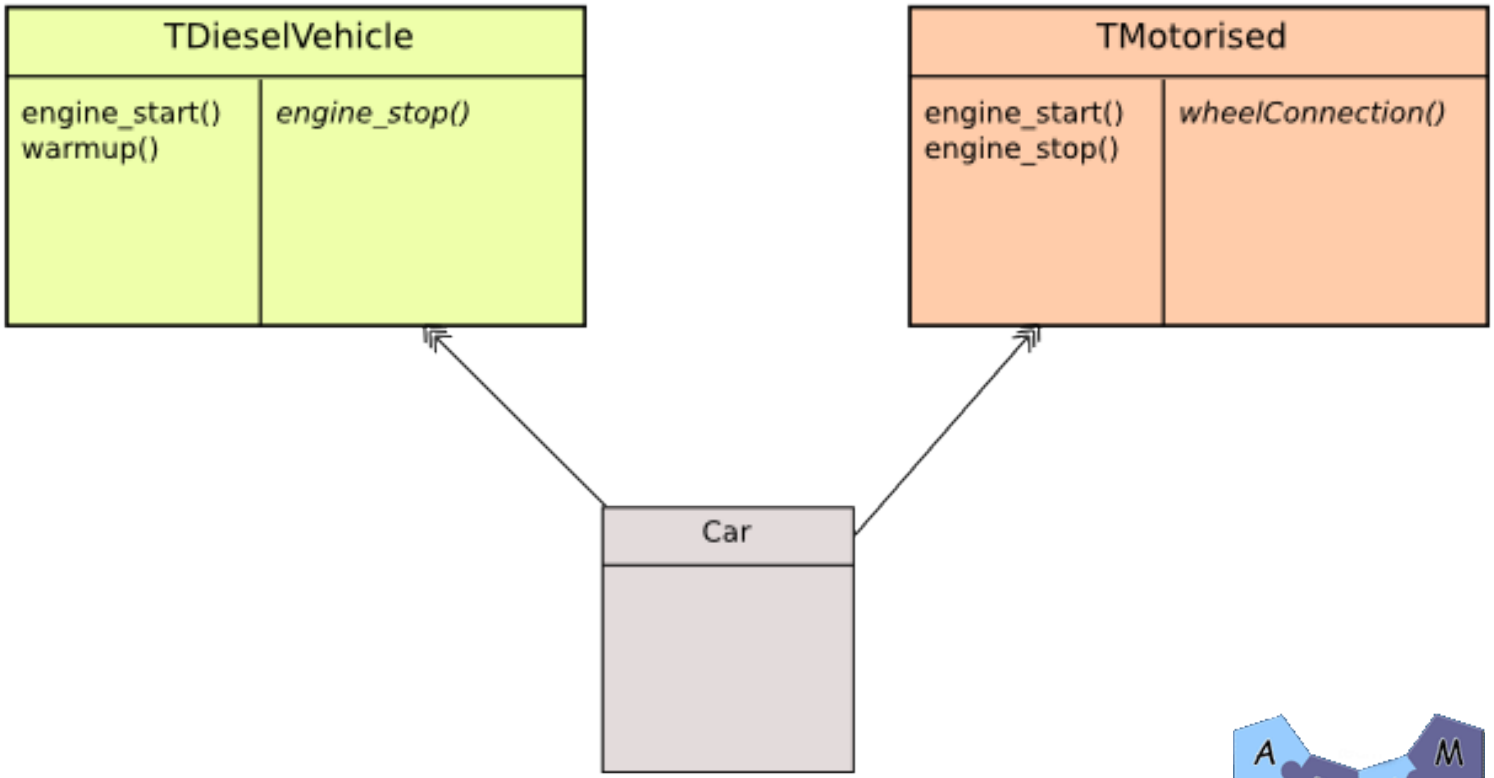
Résolution de conflits : exclusion





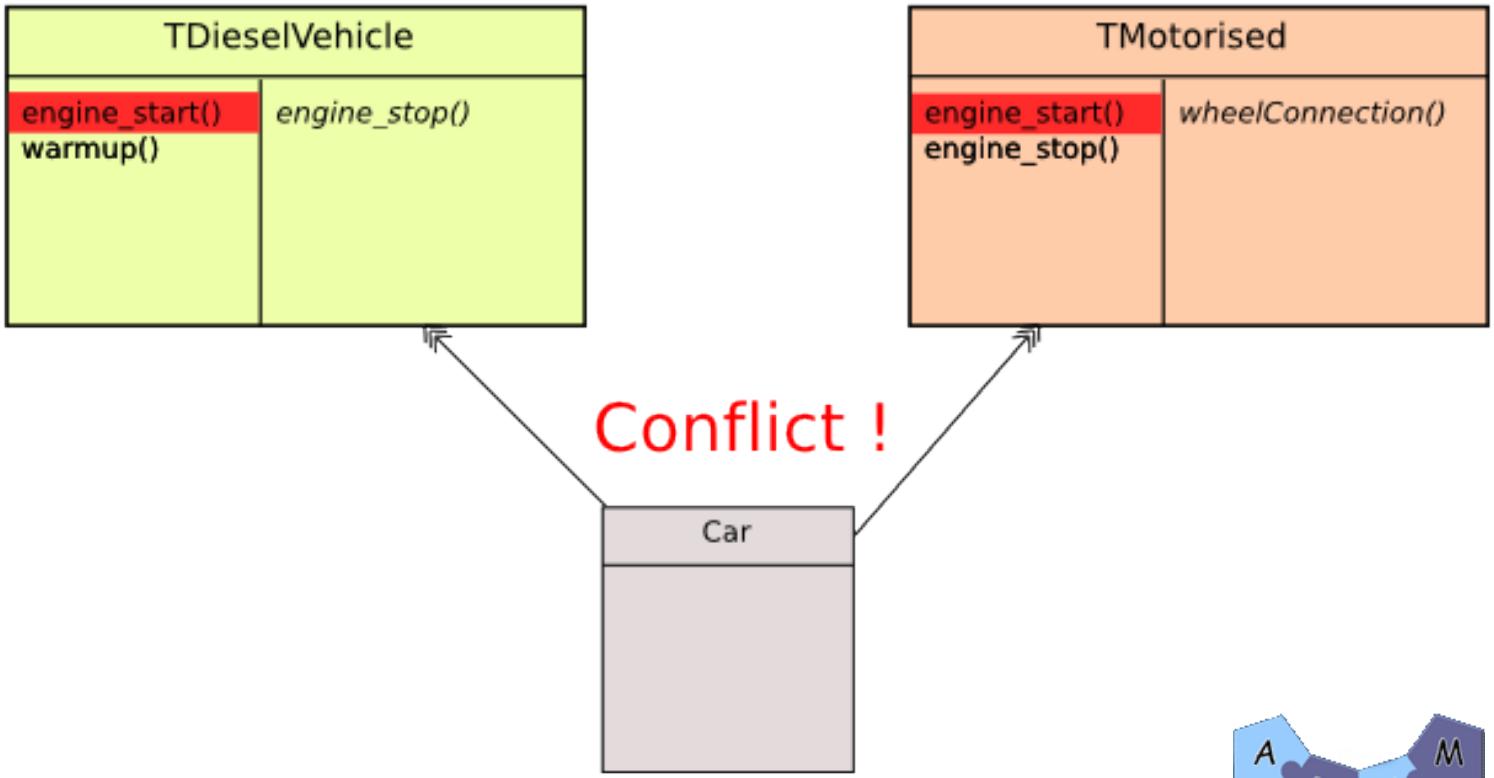
Traits : propriétés illustrées (2)

Résolution de conflits : exclusion



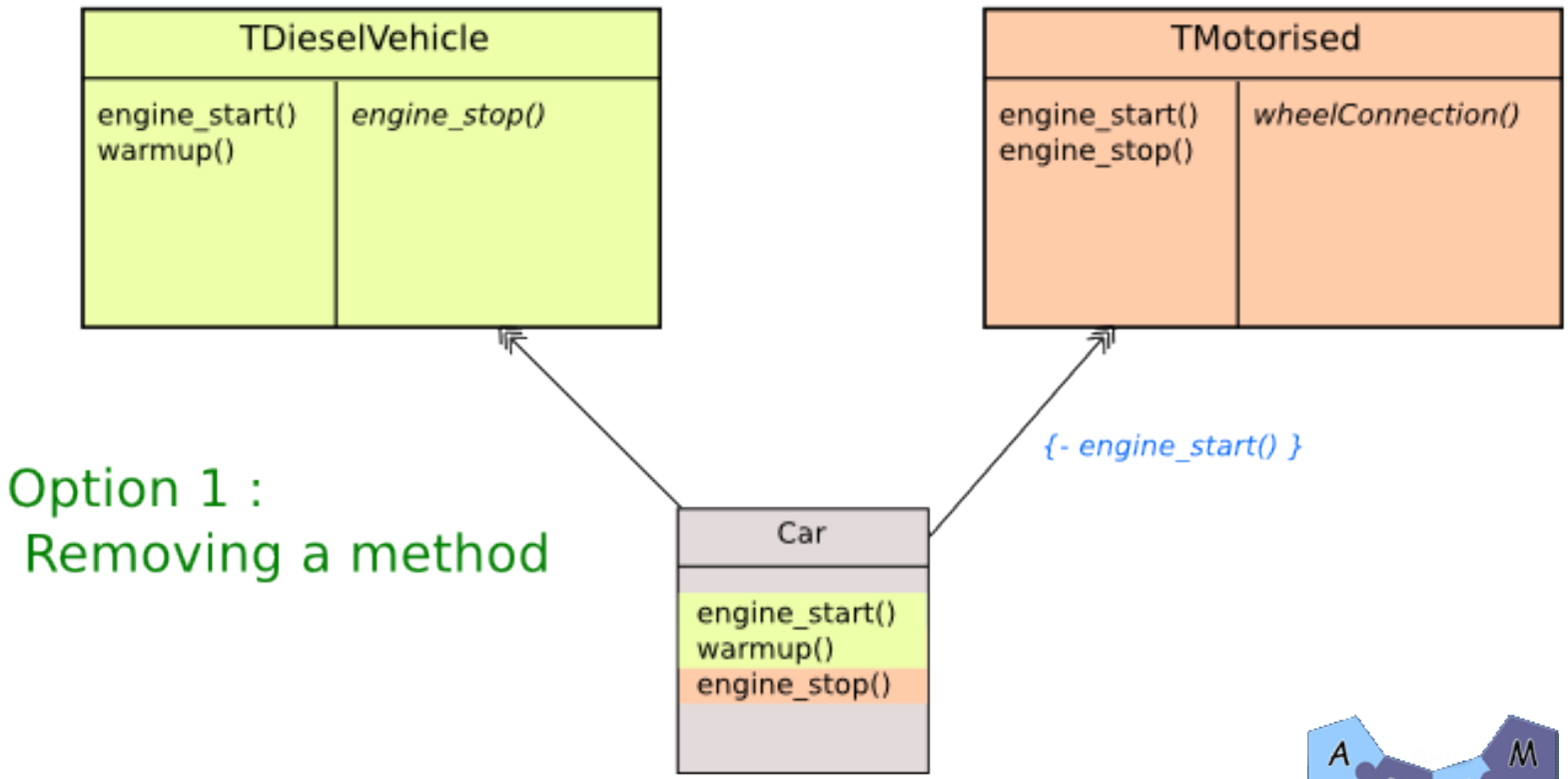
Traits : propriétés illustrées (2)

Résolution de conflits : exclusion



Traits : propriétés illustrées (2)

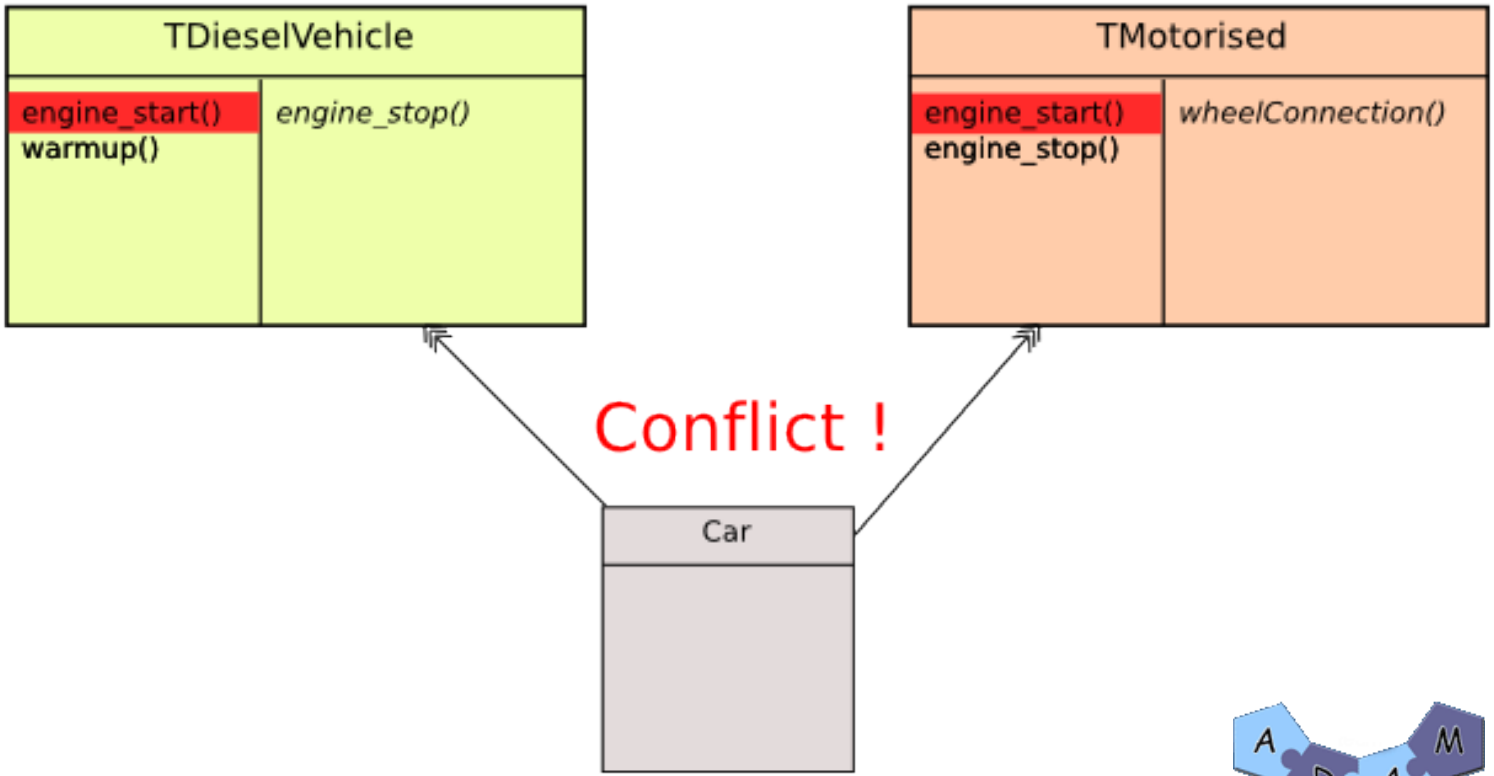
Résolution de conflits : exclusion



Option 1 :
Removing a method

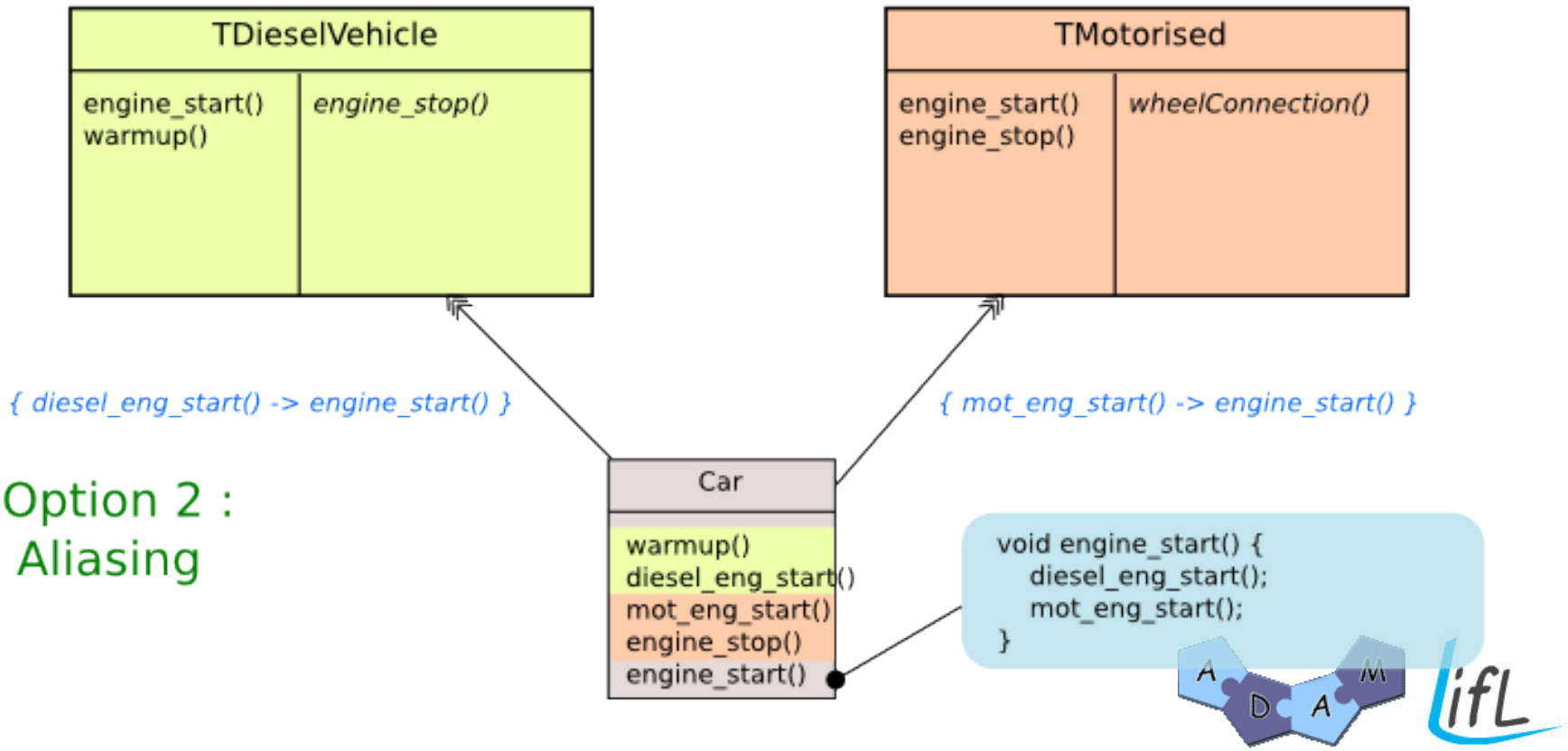
Traits : propriétés illustrées (3)

Résolution de conflits : aliasing



Traits : propriétés illustrées (3)

Résolution de conflits : aliasing



Introduction (2)

- Problématique
  - Traits efficaces
  - Traits peu éprouvés sur les grandes hiérarchies
  - Aucune solution satisfaisante pour les collections

Introduction (2)

- Problématique
  - Traits efficaces
  - Traits peu éprouvés sur les grandes hiérarchies
  - Aucune solution satisfaisante pour les collections
- Questions posées
  - Comment proposer une hiérarchie de traits pour ces cas ?
  - Comment s'assurer que la hiérarchie sera évolutive ?
  - Comment s'assurer que la hiérarchie restera consistante ?
  - Quelle est la granularité idéale des traits ?
  - Jusqu'à quel point pouvons-nous corriger les problèmes des collections ?

Introduction (2)

- Problématique
  - Traits efficaces
  - Traits peu éprouvés sur les grandes hiérarchies
  - Aucune solution satisfaisante pour les collections
- Questions posées
  - Comment proposer une hiérarchie de traits pour ces cas ?
  - Comment s'assurer que la hiérarchie sera évolutive ?
  - Comment s'assurer que la hiérarchie restera consistante ?
  - Quelle est la granularité idéale des traits ?
  - Jusqu'à quel point pouvons-nous corriger les problèmes des collections ?
- Proposition
  - Méthode outillée
  - Collections de Smalltalk

Analyse Des Collections

Les collections ?

« Une collection est un rassemblement (ou famille) d'objets. »– Wikipedia

- En bref
  - Bibliothèque
  - Indispensable

Les collections ?

« Une collection est un rassemblement (ou famille) d'objets. »– Wikipedia

- En bref
  - Bibliothèque
  - Indispensable
- Interrogations
  - Quels problèmes ?
  - Bonnes/Mauvaises pratiques ?

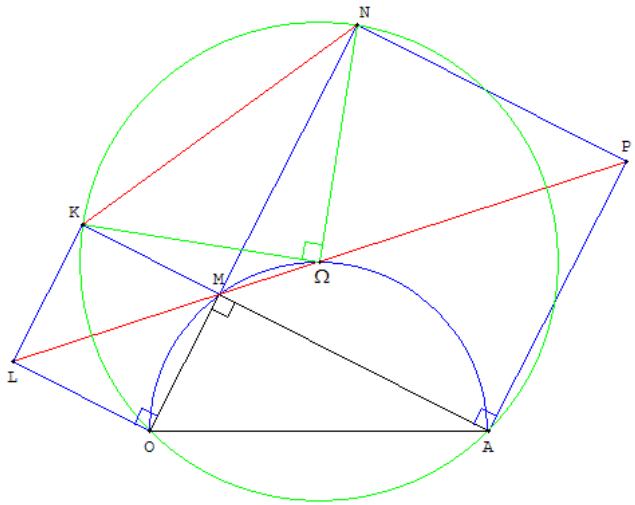
Les collections ?

« Une collection est un rassemblement (ou famille) d'objets. »– Wikipedia

- En bref
  - Bibliothèque
  - Indispensable
- Interrogations
  - Quels problèmes ?
  - Bonnes/Mauvaises pratiques ?
- Deux cas d'étude
  - Java et le *JCF*
  - Smalltalk avec *Squeak*

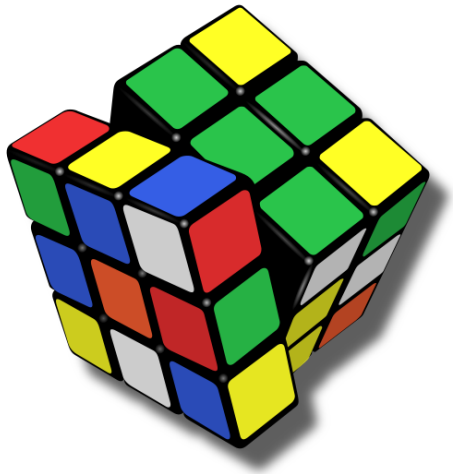
Résultat d'analyse

- Généralités
  - Nombreuses classes
  - Grande hétérogénéité
  - Orthogonalité des caractéristiques
  - Utilisation uniforme



Résultat d'analyse

- Généralités
  - Nombreuses classes
  - Grande hétérogénéité
  - Orthogonalité des caractéristiques
  - Utilisation uniforme
- Problèmes
  - Annulation de méthodes
  - Duplication de code
  - Pollution d'interfaces
  - Types primitifs
  - Réutilisation limitée



Ce que nous voulons

- De bonnes propriétés...
  - Bon polymorphisme
  - Bonne capacité d'expression
  - Bonne granularité
  - Protocoles bien définis

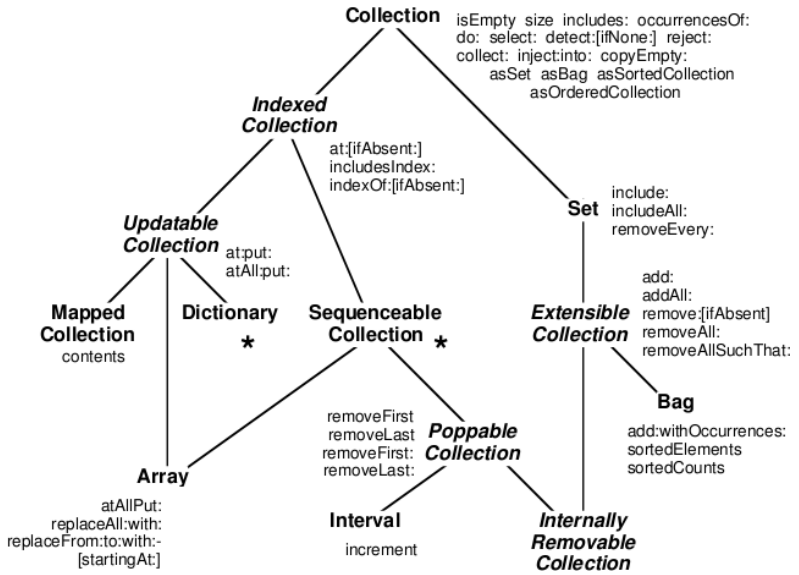
Ce que nous voulons

- De bonnes propriétés...
  - Bon polymorphisme
  - Bonne capacité d'expression
  - Bonne granularité
  - Protocoles bien définis
- En se préparant à...
  - Éviter les problèmes liés à l'héritage simple
  - Rendre les compositions orthogonales possibles
  - Prendre en compte l'hétérogénéité

Proposition

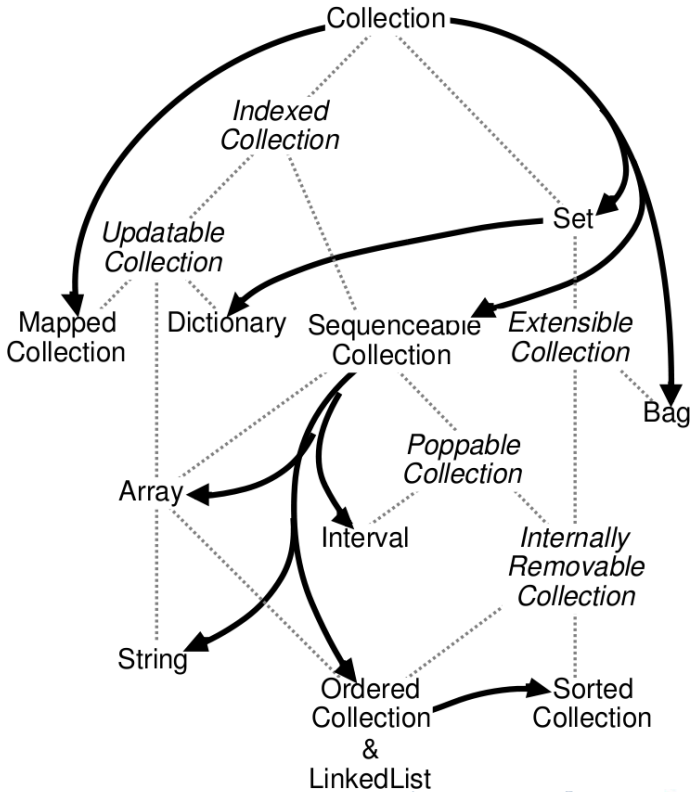
Les protocoles

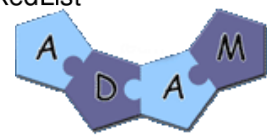
- Protocoles ?
  - Regroupement par préoccupation
  - Hiérarchique, multi-héritage
  - Indépendant de l'implantation
  - Standardisé



Les protocoles

- Protocoles ?
  - Regroupement par préoccupation
  - Hiérarchique, multi-héritage
  - Indépendant de l'implantation
  - Standardisé
- Idéal, mais en pratique...
  - Non unifié à la hiérarchie de classes
  - Force des mauvaises pratiques pour le suivre





Union des hiérarchies

- Avec les traits
  - « Héritage multiple »
  - 1 trait = 1 protocole
  - Se conformer devient facile

Union des hiérarchies

- Avec les traits
  - « Héritage multiple »
  - 1 trait = 1 protocole
  - Se conformer devient facile
- Gains
  - Implantation fidèle
  - Plus de duplications ni d'annulations
  - Meilleure compréhension
    - Développeur
    - Utilisateur

Problèmes d'hétérogénéité

- Multiplicité d'implantations
  - Caractéristiques (e.g. « Triable ») non liées à une implantation
  - Plusieurs structures de données (e.g. Tableau, Liste chaînée)

Problèmes d'hétérogénéité

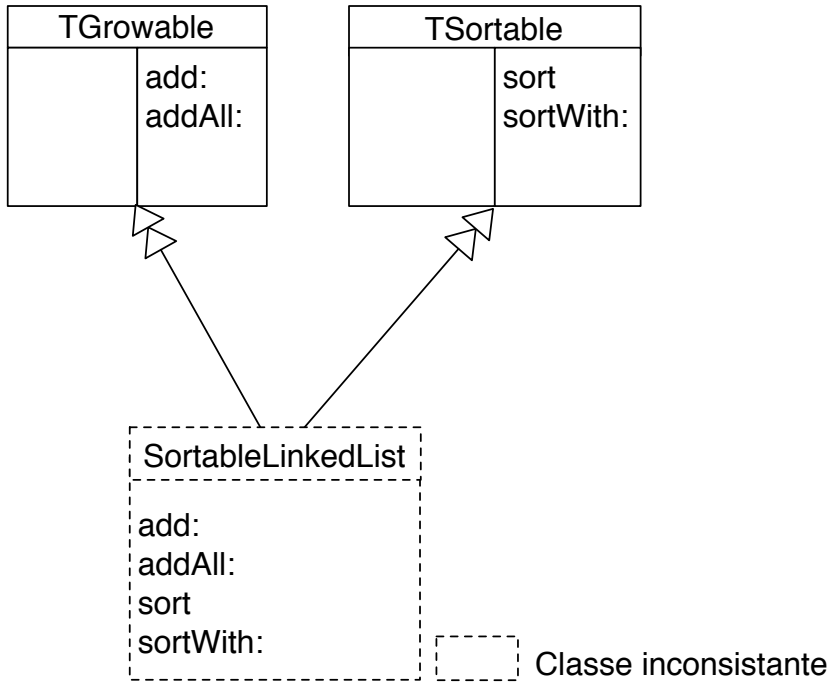
- Multiplicité d'implantations
  - Caractéristiques (e.g. « Triable ») non liées à une implantation
  - Plusieurs structures de données (e.g. Tableau, Liste chaînée)
- Système actuel : un trait par protocole

Problèmes d'hétérogénéité

- Multiplicité d'implantations
  - Caractéristiques (e.g. « Triable ») non liées à une implantation
  - Plusieurs structures de données (e.g. Tableau, Liste chaînée)
- Système actuel : un trait par protocole
- Résultat
  - Tous les cas ne sont pas exprimables
  - Mène à des inconsistances de composition

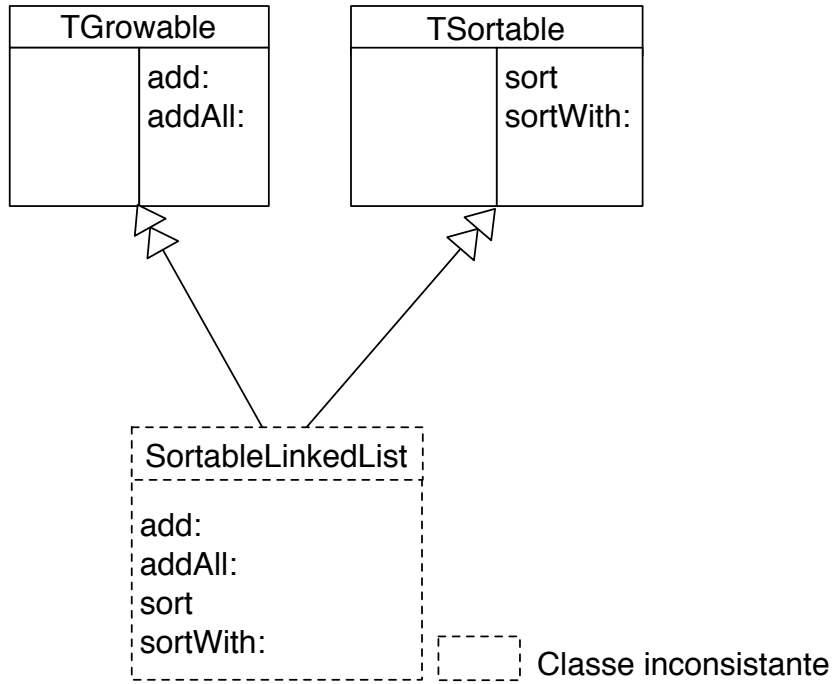
Exemple de composition inconsistante

- Exemple
  - TSortable : « Tableau »
  - TGrowable : « Liste Chaînée »
  - Quid de SortableLinkedList ?



Exemple de composition inconsistante

- Exemple
  - TSortable : « Tableau »
  - TGrowable : « Liste Chaînée »
  - Quid de SortableLinkedList ?
- Conclusion
  - Un trait par structure de données, pour chaque caractéristique
  - Exemple : TSortableArray, TSortableLinkedList

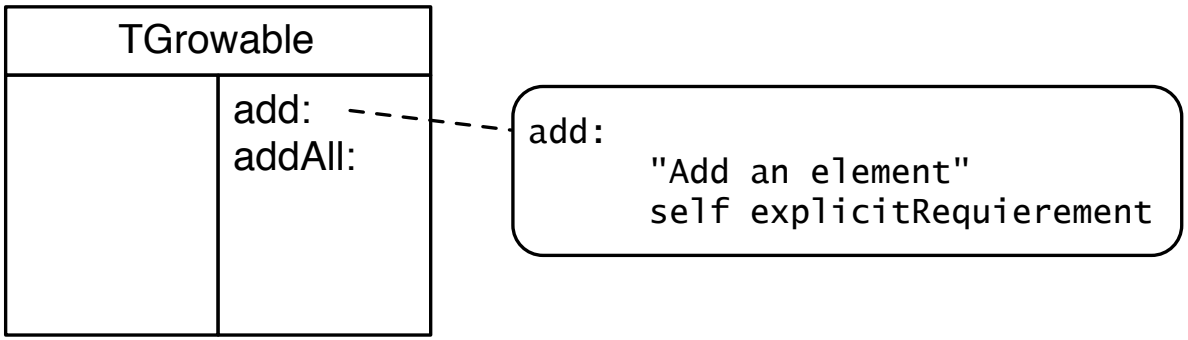


De la nécessité des protocoles

- Plusieurs traits par protocole
  - Implantations différentes mais même fonction
  - Respect du protocole, polymorphisme ?
  - Cohérence sémantique ?
  - Facilité de compréhension, d'extension ?

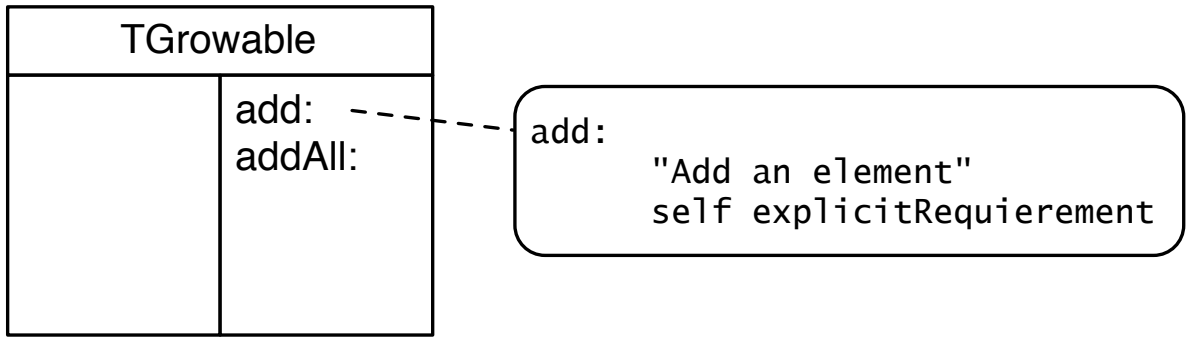
Consistance protocolaire

- Traits-protocoles
  - Donne corps aux protocoles
  - Assure le respect des messages
  - Documente automatiquement les implantations



Consistance protocolaire

- Traits-protocoles
  - Donne corps aux protocoles
  - Assure le respect des messages
  - Documente automatiquement les implantations



- En pratique
  - Chaque implantation le compose
  - Redéfinition des méthodes
  - Facilité de débogage

Consistance sémantique

- Consistance nécessaire ?
  - Messages assurés par les traits-protocoles
  - Consistance sémantique ?

Consistance sémantique

- Consistance nécessaire ?
  - Messages assurés par les traits-protocoles
  - Consistance sémantique ?
- Ajout de contrats
  - Spécification de contraintes par le protocole
  - Implanteurs soumis à ces contraintes
  - Détection des compositions inconsistantes

Contrats et traits

- Pas de support du modèle
  - Redéfinition, Aliasing, ...

Contrats et traits

- Pas de support du modèle
  - Redéfinition, Aliasing, ...
- Extension proposée
  - **Invariants** :
    $(invariant_{trait1} \wedge ... \wedge invariant_{traitn}) \wedge invariant_{composeur}$

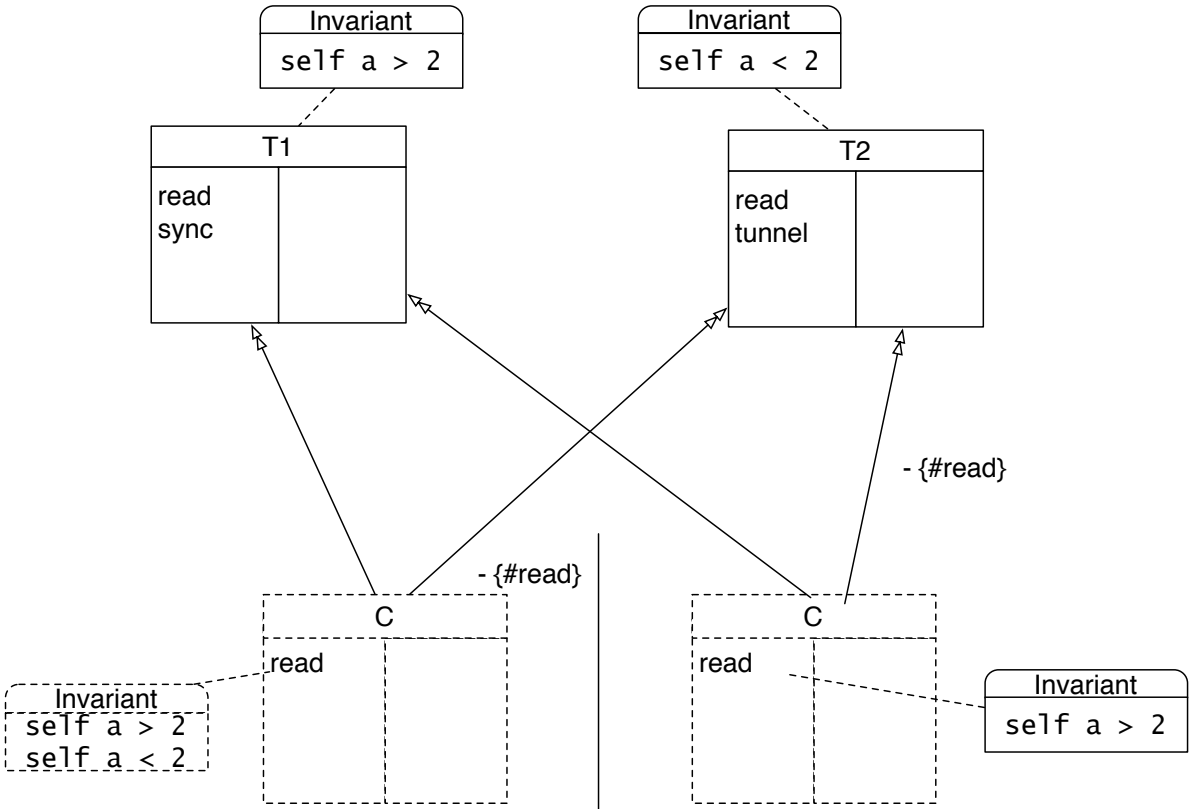
Contrats et traits

- Pas de support du modèle
  - Redéfinition, Aliasing, ...
- Extension proposée
  - **Invariants** :
    $(invariant_{trait1} \wedge ... \wedge invariant_{traitn}) \wedge invariant_{composeur}$
  - **Pré-conditions** :
    $(precond_{trait1} \vee ... \vee precond_{traitn}) \vee precond_{composeur}$

Contrats et traits

- Pas de support du modèle
  - Redéfinition, Aliasing, ...
- Extension proposée
  - **Invariants** :
    $(invariant_{trait1} \wedge ... \wedge invariant_{traitn}) \wedge invariant_{composeur}$
  - **Pré-conditions** :
    $(precond_{trait1} \vee ... \vee precond_{traitn}) \vee precond_{composeur}$
  - **Post-conditions** :
    $(postcond_{trait1} \wedge ... \wedge postcond_{traitn}) \wedge postcond_{composeur}$

Exemple de détection des inconsistances



Pollution et ouverture

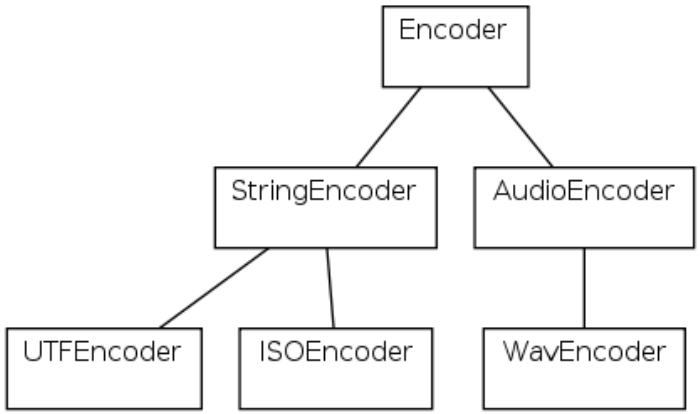
- Principalement due aux convertisseurs
  - #asPostcript, #utf8toISO, ...
  - Jusqu'à 50% des méthodes d'une classe
  - Occulte les méthodes intéressantes

Pollution et ouverture

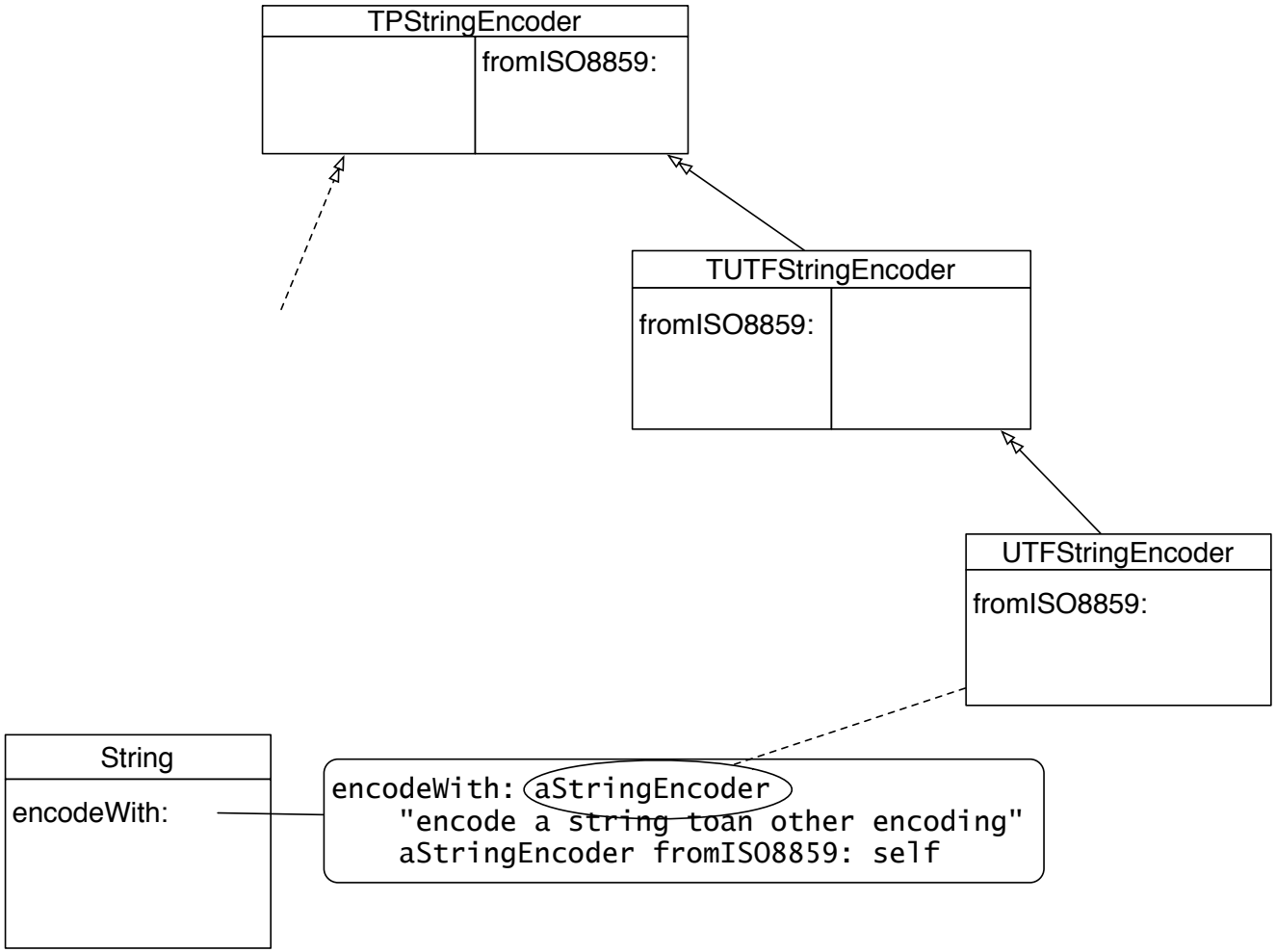
- Principalement due aux convertisseurs
  - #asPostcript, #utf8toISO, ...
  - Jusqu'à 50% des méthodes d'une classe
  - Occulte les méthodes intéressantes
- Convertisseurs génériques
  - Deux types de conversions
    - De contenu : #encodeWith :
    - De conteneur : #convertWith :

Pollution et ouverture

- Principalement due aux convertisseurs
  - #asPostcript, #utf8toISO, ...
  - Jusqu'à 50% des méthodes d'une classe
  - Occulte les méthodes intéressantes
- Convertisseurs génériques
  - Deux types de conversions
    - De contenu : #encodeWith :
    - De conteneur : #convertWith :
  - Protocoles utilisateurs
  - Convertisseurs implantant les protocoles
  - Support dans OBBrowser



Convertisseurs génériques : exemple



Affinage des protocoles

- Granularité cruciale
  - Flexibilité de la hiérarchie
  - Degré de compréhension
  - Difficile à définir

Affinage des protocoles

- Granularité cruciale
  - Flexibilité de la hiérarchie
  - Degré de compréhension
  - Difficile à définir
- Méthode outillée, 3 étapes
  1. Détermination des caractéristiques
  2. Création d'un protocole minimaliste
  3. Affinage par le concepteur

Méthode, étape 1

Détermination des caractéristiques par classe

Carac. \ Classe	Array	String	Stack	Set	...
Iterable	√	√	√	√	...
Growable				√	...
Indexable	√	√	√		...
Poppable			√		...
...	...	...	...	...	...

Méthode, étape 2

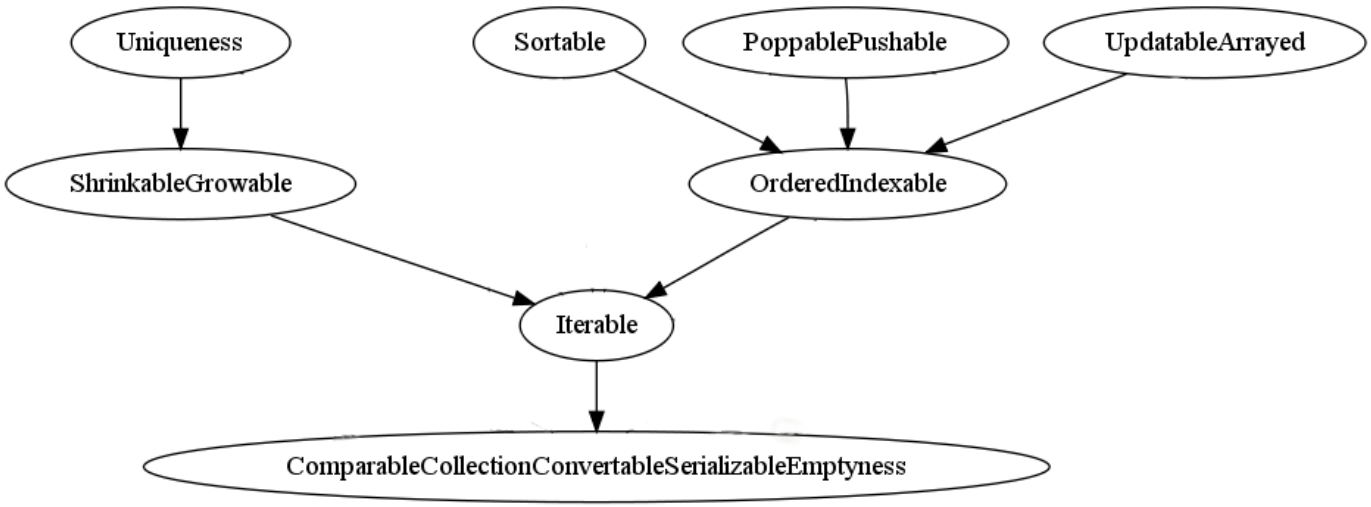
Algorithme de détermination de hiérarchie minimale

3 étapes

1. Créer des protocoles selon les associations assimilées identiques
2. Ordonner les protocoles selon leur relation de couverture ($\prec$)
   $E_n = \{$ Classe $\mid$ Classe utilise n $\}$
   $p \prec p'$ ssi $E_p \subset E_{p'}$
3. Pour chaque protocole, privilégier les liens indirects

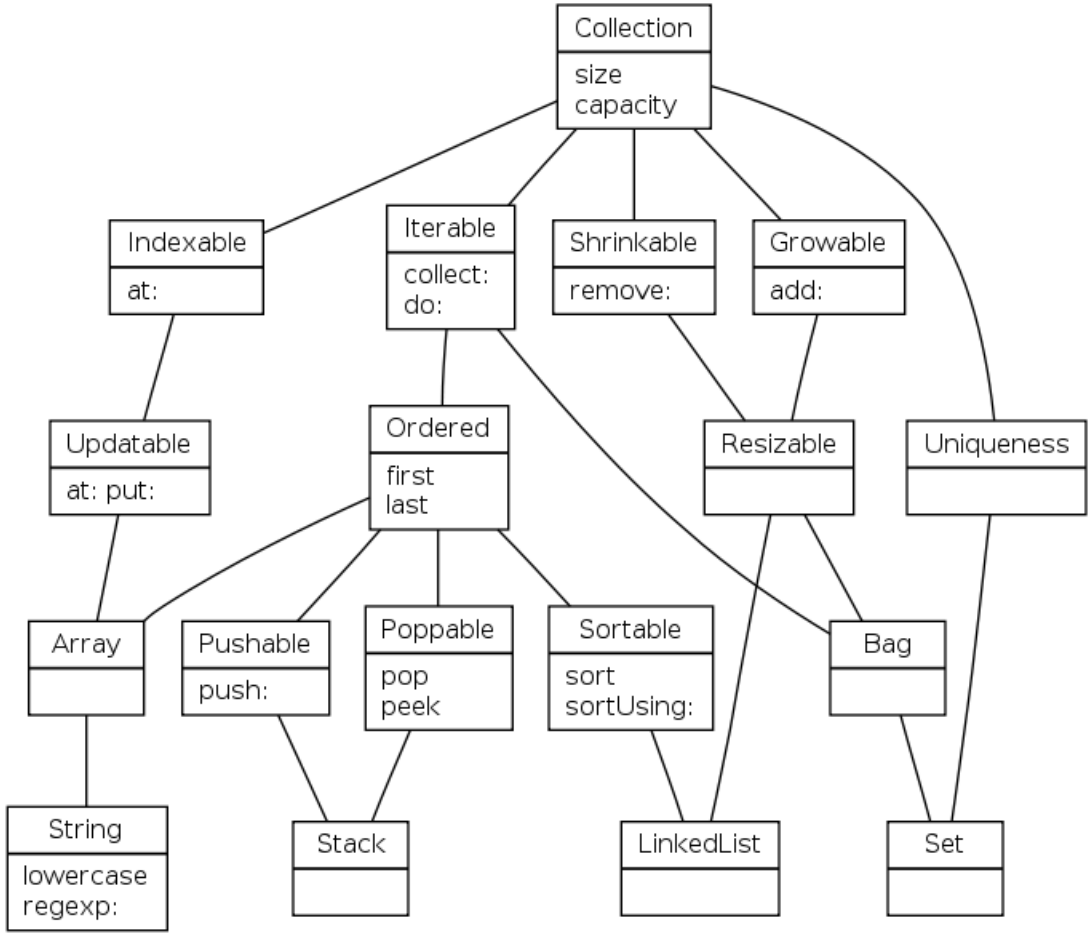
Méthode, étape 2 (2)

Algorithme de détermination de hiérarchie minimale



Méthode, étape 3

Affinage de la hiérarchie de protocoles générée



Synthèse

Modèle permettant de refactoriser les collections

- Unification des vues structurelles
  - Règle les problèmes liés à l'héritage simple
  - Améliore la compréhension

Synthèse

Modèle permettant de refactoriser les collections

- Unification des vues structurelles
  - Règle les problèmes liés à l'héritage simple
  - Améliore la compréhension
- Traits-protocoles
  - Gestion de l'hétérogénéité
  - Amélioration de la consistance polymorphique et sémantique

Synthèse

Modèle permettant de refactoriser les collections

- Unification des vues structurelles
  - Règle les problèmes liés à l'héritage simple
  - Améliore la compréhension
- Traits-protocoles
  - Gestion de l'hétérogénéité
  - Amélioration de la consistance polymorphique et sémantique
- Convertisseurs et protocoles utilisateurs
  - Pollution d'interfaces

Synthèse

Modèle permettant de refactoriser les collections

- Unification des vues structurelles
  - Règle les problèmes liés à l'héritage simple
  - Améliore la compréhension
- Traits-protocoles
  - Gestion de l'hétérogénéité
  - Amélioration de la consistance polymorphique et sémantique
- Convertisseurs et protocoles utilisateurs
  - Pollution d'interfaces
- Amélioration des protocoles
  - Meilleure expressivité
  - Outillage de détermination de granularité

Perspectives

- Mettre en oeuvre la refactorisation
  - Post-doc

Perspectives

- Mettre en oeuvre la refactorisation
  - Post-doc
- Identifier des patrons
  - Pour les traits
  - Pour les collections

Perspectives

- Mettre en oeuvre la refactorisation
  - Post-doc
- Identifier des patrons
  - Pour les traits
  - Pour les collections
- Traits dynamiques

Questions ?

Merci pour votre attention,

Des questions ?