La Réflexion
dans les langages à objets

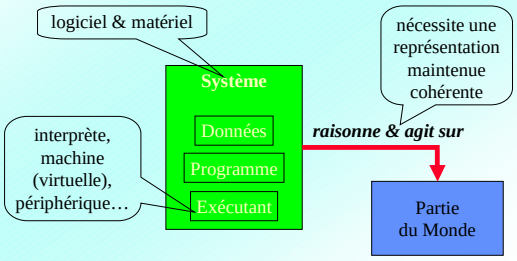
Noury Bouraqadi
http://csl.ensm-douai.fr/noury
Ecole des Mines de Douai

Caen - 14 Janvier 2003

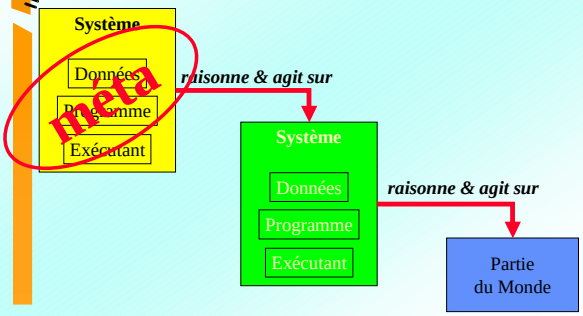
Plan

1. **Définition de la réflexion**
   - Dans un contexte objet
2. **MetaclassTalk**
   - Extension réflexive de Smalltalk
3. **Héritage à base de Mixins**
   - Exemple d'extension de langage avec la réflexion
4. **Conclusion**

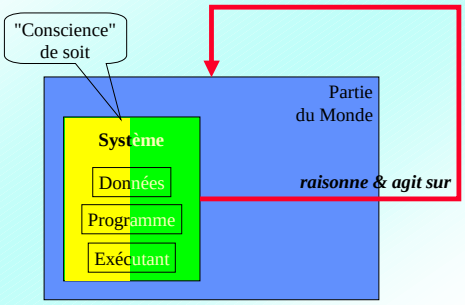
Structure d'un système informatique



Méta-système



Système réflexif



Définitions

- **Réflexion : Capacité d'un système à**
  - s'auto-observer
    - Raisonner sur soit même
  - s'auto-modifier
    - Altérer sa structure (programme)
    - Altérer son comportement (exécutant)
- **Réifier ("chosifier"): Représenter sous forme explicite**
  - les éléments constituants le programme et l'exécutant
- **Langage réflexif**
  - Langage dont les constructions & "l'interprète" sont réifiés

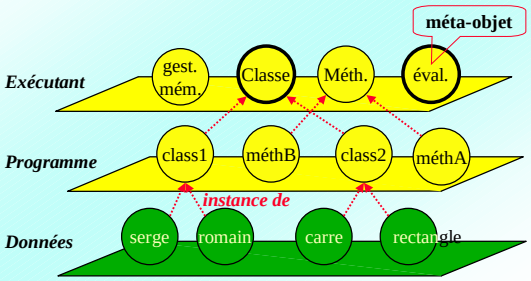
Système dans un langage à objets

- **Exécutant = machine (virtuelle), interprète, …**
  - sémantique du langage
  - chargement/compilation de programme
  - gestion mémoire
  - …
- **Programme**
  - classes, méthodes, champs, messages …
- **Données**
  - objets : banque, clients, comptes, …

+ ou – réifiés

Degré de réflexion

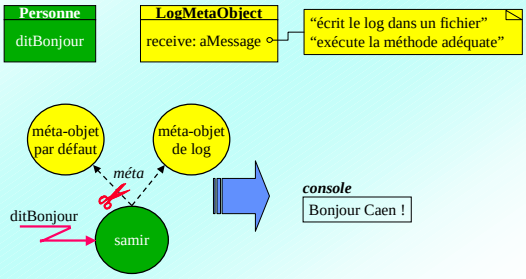
Système dans un langage réflexif à objets



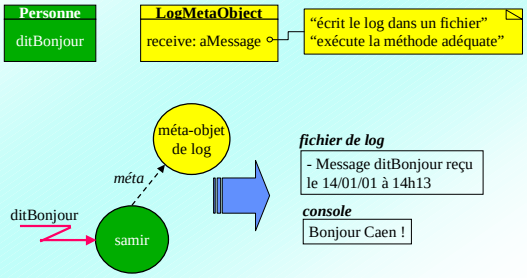
Langage réflexif à objets

- **Deux niveaux de programmation**
  - programme
  - exécutant
- **Chaque objet est instance d'une classe**
  - les classes sont aussi des objets
  - donc instances d'autres classes : les **méta-classes**
- **Exécution à la charge d'un méta-objet (évaluateur)**
  - le méta-objet est un objet
  - contrôle les messages, les accès aux champs, …
  - possibilité de lier un objet à un méta-objet particulier

Exemple de méta-objet - 1



Exemple de méta-objet - 2



Intérêts d'un langage réflexif

- **Développement d'outils de programmation**
  - Navigateur, Débogueur, Générateurs de code, …
- **Flexibilité à l'exécution**
  - Qualité de service, Evolutions non-prévues, …
- **Adapter et étendre le langage**
  - Héritage multiple, Communication asynchrone, Allocation paresseuse…
- **Simplifier le développement**
  - Code générique (donc plus réutilisable)
  - Séparer les codes fonctionnels et non-fonctionnels

Plan

1. *Définition de la réflexion*
   - *Dans un contexte objet*
2. ***MetaclassTalk***
   - *Extension réflexive de Smalltalk*
3. *Héritage à base de Mixins*
   - *Exemple d'extension de langage avec la réflexion*
4. *Conclusion*

Pourquoi Smalltalk ?

- ***Simple : permet de focaliser sur les concepts et la conception***
  - *tout est objets : y compris les classes*
  - *1 action = 1 message (y compris la création d'objet)*
  - *Syntaxe simple (sur 1 carte postale !)*
  - *Pas de typage = typage dynamique = pas de "cast"*
- ***Puissant***
  - *Bibliothèque riche copié (collections, streams, …)*
  - *Nombreuses capacités réflexives*
  - *Extensible : écrit en lui-même (y compris outils de dév.)*
  - *Outils de dév. puissants (Refactoring Browser, SUnit…)*

Modèle objet de Smalltalk

- ******Tout*** est objet***
  - *Pas de types primitifs !*
  - *Manipulation de références d'objets*
  - *Communication par messages*
- ***Champs = Variables d'instances***
  - *"private" à l'objet!*
  - *visibles dans les sous-classes*
- ***Méthodes***
  - *toutes "public"*
  - *toutes retournent une valeur*
- ***Héritage simple***
- ***Machine virtuelle***
- ***Garbage collector***

Syntaxe Smalltalk

- ***Mots réservés***
  - ***nil*** *référence nulle*
  - ***true***, ***false*** *objets booléens*
  - ***self*** *(l'objet lui-même)* ***super*** *(l'objet dans le contexte de la superclasse)*
  - ***thisContext*** *partie de la pile d'exécution représentant la méthode courante*
- ***Caractères réservés***
  - ***:=*** *(ou ←) affectation*
  - ***^*** *(ou ↑) = return*
  - ***\|*** *varTemp1 varTemp2 varTemp3* ***\|***
  - ***$*** *caractère*
  - ***#*** *literal*
  - ***.*** *termine toute expression*
  - *; cascade de messages*
  - ***( )*** *précedence*
  - ***[ ]*** *bloc of code*
  - ***"*** *commentaire****"***
  - ***'*** *chaine de caractère* ***'***

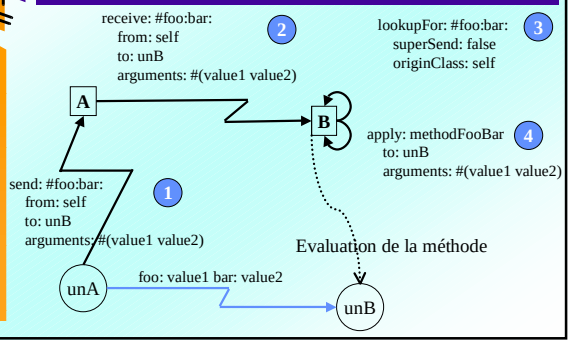
Pourquoi étendre Smalltalk ?

- ***Difficile de définir de nouvelles sortes de classes***
  - *Métaclasses implicites*
- ***Difficile de changer l'évaluateur***
- ***pas d'API ou framework simple pour les extensions***

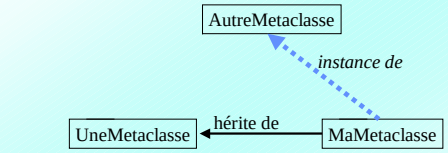
Qu'est-ce que MetaclassTalk

- ***Metaclass****es explicites* ***+*** *Small****Talk = MetaclassTalk***
  - *nouvelles sortes de classes*
  - *nouvelles constructions dans le langage*
- ***Etendre le processus d'évaluation des programmes***
  - *création d'objets (allocation mémoire)*
  - *lecture/écriture des champs*
  - *envoi/réception de messages*
  - *choix des méthodes à évaluer (héritage)*
  - *évaluation des méthodes*
- ***Par défaut la classe = méta-objet pour ses instances***

Décomposition d'une communication

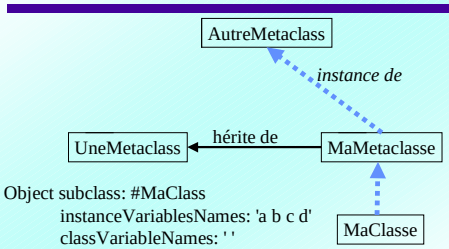


Comment utiliser MetaclassTalk ? - 1



UneMetaclasse subclass: #MaMetaclasse
instanceVariablesNames: 'x y'
classVariableNames: ' '
poolDictionaries: ' '
category: 'MetaclassTalk-Example'
metaclass: AutreMetaclasse

Comment utiliser MetaclassTalk ? - 2



Object subclass: #MaClass
instanceVariablesNames: 'a b c d'
classVariableNames: ' '
poolDictionaries: ' '
category: 'MetaclassTalk-Example'
metaclass: MaMetaclasse

Quelques applications de MetaclassTalk

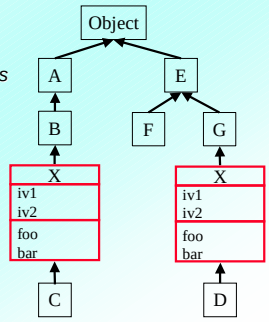
	Allocate	New	Read	Write	Send	Receive	Lookup	Apply
Abstract Class		x					x	
Asynchronous Communication					x	x		
Break Point			x	x		x		
Lazy Memory Allocation	x		x	x				
Message Caching						x		
Multiple Inheritance							x	
Persistent Objects (Data Base)	x		x	x				
Sole Instance (Pattern)		x						
Subject Class (Observer Pattern)				x				x
Synchronisation			x	x				x
Logging			x	x	x	x		
Type Checking							x	

Plan

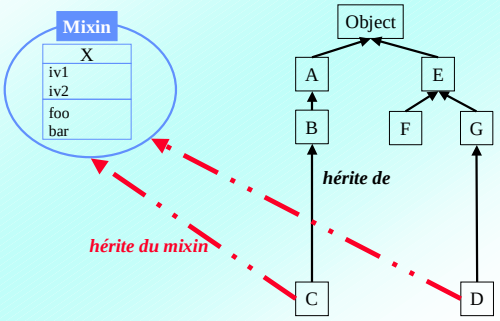
1. ***Définition de la réflexion***
   - *Dans un contexte objet*
2. ***MetaclassTalk***
   - *Extension réflexive de Smalltalk*
3. ***Héritage à base de Mixins***
   - *Exemple d'extension de langage avec la réflexion*
4. ***Conclusion***

Qu'est-ce que les Mixins ?

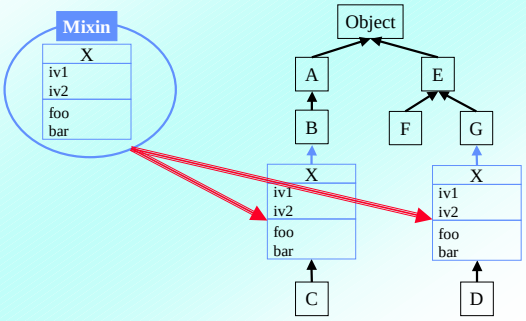
- ***Contexte :***
  - *Héritage simple*
  - *Hiérarchies indépendantes*
  - *Propriétés identiques*
- ***Problème :***
  - *Duplication de code*
- ***Solution = Mixin***
  - *Générateur de classes*
  - *[Bracha 90]*



Fonctionnement des mixins - 1



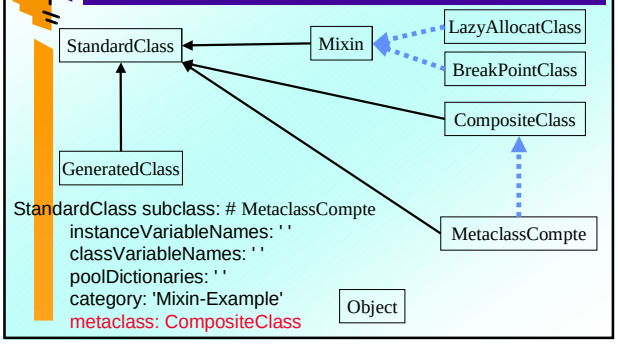
Fonctionnement des mixins - 2



Implanter des mixins avec MetaclassTalk

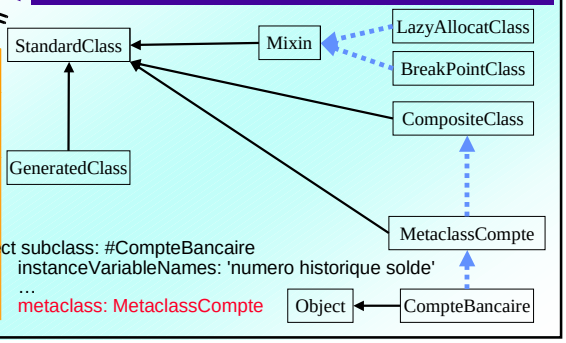
- ***3 méta-classes :***
  - *Mixin*
  - *GeneratedClass : Classes générées par les mixins*
  - *CompositeClass : Classes qui "héritent" des mixins*
- ***Mise à jour des classes générées si modif. mixin***
- ***Recours aux capacités réflexives de Smalltalk:***
  - *Compilation des méthodes*
  - *Construction des classes*
  - *Ajout/suppression de méthodes et champs à l'exécution*

Exemple de Méta-classes Mixins



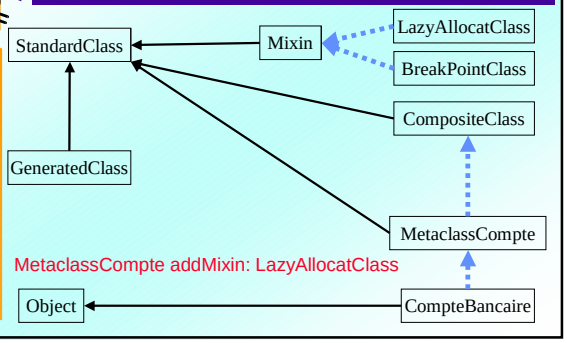
StandardClass subclass: # MetaclassCompte
instanceVariableNames: ' '
classVariableNames: ' '
poolDictionaries: ' '
category: 'Mixin-Example'
metaclass: CompositeClass

Exemple de Méta-classes Mixins



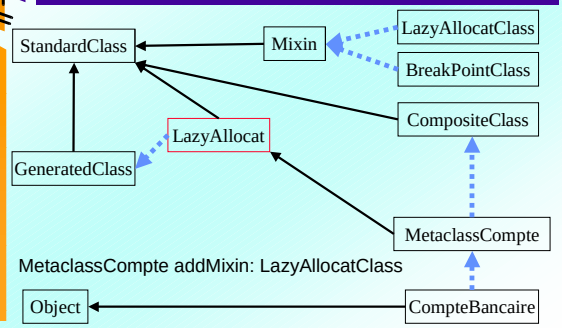
Object subclass: #CompteBancaire
instanceVariableNames: 'numero historique solde'
…
metaclass: MetaclassCompte

Exemple de Méta-classes Mixins



MetaclassCompte addMixin: LazyAllocatClass

Exemple de Méta-classes Mixins



MetaclassCompte addMixin: LazyAllocatClass

Exemple de Méta-classes Mixins

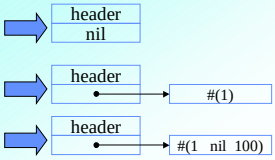
```
StandardClass subclass: #MetaclassCompte
    instanceVariableNames: ' '
    classVariableNames: ' '
    poolDictionaries: ' '
    category: 'MetaclassTalk-Example'
    metaclass: CompositeClass.

MetaclassCompte mixins: {LazyAllocatClass}
```


Exemple de Méta-classes Mixins

```
Object subclass: #CompteBancaire
    instanceVariableNames: 'numero historique solde'
    classVariableNames: ' '
    poolDictionaries: ' '
    category: 'MetaclassTalk-Example'
    metaclass: MetaclassCompte.
```

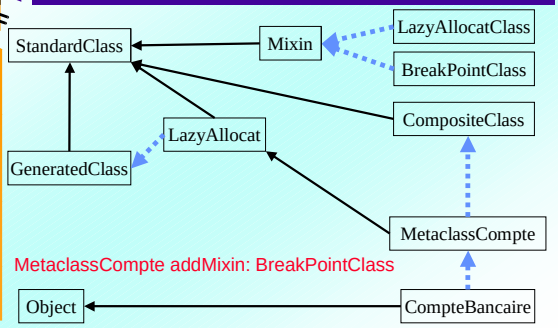
|compte|
compte := CompteBancaire new.



compte numero: 1.

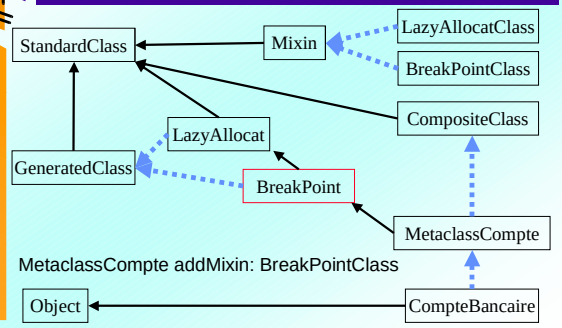
compte solde: 100.

Exemple de Méta-classes Mixins



MetaclassCompte addMixin: BreakPointClass

Exemple de Méta-classes Mixins



MetaclassCompte addMixin: BreakPointClass

Exemple de Méta-classes Mixins

```
StandardClass subclass: #MetaclassCompte
    …
    metaclass: CompositeClass.

MetaclassCompte mixins:
                    {LazyAllocatClass, BreakPointClass}
```

```
Object subclass: #CompteBancaire
    instanceVariableNames: 'numero historique solde'
    …
    metaclass: MetaclassCompte.

CompteBancaire breakOnSelectors: #( ) andIVs: #( )
```

Plan

1. *Définition de la réflexion*
   - *Dans un contexte objet*
2. *MetaclassTalk*
   - *Extension réflexive de Smalltalk*
3. *Héritage à base de Mixins*
   - *Exemple d'extension de langage avec la réflexion*
4. ***Conclusion***

Bilan

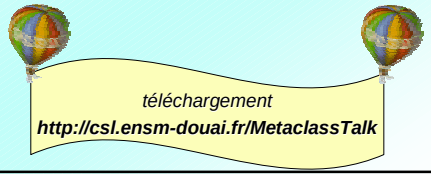
- ***La réflexion est utile***
  - *outils, extensions, qualité de service, simplifier le développement*
- ***Exemple d'extension du langage***
  - *Héritage à base Mixin*
  - *Utilisation au niveau méta*
  - *D'autres extensions sont possibles*
- ***Implantation disponible sous Squeak 3.2 (Smalltalk libre)***

Perspectives

- ***Différentes extensions du langage***
  - *Agents*
  - *Composants logiciels*
- ***Résoudre les problèmes de performance***
  - *code inlining*
  - *Etendre la machine virtuelle*
- ***Applications***
  - *Systèmes distribués*
  - *Systèmes embarqués*

Merci pour votre attention

Questions? Commentaires?



téléchargement

http://csl.ensm-douai.fr/MetaclassTalk