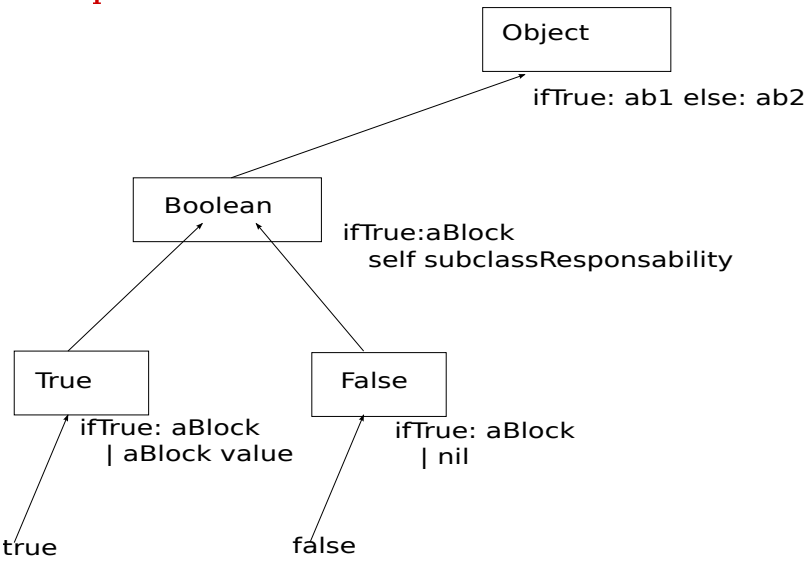
Questions

- Java est il entièrement objet ? Non
- Une classe est elle un objet en java ? Non
- La définition d'une classe est-elle un service dans un LO ? Oui
- Un programme est-il un objet dans un LO ? Oui
- Les instructions de contrôle existent-elles dans un LO ? Non

Exemple



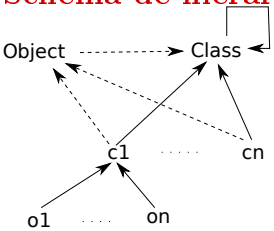
Versions possibles : 2.5, Pharo, Cincom

Plan du cours :

- Smalltalk : langage objet

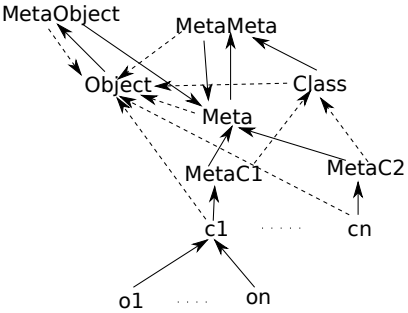
- Langage tout objet
- Autres caractéristiques (MVC, générateur d'interfaces, objets dépendants, gestion d'erreu, processus)
- Travaux pratiques

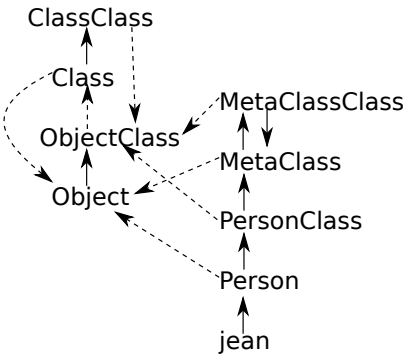
Schéma de hiérarchie de smalltalk 76 :



Problème : Toutes les classes ont le même comportement (ex : le même constructeur)

Schéma de hiérarchie de smalltalk 80 :





Cas d'utilisation de smalltalk : Tout domaine nécessitant de l'objet, sans contrainte de temps d'exécution.

Biblio :

- Smalltalk-80 The language and its implementation
- Inside Smalltalk
- SmallTalk with style
- SmallTalk by Example

Conventions d'écriture

- Les noms de classe et les variables globales commencent par une majuscule (cf Smalltalk inspect. en eyetreeInspector pour voir la liste de toutes les classes).
- Les noms des variables d'instance et des méthodes commencent par une minuscule.
- les caractères sont préfixés par $
- Les chaînes de caractère sont entre simple quotes
- Les commentaires sont entre guillemets
- Les tableau #(el1 el2 ...)
- Les symboles sont préfixés par #

Concepts de base

- Tout objet est instance unique d'une classe (pas de prototype : il faut créer la classe en premier, et ensuite on ne peut pas changer de classe).
- Encapsulation des données : les variables d'instance sont protected.
- Les variables sont non typés (on cherche la méthode d'après la classe de l'objet référencé par la variable et non d'après le type de la variable).

- Liaison purement dynamique.
- self = pseudo variable désignant le receveur du message, utilisable uniquement dans les méthodes et obligatoire pour appeler une autre méthode de la même classe.
- super = pseudo variable désignant le receveur du message, utilisable uniquement dans les méthodes, indiquant que la recherche de la méthode doit se faire à partir de la surclasse de la classe qui implémente la méthode.
- Envoi de message < objet receveur > < message >. Tout envoi de message retourne un objet (l'objet receveur si pas précisé dans la définition de la méthode).
- Il n'y a pas de structure de contrôle (uniquement des envois de messages)
- Comportement différent ⇒ Classes différentes

3 types de messages

- unaire sans paramètre
- binaire 1 ou 2 caractères non alphanumériques, 1 seul argument
- "keywords" au moins 1 argument

Priorité : Parenthèses, unaires, binaires, keywords

Services de contrôle :

- Conditionnelle : cas particulier de ifClasse

- Itérations :
  - whileTrue, whileFalse :

    whileTrue: aBlock

    self value

    ifTrue: [

    aBlock value.

    self whileTrue: aBlock]
  - 1 to: 10 do: [:i | x := x + i]

Idée générale

Idée implicite Les vues dépendent du modèle $\Rightarrow$ notion d'objets dépendants.

En smallTalk

- La classe ApplicationModel
- La classe Dialog
- La classe TextCollector et TextCollectorView
- Attention à l'indépendance Vue/Modèle