Les Collections

Noury Bouraqadi
http://car.mines-douai.fr/noury

Option ISIC
Ecole des Mines de Douai

Collections

- **Collection**
  - Notion plus générale que "tableau"
  - Objet qui référence un nombre arbitraire d'objets
    - Ces objets sont appelés éléments de de la collection
- **Différentes classes de collections**
  - Collections ordonnées ou pas
  - Nombre d'éléments extensible ou fixe
  - Différentes façons d'accéder aux éléments
    - Par indice
    - Avec une clé (dictionnaires)
    - Par parcours seulement (ensembles)

Quelques classes de collections

- **Collections ordonnées**
  - Taille fixe
    - **Array**
  - Taille variable
    - **OrderedCollection**
    - **SortedCollection** : éléments automatiquement triés
- **Collections non-ordonnées (taille variable)**
  - Ensembles
    - **Set** : Deux éléments ne peuvent pas être égaux
    - **IdentitySet** : Un élément ne peut pas apparaître deux fois
  - Dictionnaires : table de correspondance clé-valeur
    - **Dictionary** : Deux clés ne peuvent pas être égales
    - **IdentityDictionary** : Une même clé ne peut être utilisée deux fois

Création de collection - 1

- **Message *new***
  - Exemples : Set new. OrderedCollection new.
- **Message new:** taille
  - Paramètre = taille (initiale) de la collection
  - Usage typique : tableaux (Array)
  - Exemple :

Array new: 5

Création de collection - 2

- **En fournissant les éléments d'un tableau (Array)**
  - {expression1. expression2. expression3}
    - Exemple :

    {Object new. 345 * 21. (1 to: 45 by: 5). 'Hello World'. 2014}
  - #(valeur1 valeur2 valeur3)
    - Exemple :

    #(45 'mines de douai' $A)
- **Tableau d'octets**
  - #[entier1 entier2 entier3 entier4]

Création de collection - 3

- **En fournissant les éléments d'un Dictionnaire**

  {clé1 -> expression1.
  clé2 -> expression2.
  clé3 -> expression3} **asDictionary**
  - Exemple :

  {#unObjet -> Object new.
  #nombre -> (345 * 21).
  #interval ->(1 to: 45 by: 5).
  #hello -> 'Hello World'.
  #annee -> 2014} asDictionary

Accès aux collections à taille variable (ordonnées ou non)

- **add: unObjet**
  - Ajoute unObjet à la collection
- **addAll: collection**
  - Ajoute tous les éléments de la collection
- **remove: unObjet ifAbsent: ["traitements"]**
  - Retire unObjet de la collection
  - Exécute le bloc de traitements si unObjet n'appartient pas à la collection
- **remove: unObjet**
  - Erreur si unObjet n'appartient pas à la collection
- **removeAll: collection**
  - Retire tous les éléments de la collection

Accès aux collections ordonnées (de taille fixe ou variable)

- **first**
  - Retourne le premier élément
- **last**
  - Retourne le dernier élément
- **at: indice**
  - Retourne l'élément dont la position correspond à indice
  - Erreur si indice est hors des limites de la collection
    - premier indice = 1
    - dernier indice = taille de la collection
- **at: indice put: unObjet**
  - Stocke unObjet à la cellule dont le numéro est indice

Parcours des collections (boucles) - 1

- **do: [:element| "traitements"]**
  - element = variable qui référence successivement les différents éléments de la collection
  - Les traitements définis dans le bloc sont exécutés successivement pour chaque élément de la collection.
- **collect: [:element| "traitements"]**
  - Retourne une collection du même type que celle parcourue
  - avec les résultats des traitements pour chaque élément

Parcours des collections (boucles) - 2

- **select: [:element| "expression booléenne"]**
  - Retourne les éléments pour lesquels l'expression booléenne est vraie (true)
- **reject: [:element | "expression booléenne "]**
  - inverse de select
- **reverseDo: unBlock**
  - Parcours la collection en ordre inverse
  - Pour les collections ordonnées seulement !

Parcours des collections (boucles) - 3

- **detect: [:element | "expression booléenne"] ifNone: ["traitements alternatifs"]**
  - Retourne le 1er élément pour lequel l'expression booléenne est vraie
  - Exécute le bloc de traitements alternatifs si l'expression booléenne est fausse pour tous les éléments
- **detect: [:element | "expression booléenne"]**
  - Provoque une erreur si aucun élément n'est trouvé (expression booléen est toujours fausse)

Exemple de parcours d'une collection

```
|notes total moyenne notesCorrectes noteEliminatoire |
notes := Set new.
notes add: 12.
notes add: 6.5.
notes add: 15.
total := 0.
notes do: [:uneNote| total := total + uneNote].
moyenne := total / (notes size).
notesCorrectes := notes select: [:uneNote| uneNote >= 12].
noteEliminatoire := notes
                        detect: [:uneNote| uneNote < 8]
                        ifNone: [nil].
```

Parcours de 2 collections ordonnées

- **with: collection2**
  do: [:elementCol1 :elementCol2 |
  "traitement"]
  – Exemple :
  #(1 2 3) with: #(5 6 7) do: [:e1 :e2|
  Transcript cr; show: e1 * e2]
- **with: collection2**
  collect: [:e1 :e2 | "traitement"]
  – Exemple
  |sum|
  sum := #(1 2 3) with: #(5 6 7) do: [:e1 :e2| e1 + e2]

Manipulation des dictionnaires - 1

- **at: clé**
  – retourne la valeur (l'élément) associé à la clé
  – clé référence n'importe quel objet
- **at: clé put: valeur**
  – valeur référence un objet quelconque
  – associe valeur à clé
  – Si la clé est déjà utilisée, l'ancienne valeur est remplacée
- **removeKey: clé**
  – retire la clé et la valeur correspondante

Manipulation des dictionnaires - 2

- **at: clé ifAbsent: ["traitements"]**
  – Exécute les traitements du bloc si la clé n'est pas utilisée dans le dictionnaire
- **at: clé ifAbsentPut: ["traitements"]**
  – Le résultat de la dernière expression du block est stockée comme valeur associée à la clé si elle n'est pas utilisée
- **at: clé ifPresent: ["traitements"]**
  – Exécute les traitements du bloc si la clé est utilisée dans le dictionnaire

Parcours des dictionnaires (boucles)

- **keysAndValuesDo: [:clé :valeur| "traitements"]**
  – parcourt le dictionnaire
  – exécute les traitements pour chaque couple clé-valeur
- **keysDo: [:clé | "traitements"]**
  – Exécute les traitements pour chaque clé
- **valuesDo: [:valeur| "traitements"]**
  – Exécute les traitements pour chaque valeur
- **collect: [:valeur| "traitements"]**
  – Retourne un ensemble (Set) avec les résultats des traitements pour chaque valeur

Dictionnaires - Exemple

```
|notesPromo|
notesPromo := Dictionary new.
notesPromo at: 'Marie' put: 14.
notesPromo at: 'Rémi' put: 12.
notePromo at: 'Didier' put: 16.
notesPromo keysAndValuesDo: [:nom :note|
    Transcript cr.
    Transcript show: nom.
    Transcript tab.
    Transcript show: note]
```


"Conversion" des collections

- **Produit une copie**
  - asArray
  - asOrderedCollection
  - asSet
    - Exemple : #(1 2 3) asSet
  - asSortedCollection
    - Retourne une nouvelle collection triée par ordre croissant
  - asSortedCollection: blockBoolean
    - Le paramétre est un block qui retourne un boolée indiquant si deux éléments sont en ordre croissant
    - Exemple #(51 23 43) asSortedCollect: [:a :b | a < b]

Taille d'une collection

- **Message pour obtenir la taille**
  - size
- **La taille correspond au...**
  - Nombre de cases (même vides) d'un tableau
  - Nombre d'éléments d'une collection élastique

Taille d'une collection

- **Exemple avec Array**

```
a := Array new: 25.
a size. "→ 25"
a at: 8 put: 'bonjour Isici'.
a size. "→ 25"
```

- **Exemple avec Set**

```
s := Set new: 25.
s size. "→ 0"
s add: 'bonjour Isici'.
a size. "→ 1"
```

Taille d'une collection

- **Tester la taille**
  - isEmpty
  - ifEmpty: unBlock
  - ifNotEmpty: unBlock

- **Exemples :**

  #() isEmpty. "→ *true*"

  {Object new. #(). 123}
  ifNotEmpty: [Transcript cr; show: 'Bonjour']
  "Affiche Bonjour sur le Transcript"

Tests d'inclusion

- **includes: unObjet**
  - Retourne **true** si le paramètre est dans la collection

  #(123 'hello' -78) includes: 45
    - Retourne **false**

- **includesAny: uneAutreCollection**
  - Retourne **true** si au moins un élément du paramètre est dans la collection
  - Exemple : #(123 'hello' -78) includesAny: #('world' 123)
    - Retourne **true**

- **includesAll: uneAutreCollection**
  - Retourne **true** si tous les éléments du paramètre sont dans la collection
  - Exemple : #(123 'hello' -78) includesAll: #('world' 123)
    - Retourne **false**

Tests Arbitraires

- **allSatisfy: blockBooleen**
  - Retourne **true** si le bloc retourne **true** pour tous les éléments
- **anySatisfy: blockBooleen**
  - Retourne **true** si le bloc retourne **true** pour au moins un élément
- **noneSatisfy: blockBooleen**
  - Retourne **true** si le bloc retourne **false** pour tous les éléments
- **Exemples :**

  #(1 2 3) allSatisfy: [:element| element * 10 < 24]
    - Retourne **false**

  #(1 2 3) anySatisfy: [:element| element * 10 < 24]
    - Retourne **true**

  #(1 2 3) noneSatisfy: [:element| element * 5 = 20]
    - Retourne **true**

Copie

- **copyWith: unObjet**
  - Retourne une collection du même type avec tous les éléments + unObjet

- **copyWithout: unObjet**
  - Retourne une collection du même type avec tous les éléments sauf unObjet

Copie de collections ordonnées

- **allButFirst**
  - Copie avec tous les éléments sauf le premier
- **allButFirst: n**
  - Copie avec tous les éléments sauf les **n** premiers
- **allButLast**
  - Copie avec tous les éléments sauf le dernier
- **allButLast: n**
  - Copie avec tous les éléments sauf les **n** derniers

Concaténation

uneCollection , autreCollection

- Produit une nouvelle collection avec les éléments des 2 collections
- Exemple : l'expression suivante

#(4 5) , #(1 2 3)

- a pour résultat :

#(4 5 1 2 3)

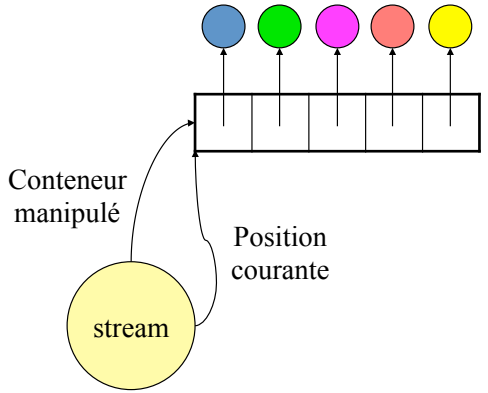
Les Streams (Flots)

Notion de Stream (flot)

- **Stream (flot en français)**
  - Manipulateur séquentiel de données
    - Manipulation == lecture ou écriture
- **Données dans un conteneur**
  - Différents conteneurs :
    - Chaîne de caractères
    - Collection
    - Fichier
    - Socket réseau
- **Données lues et écrites de n'importe quel type**
  - objets, caractères, octets (images, sons, …)

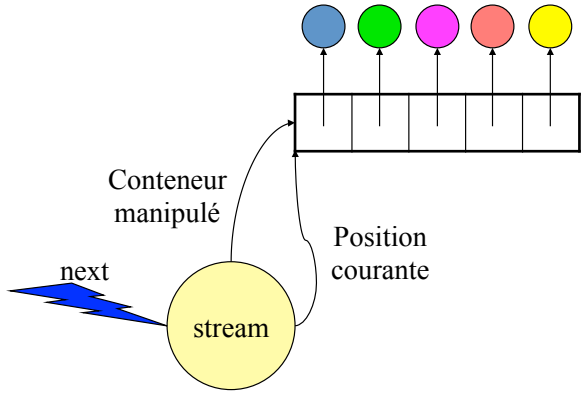
Stream de lecture

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



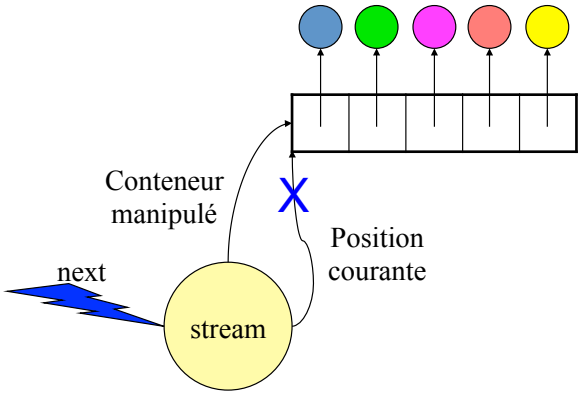
Stream de lecture

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



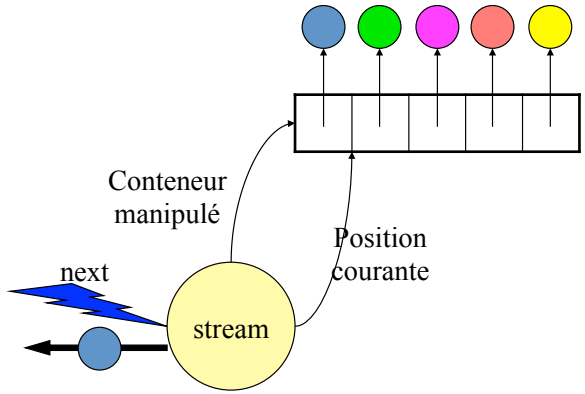
Stream de lecture

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

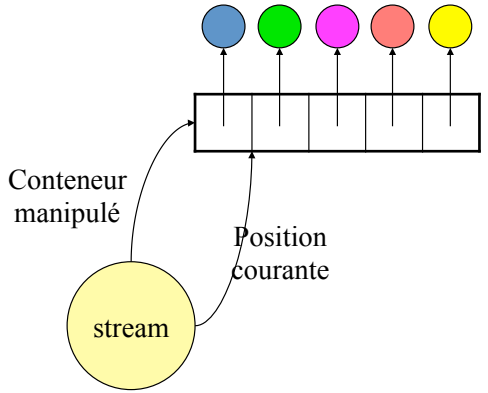


Stream de lecture

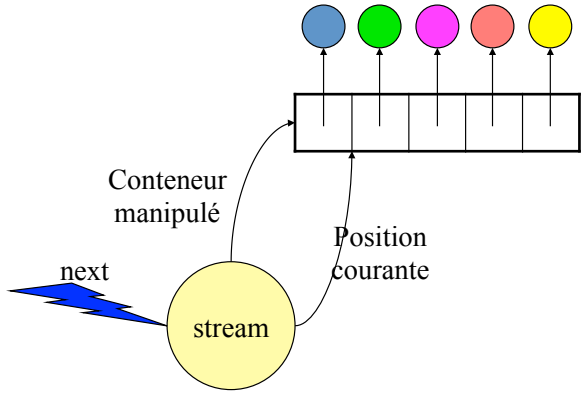
Noury Bouraqadi - option ISIC - Dépt. I. A.



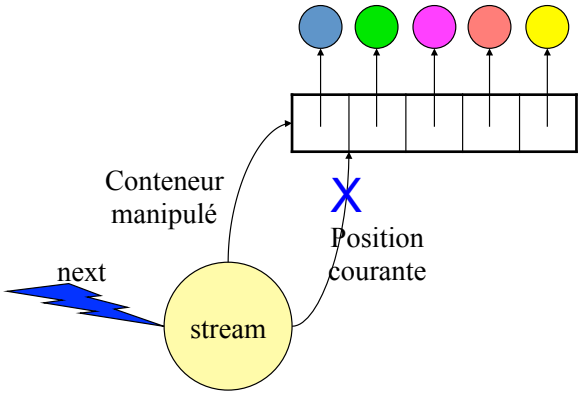
Stream de lecture



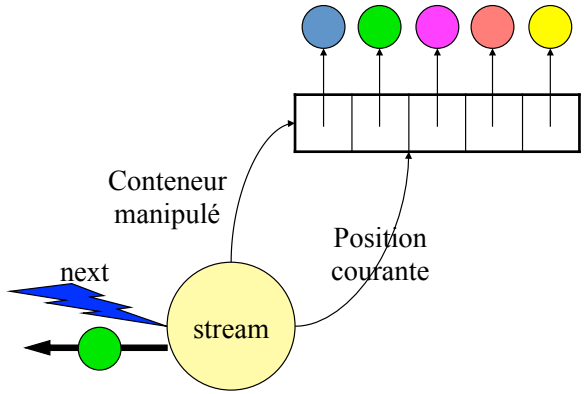
Stream de lecture



Stream de lecture

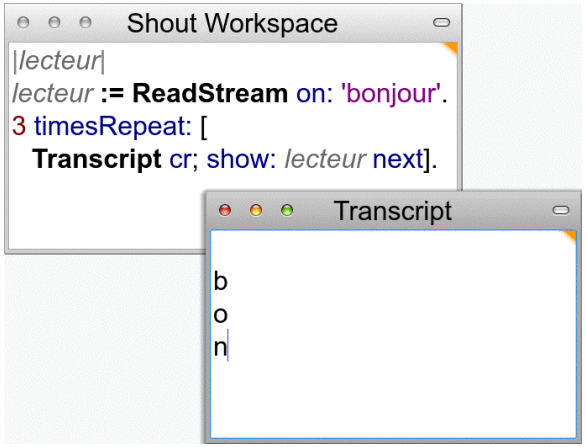


Stream de lecture



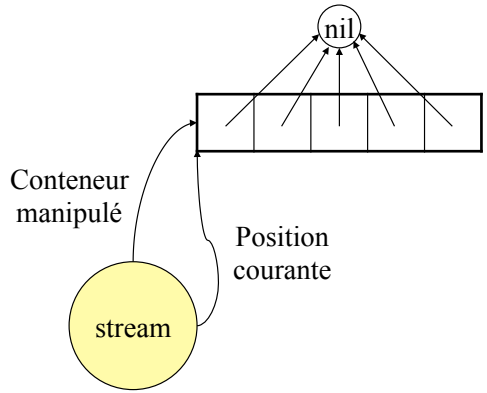
Stream de lecture - Le code

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



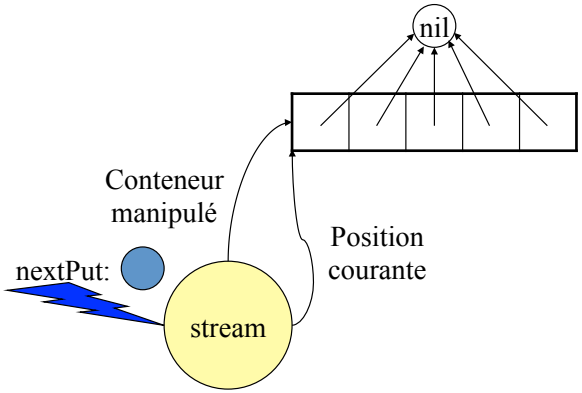
Stream d'écriture

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



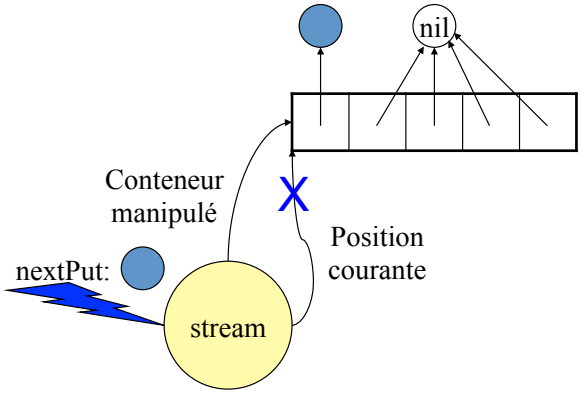
Stream d'écriture

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



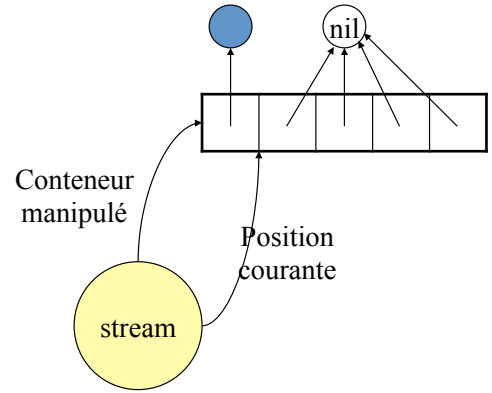
Stream d'écriture

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



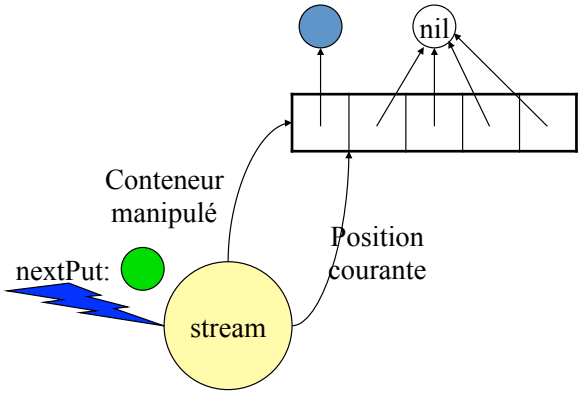
Stream d'écriture

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



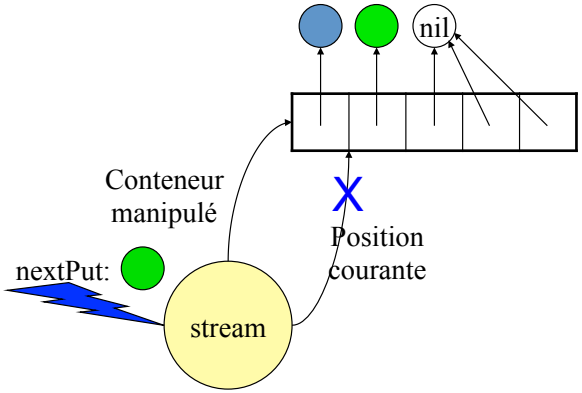
Stream d'écriture

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



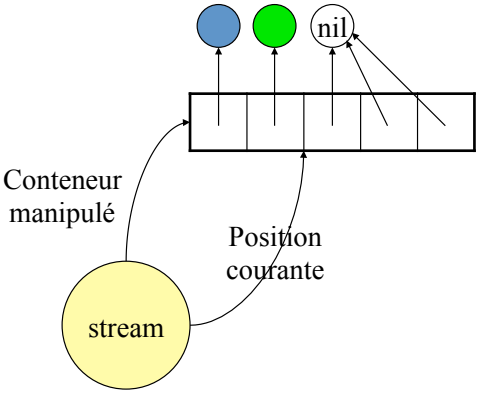
Stream d'écriture

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

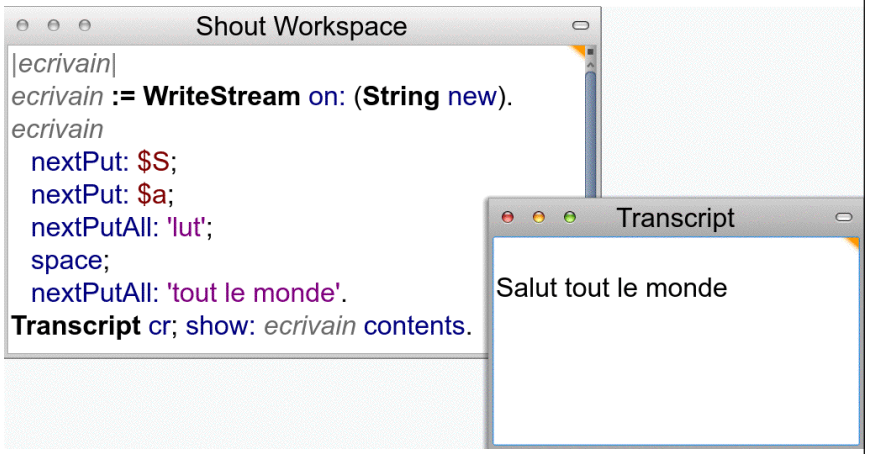


Stream d'écriture

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



Stream d'écriture - Le code



Streams & Représentation textuelle des objets

- **Représentation textuelle d'un objet**
  - Chaîne de caractères
  - Décrit l'objet
  - Obtenu en envoyant le message printString
    - monObjet printString
- **printString envoie à self le message**
  - self printOn: StreamDEcriture
- **Définir la méthode printOn:**
  - Pour modifier la représentation textuelle d'un objet

Exemple de représentation textuelle

- **Lampe>>printOn: unStream**

```
unStream
    nextPutAll: 'lampe';
    space.
self estAllumee ifTrue: [
    ^unStream nextPutAll: 'allumée'].
unStream nextPutAll: 'eteinte'
```