



Premiers pas en Smalltalk

Illustration avec un Smalltalk libre: Pharo

Noury Bouraqadi
<http://car.mines-douai.fr/noury>
Option ISIC
Ecole des Mines de Douai



Avertissement

"Smalltalk is dangerous.

It is a **drug**.

My advice to you would be **don't try it**.
It could ruin your life."

Andy Bower

CEO

Object Arts Ltd.

Pourquoi Smalltalk?

- Parce que programmer avec un autre langage c'est comme se frotter le dos contre un cactus
 - Au début ça pique
 - Après on peut s'habituer
 - On peut même y prendre du plaisir
 - Mais, au final cela reste du masochisme

Citation reprise et adaptée de

Sébastien Dupire développeur Java/J2EE

dans son billet "Un an dans le monde Java"

du 15/01/2013

<http://sebastien-dupire.info/un-an-dans-le-monde-java.html>

Pourquoi Smalltalk ?

- Puissance au service des programmeurs
 - Syntaxe et concepts réduits au minimum
 - Minimise le "bruit"
 - Capacités pour analyser et modifier les programmes "in vivo"
 - Infrastructure réflexive
 - Outils comme les inspecteurs
- Très différent de Java
 - Syntaxe, typage, environnement, philosophie...
 - Proche d'Objective-C
 - Mais, plus simple

Pourquoi Smalltalk ?

"Smalltalk is a programming language **focused on human beings** rather than the computer"

Alan Knight
Engineering Manager
Cincom Systems

Noury Bouraqadi - option ISIC - Dépt. I. A.

Historique

- Naissance au prestigieux Xerox PARC
- Un langage rôdé
 - première version en 1972
 - Utilisation industrielle à partir de 1980
- Premier langage à objets moderne
 - Utilisation industrielle encore aujourd'hui
 - Outils de développement en équipe
 - (depuis les années 80 !)
- Squeak : Première version libre en 1997
 - Finalité principalement éducative et expérimentale
 - Avec des applications industrielles
- Pharo : Version libre Issue de Squeak en 2008
 - Finalité principalement industrielle

Noury Bouraqadi - option ISIC - Dépt. I. A.

Un langage simple à apprendre - 1

- Syntaxe simple (reprise dans Objective-C == Mac & iPhone)
 - proche du langage naturel
 - mots et caractères spéciaux = 1 transparent
 - permet de focaliser sur les concepts et la conception
- Typage dynamique
 - pas de type à saisir
 - contrôle de type à l'exécution
- Tous les sources sont disponibles
 - y compris les bibliothèques de base

Noury Bouraqadi - option ISIC - Dépt. I. A.

Un langage simple à apprendre - 2

- Très peu de concepts
 - Objet, Classe, Message, Variable
 - Toute méthode retourne une valeur
- Règles de visibilité fixes
 - Classes et Méthodes accessibles par tout le monde
 - Champs accessibles uniquement par leur propriétaire
- Uniformité (sans exception)
 - Tout est objet
 - nombres, outils de dev., le compilateur, ...
 - 1 action = 1 message
 - y compris la création d'objet

Noury Bouraqadi - option ISIC - Dépt. I. A.

Un langage très puissant - 1

- Bibliothèque riche copiée
 - collections, streams, ...
- Langage ouvert (capacités réflexives)
 - Nombreux éléments/concepts du langage accessibles
 - Langage : Classe, méthode, ...
 - Exécution : Pile d'exécution, Contrôle de type, ...
 - Outils : Compilateur, Débogueur, ...
 - Extensible
 - Code éléments du langage disponible
 - Possibilité de le modifier/étendre

Noury Bouraqadi - option ISIC - Dépt. I. A.

Un langage très puissant – 2

Quelques outils usuels de Smalltalk

- Navigation/Codage
 - Browsers: System, Hierarchy, Class, Category, Refactoring, ...
 - method Implementors, Method Finder
 - message Senders
- Débogue
 - Debugger
 - Inspector
 - Test Runner (SUnit)
 - Workspace
- Gestion de code
 - Project
 - Change Set, Change Sorter
 - Gestion de version automatique
 - fileIn/fileOut, FileList
- ...

Noury Bouraqadi - option ISIC - Dépt. I. A.

Syntaxe Smalltalk

- Mots réservés
 - **nil** objet indéfini (valeur par défaut des champs et variables)
 - **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
 - := affectation
 - ^ return
 - | varTemp1 varTemp2 varTemp3 |
 - \$ caractère
 - # littéral
 - . termine toute expression
 - ; cascade de messages
 - () précedence
 - [] bloc de code
 - " commentaire"
 - ' chaîne de caractères '

Noury Bouraqadi - option ISIC - Dépt. I. A.

L'environnement de développement Smalltalk

Noury Bouraqadi - option ISIC - Dépt. I. A.

Modèle d'exécution Smalltalk

- 4 fichiers fondamentaux
 - Image mémoire (.image)
 - changements effectués (.change)
 - code source des bibliothèques de base (.source)
 - Machine virtuelle (.exe)
- Gestion automatique de la mémoire
 - Garbage collector (Ramasse Miettes)
- Tout se passe à l'exécution
 - Développement incrémental
 - Sauvegarde automatique des changements (.change)
 - Perte de code quasi-impossible

} Dupliqués pour
Chaque projet

Noury Bouraqadi - option SIC - Dépt. I. A.

4 fichiers fondamentaux

- Machine virtuelle (.exe)
 - VM (Virtual Machine)
- Fichier sources standards (.sources)
 - Texte
- Fichier image mémoire (.image)
 - Binaire
- Fichier des changements (.change)
 - Texte
 - Code source des changements effectués

Noury Bouraqadi - option SIC - Dépt. I. A.

Fichiers fondamentaux - 1

- Machine virtuelle (.exe)
 - VM (Virtual Machine)
 - Spécifique à une plateforme
 - Mac, Linux, Windows, ...

Noury Bouraqadi - option SIC - Dépt. I. A.

Fichiers fondamentaux - 2

- Fichier sources standards (.sources)
 - Texte
 - Lecture seule

Noury Bouraqadi - option SIC - Dépt. I. A.

Fichiers fondamentaux - 3

- Fichier image mémoire (.image)
 - Binaire
 - Lecture automatique + enregistrement à la demande
 - "Photographie" de la mémoire à l'instant de la sauvegarde
 - Possibilité d'avoir plusieurs fichiers images

Noury Bouraqadi - option ISIC - Dépt. I. A.

Fichiers fondamentaux - 4

- Fichier des changements (.change)
 - Texte
 - Code source des changements effectués
 - 1 fichier .change par fichier image (même nom)
 - Enregistrement automatique
 - Sauvegarde de toutes les opérations réalisées !!
 - Lecture à la demande
 - Récupération manuelle en cas d'arrêt intempestif

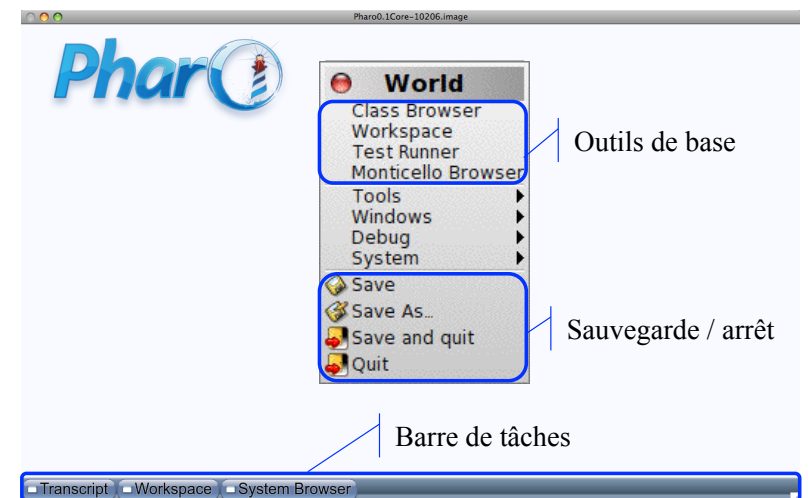
Noury Bouraqadi - option ISIC - Dépt. I. A.

Lancement de Pharo

- Glisser/Déposer
 - le fichier .image sur la machine virtuelle (.exe)
- Double-clic
 - sur le fichier .image
 - si l'extension .image est associée à la machine virtuelle
 - sur la machine virtuelle (.exe)
 - Ensuite, choix du fichier .image à lancer
 - Plus lent que les autres
 - Recherche de tous les fichiers .image sur la totalité du disque

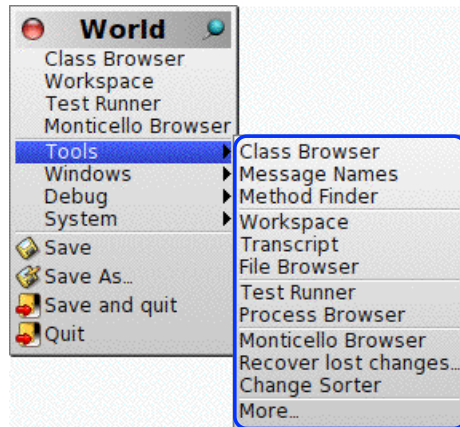
Noury Bouraqadi - option ISIC - Dépt. I. A.

Menu "World"



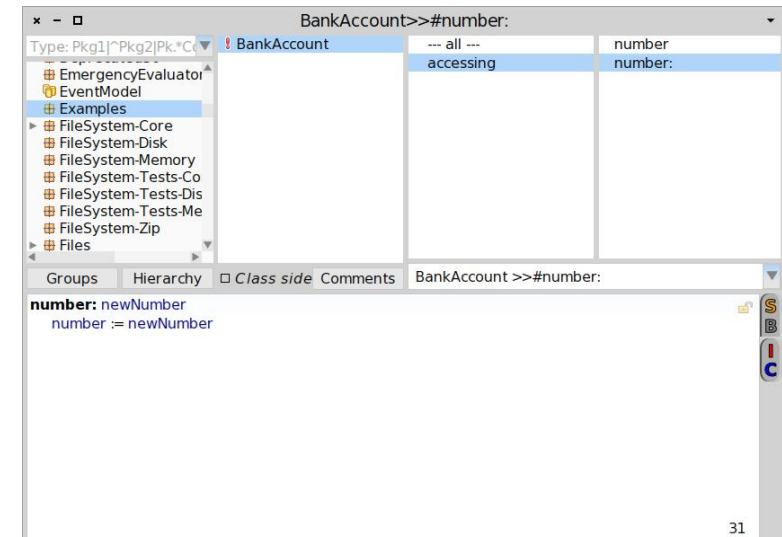
Noury Bouraqadi - option ISIC - Dépt. I. A.

Autres outils



Noury Bouraqadi - option ISIC - Dépt. I. A.

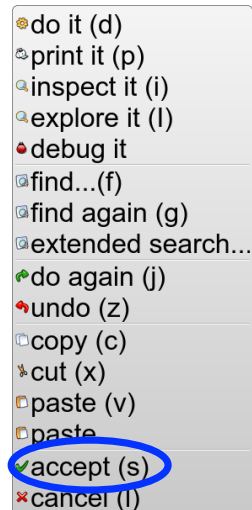
Browser - 1



Noury Bouraqadi - option ISIC - Dépt. I. A.

Browser - 2

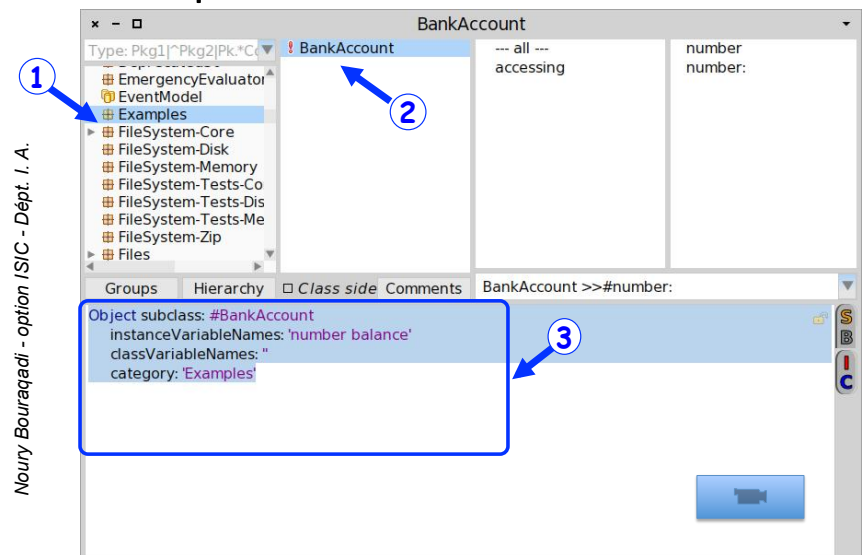
- Découpage logique
 - Catégories de classes
 - Catégories de méthodes
- Compilation
 - Transparente à la volée
 - Incrémentale
 - versions des méthodes



Noury Bouraqadi - option ISIC - Dépt. I. A.

- Utilisation du menu "accept"

Exemple de création d'une classe - 1



Noury Bouraqadi - option ISIC - Dépt. I. A.

Exemple de création d'une classe - 2

Object subclass: #BankAccount

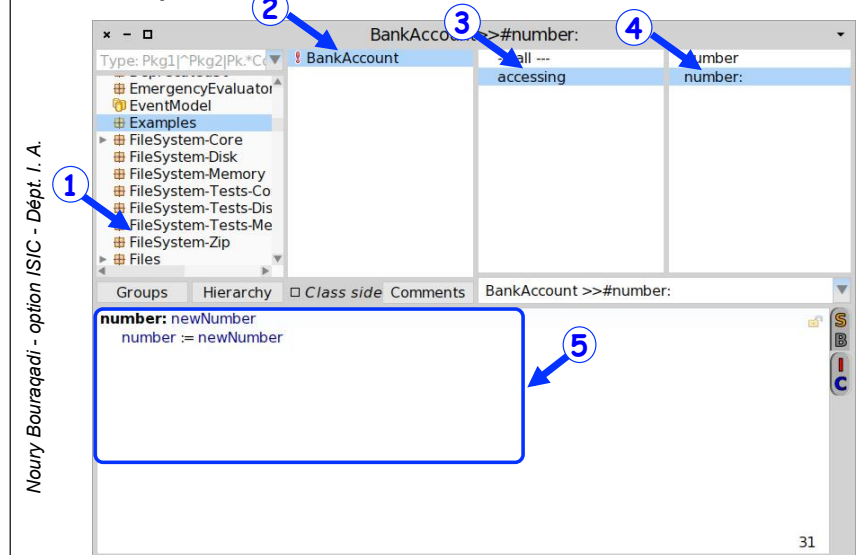
instanceVariableNames: 'number balance'

classVariableNames: 'CreatedAccounts'

category: 'Examples-Banking'

Noury Bouraqadi - option /SIC - Dépt. I. A.

Exemple de définition d'une méthode - 1



Noury Bouraqadi - option /SIC - Dépt. I. A.

Exemple de définition d'une méthode - 2

```
number: newNumber
```

```
number := newNumber
```

Noury Bouraqadi - option /SIC - Dépt. I. A.

Exemple de messages Smalltalk

```
|myAccount yourAccount|
```

```
"creates a new account"
```

```
yourAccount := BankAccount new.
```

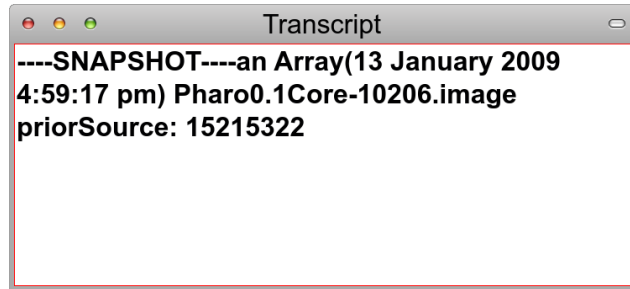
```
yourAccount deposit: 2000.
```

```
myAccount := BankAccount new.
```

```
yourAccount transfer: 1500 to: myAccount.
```

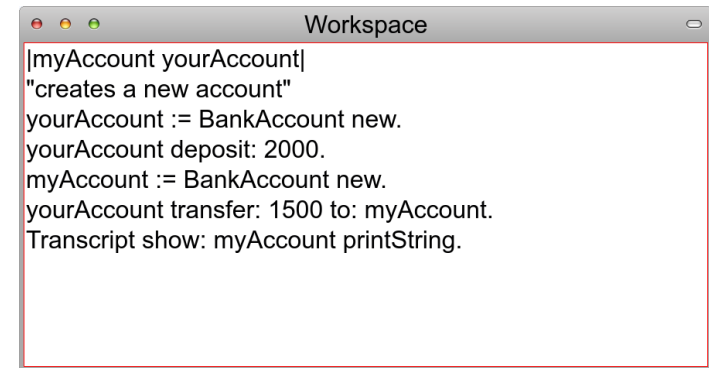
Noury Bouraqadi - option /SIC - Dépt. I. A.

Transcript = "Console"



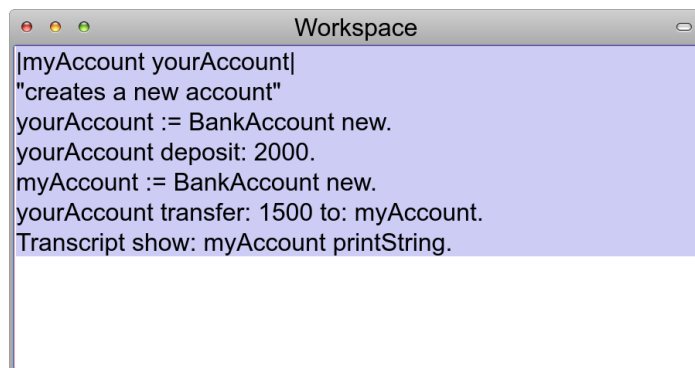
Noury Bouraqadi - option /SIC - Dépt. I. A.

Workspace : exécuter, afficher, inspecter...



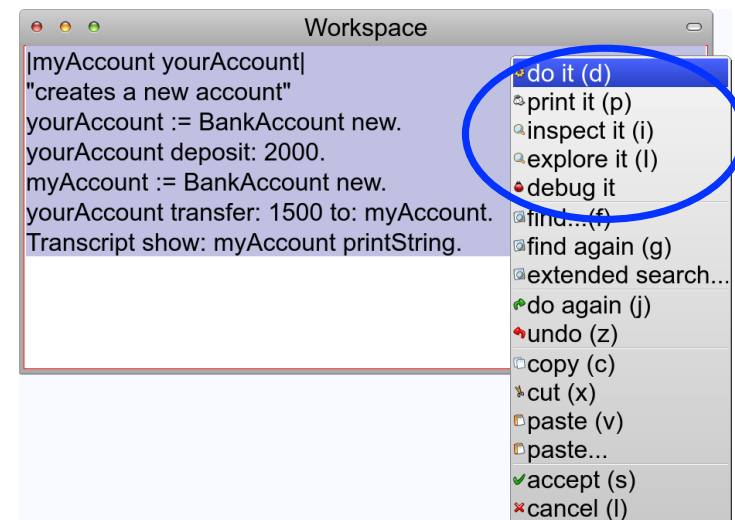
Noury Bouraqadi - option /SIC - Dépt. I. A.

Workspace : exécuter, afficher, inspecter...



Noury Bouraqadi - option /SIC - Dépt. I. A.

Workspace : exécuter, afficher, déboguer...

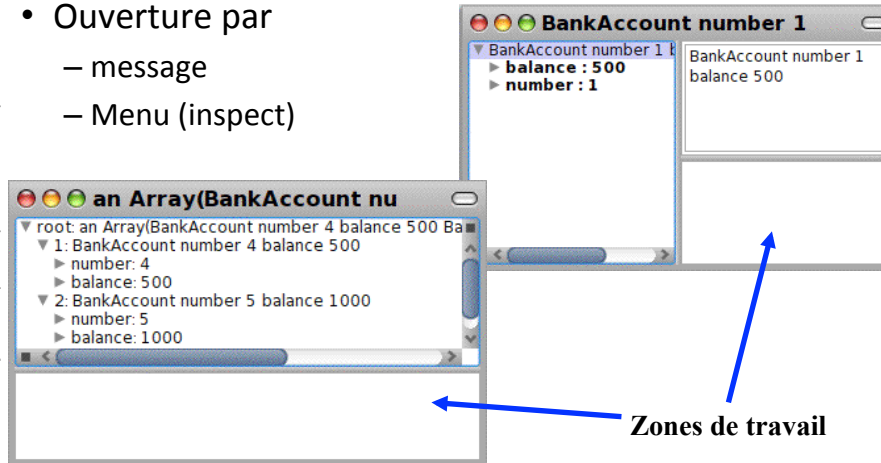


Noury Bouraqadi - option /SIC - Dépt. I. A.

Inspecteur & explorateur

- Donnent accès à la structure d'un objet
- Ouverture par
 - message
 - Menu (inspect)

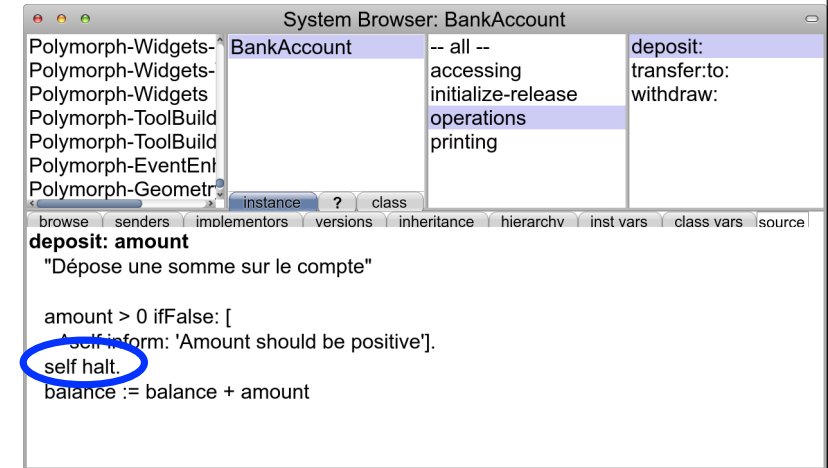
Noury Bouraqadi - option ISIC - Dépt. I. A.



Débogueur – Placer un point d'arrêt

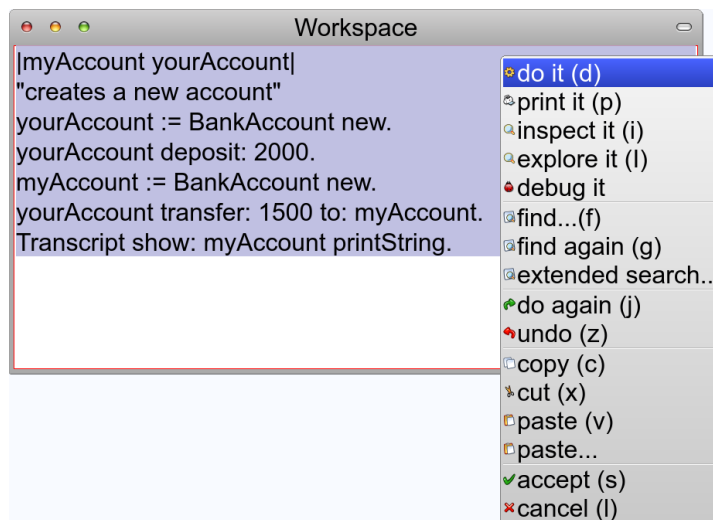
- Point d'arrêt = self halt

Noury Bouraqadi - option ISIC - Dépt. I. A.



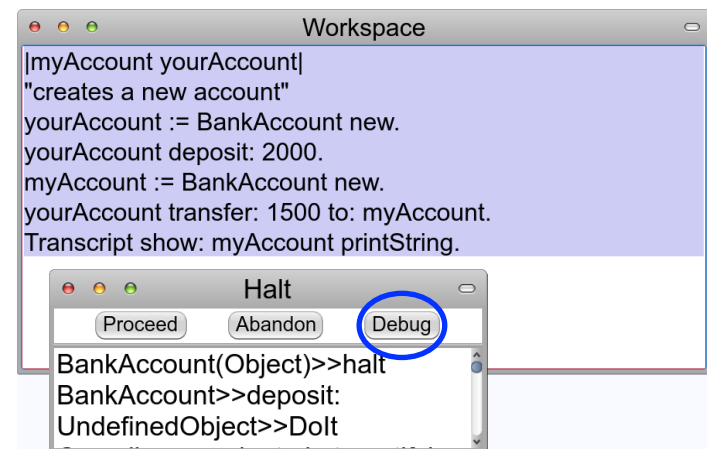
Débogueur– Ouverture (1)

Noury Bouraqadi - option ISIC - Dépt. I. A.

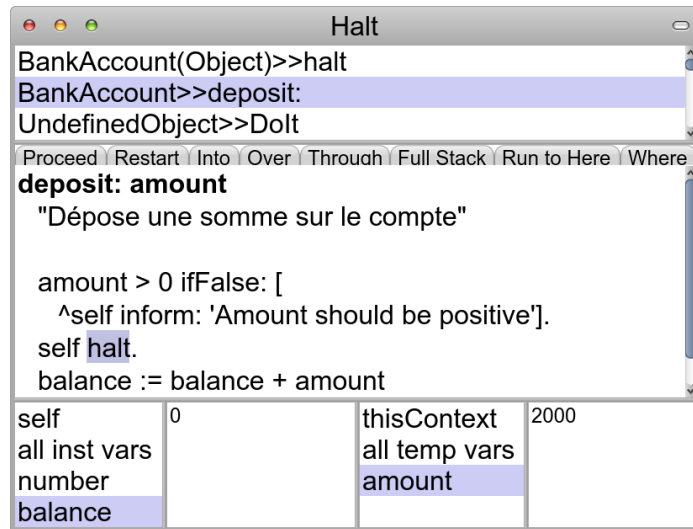


Débogueur– Ouverture (2)

Noury Bouraqadi - option ISIC - Dépt. I. A.



Le Débogueur !



Noury Bouraqadi - option /SIC - Dépt. I. A.

Éléments de base de Smalltalk

Noury Bouraqadi - option /SIC - Dépt. I. A.

Envoi de message

- Receveur = destinataire d'un message
- Toute méthode retourne un résultat
 - Par défaut c'est l'objet lui-même (self)
 - Si la méthode ne se termine pas par un retour
 - le compilateur insère **^self** automatiquement
- Cascade de messages
 - Plusieurs messages destinés au même receveur
 - Receveur écrit une seule fois
 - Messages séparés par des point-virgules
 - compte deposit: 200; deposit: 50; withdraw: 100

Noury Bouraqadi - option /SIC - Dépt. I. A.

Méthodes et messages - 1

- Différents types
 - Unaire
 - message sans argument
 - unObjet **printString**
 - Binaire
 - le sélecteur inclus un symbol non-alpha-numérique
 - 1 + 2
 - Avec (1 ou plusieurs) mot-clés
 - Set **new:** 100
 - votreCompte **transferer:** 50 **vers:** monCompte

Noury Bouraqadi - option /SIC - Dépt. I. A.


Méthodes et messages - 2

- Précédence :
 - de gauche à droite
 - messages unaires, puis binaires et enfin à mot-clé
- Sélecteur
 - identifiant unique
 - instance de la classe Symbol
 - Exemple de sélecteurs
 - size
 - +
 - new:
 - transferer:vers:

Noury Bouraqadi - option /SIC - Dépt. I. A.

Création d'instances initialisées

- Tous les champs d'un nouvel objet référencent nil
 - Besoin de leur donner une valeur par défaut
 - Nécessaire pour le bon fonctionnement de l'objet



couleur	<i>nil</i>
estAllumee	<i>nil</i>

Noury Bouraqadi - option /SIC - Dépt. I. A.


Initialisation des objets

- Définir une méthode pour chaque valeur par défaut
 - Retourne la valeur par défaut d'un champ donné
- Définir la méthode **initialize**
 - Obtenir les valeurs par défaut
 - envois de messages à **self**
 - Affecter les valeurs par défaut aux champs

Noury Bouraqadi - option /SIC - Dépt. I. A.

Une nouvelle lampe doit être éteinte & jaune

- Lampe>>estInitialementAllumee
^false
- Lampe>>couleurParDefaut
^Color yellow
- Lampe>>**initialize**
super initialize. "obligatoire ! cf. héritage"
estAllumee := **self estInitialementAllumee.**
couleur := **self couleurParDefaut**



couleur	<i>Color yellow</i>
estAllumee	<i>false</i>

Noury Bouraqadi - option /SIC - Dépt. I. A.

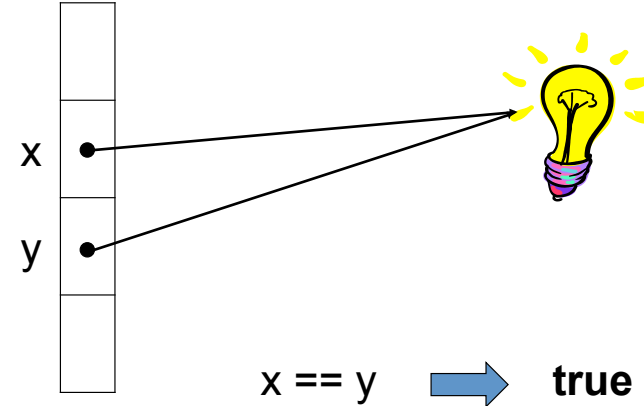
Quelque messages usuels

- Représentation textuelle
 - **printString**
 - Exemple : 1500 **printString** → '1500'
- Comparaison
 - Teste l'identité **==** Compare les **variables**
 - Teste l'égalité **=** Compare les **objets**
- Conditionnelles
 - **ifTrue: [] ifFalse: []**
- Boucles
 - **timesRepeat: []**
 - **[] whileTrue: []**

Noury Bouraqadi - option ISIC - Dépt. I. A.

Test d'identité ==

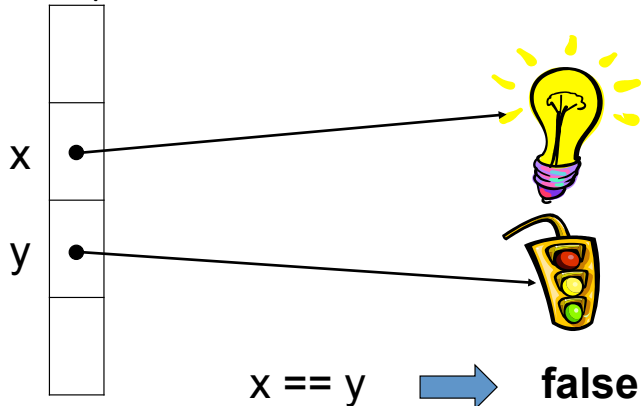
- Compare les variables



Noury Bouraqadi - option ISIC - Dépt. I. A.

Test d'identité ==

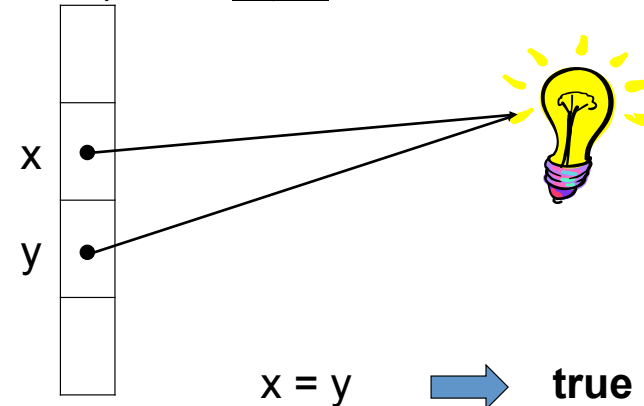
- Compare les variables



Noury Bouraqadi - option ISIC - Dépt. I. A.

Test d'égalité =

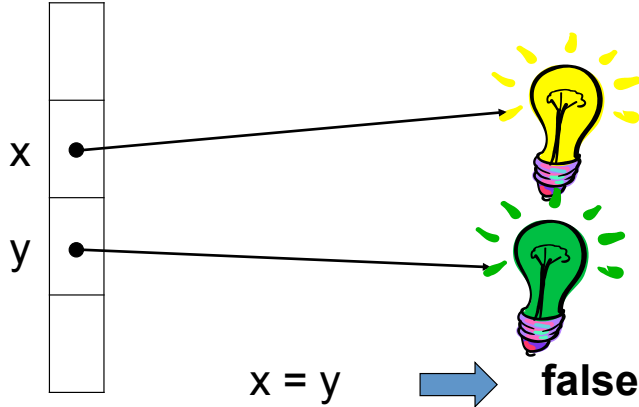
- Compare les objets



Noury Bouraqadi - option ISIC - Dépt. I. A.

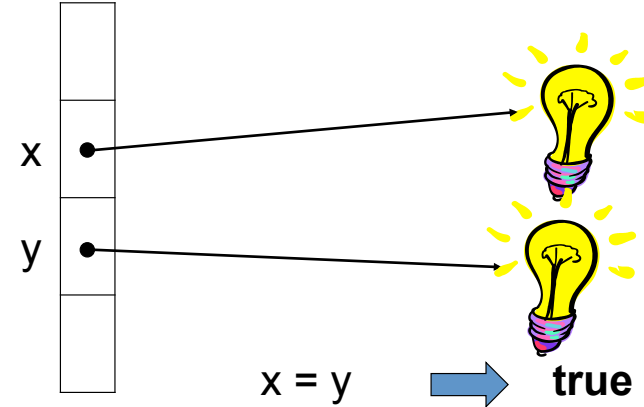
Test d'égalité =

- Compare les objets



Test d'égalité =

- Compare les objets



Test d'égalité =

- Les classes des objets comparés doivent définir :
 - La méthode `=`
 - La méthode `hash`
- Exemple avec Lampe
- `= autreLampe`

```
^(self couleur = autreLampe couleur) and: [
    self estAllumee = autreLampe estAllumee]
```
- `hash`

```
^(self couleur hash) bitXor: (self estAllumee hash)
```

Exemple de comparaisons

Expressions retournant "true"

`a == b. a = b. a = d.`

`b == a. b = a. b = d.`

`d = a. d = b.`

Expressions retournant "false"

`a == c. a = c. a == d.`

`b == c. b = c. b == d.`

`c == a. c = a. c = b. c == b. c = d.`

`c == d.`

`d == a. d == b. d == c. d = c.d`

Conditionnelle (le "if")

- Booléens = objets true et false
- Conditionnels = messages destinés aux booléens
- 3 formes = 3 méthodes
 - expressionBooléenne **ifTrue:** ["bloc de code"]
 - expressionBooléenne **ifFalse:** ["bloc de code"]
 - expressionBooléenne
 - ifTrue:** ["bloc de code"]
 - ifFalse:** ["bloc alternatif"]

Noury Bouraqadi - option /SIC - Dépt. I. A.

Boucles

- Boucles = Messages
- "for"
 - 1 **to:** 10 **do:** [:i | "traitement boucle"]
 - 15 **timesRepeat:** ["bloc de code répété"]
- "while"
 - ["expression booléenne"] **whileTrue:** ["bloc de code"]
 - ["expression booléenne"] **whileFalse:** ["bloc de code"]

Noury Bouraqadi - option /SIC - Dépt. I. A.

Biblio/Webo-graphie

Noury Bouraqadi - option /SIC - Dépt. I. A.

Livres disponibles au centre de Doc



Noury Bouraqadi - option /SIC - Dépt. I. A.

<http://stephane.ducasse.free.fr/FreeBooks.html>



LE Livre

Free
PDF

Pharo 1.0 !



<http://pharobyexample.org/fr/index.html>



LE Livre

Free
PDF

Pharo 2.0 !

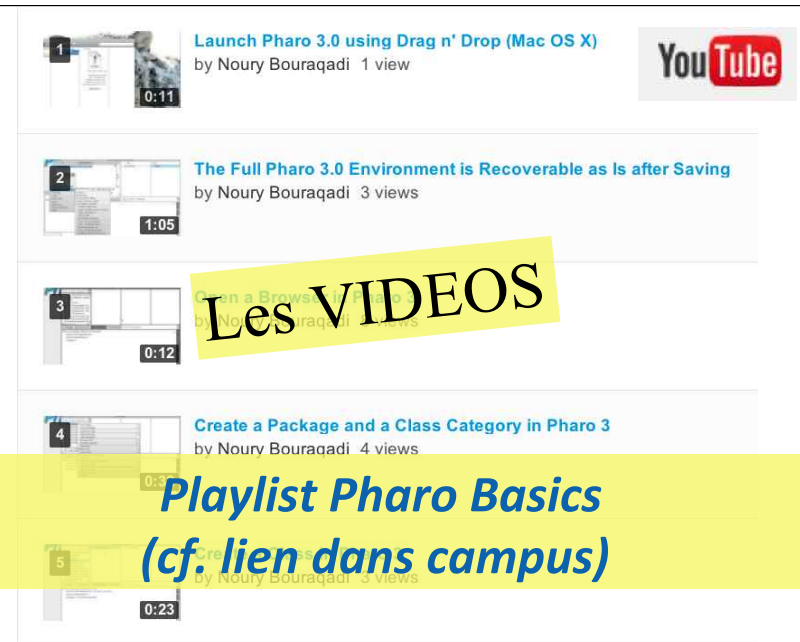


<http://deepintopharo.com/>



LE Site

<http://www.pharo.org/>



Les VIDEOS

Playlist Pharo Basics
(cf. lien dans campus)



Packages

- Graphics-Display Objects**
(Package Graphics-Display Objects)
- jQuery-Core-Objects**
(Package jQuery-Core-Objects)
- jQuery-UI-Objects**
(Package jQuery-UI-Objects)
- Kernel-BasicObjects**
(Package Kernel-BasicObjects)
- Kernel-Objects**
(Package Kernel-Objects)
- KernelTests-Objects**
(Package Kernel-Tests-Objects)
- NativeBoost-Code-Objects**
(Package NativeBoost-Code-Objects)
- System-Object Events**
(Package System-Object Events)

Classes

- Boolean** < Object
- DeepCopier** < Object
- False** < Boolean
- MessageSend** < Object
- Object** < ProtoObject
- ProtoObject** < nil

Object

Subclass of: ProtoObject@

inheritance diagram or package diagram

Overview

Object is the root class for almost all of the other classes in the class hierarchy. It provides default behavior common to all normal objects, such as access, copying, comparison, error handling, message sending, and reflection. Also utility messages that all objects should respond to are defined here.

Object has no instance variables, nor should any be added. This is due to several classes of objects that inherit from Object that have special implementations (SmallInteger and UndefinedObject for example) or the VM knows about and depends on the structure and layout of certain standard classes.

Message Variables: Dependence should be implemented by IdentityDictionary Provides a virtual news, synchronized by the changed /update: protocol. Note that class Model has a real slot for its dependents, and overrides the associated protocol with more efficient implementations. EventsFieldsan IdentityDictionary that maps each object to its dependents. Registers a message send (consisting of a selector and a receiver object) which should be performed when anEventSymbol is triggered by the receiver. Part of a

Doc Bibliothèques de base

<http://files.pharo.org/doc/>