Langages informatiques

Fabrice EUDES, Pascal EVRARD, Thi-Lan LUU,
François RECHER & Yann SECQ

30 mai 2016

Algorithme & langage

- Comment exprimer la résolution automatisée d'un problème ?

- Le langage naturel est ambigu et les machines ne disposent que de primitives très simples

- Algorithme = description formalisée d'un processus automatisé de résolution d'un problème donné

- Langage = grammaire formelle permettant l'expression d'un programme transformable en un code interprétable par une machine

Du problème à l'algorithme

- Plusieurs étapes pour aboutir à un algorithme pour un problème donné:
  - Compréhension du problème à résoudre (exemple)
  - Représentation des données du problème
  - Recherche d'un procédé (général!) de résolution
  - Description détaillée de ce procédé de résolution avec un ensemble restreint de « mots »
  - Vérification de la validité de l'algorithme sur différents jeux de données (tests)
- Loin d'être simple … mais c'est tout l'intérêt ;)

Langages informatiques

- **Comment structurer les données et les traitements lorsque les algorithmes ou systèmes à réaliser deviennent complexes ?**

- Il existe plusieurs paradigmes de programmation:

    - programmation impérative/structurée
    - programmation orientée objets
    - programmation fonctionnelle
    - programmation logique
    - et plein d'autres (descriptive, événementielle, orienté aspects, …)

- **Ces paradigmes sont différentes manière de représenter les données et les traitements**

Programmation impérative

- Aussi appelée programmation structurée
- Métaphore de la « recette de cuisine »
- L'algorithme décrit une séquence d'instructions transformant les données pour aboutir au résultat
- Organisation de la complexité grâce aux sous-programme et modules
- Typiquement: **C**, Pascal, BASIC, ADA, assembleur

Définition d'une fonction calculant la factorielle d'un nombre en programmation impérative (C)

`Rappel: n! = 1*2*…*n (ou n! = n*(n-1)! et 0! = 1)`

nature du résultat

nom de la fonction

paramètre(s) de la fonction

variable locale à la fonction (accumulateur)

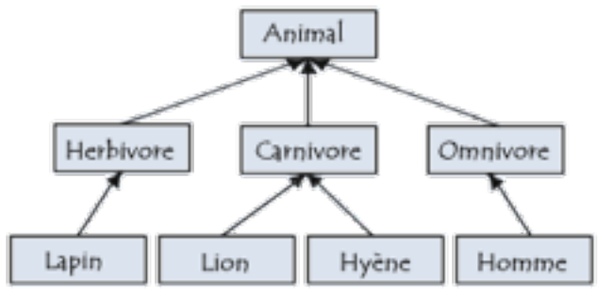
```c
int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; ++i)
        result *= i;
    return result;
}
```

mission accomplie: retourne le résultat du calcul de la fonction

répétition accumulant le produit des nombres de 1 à n

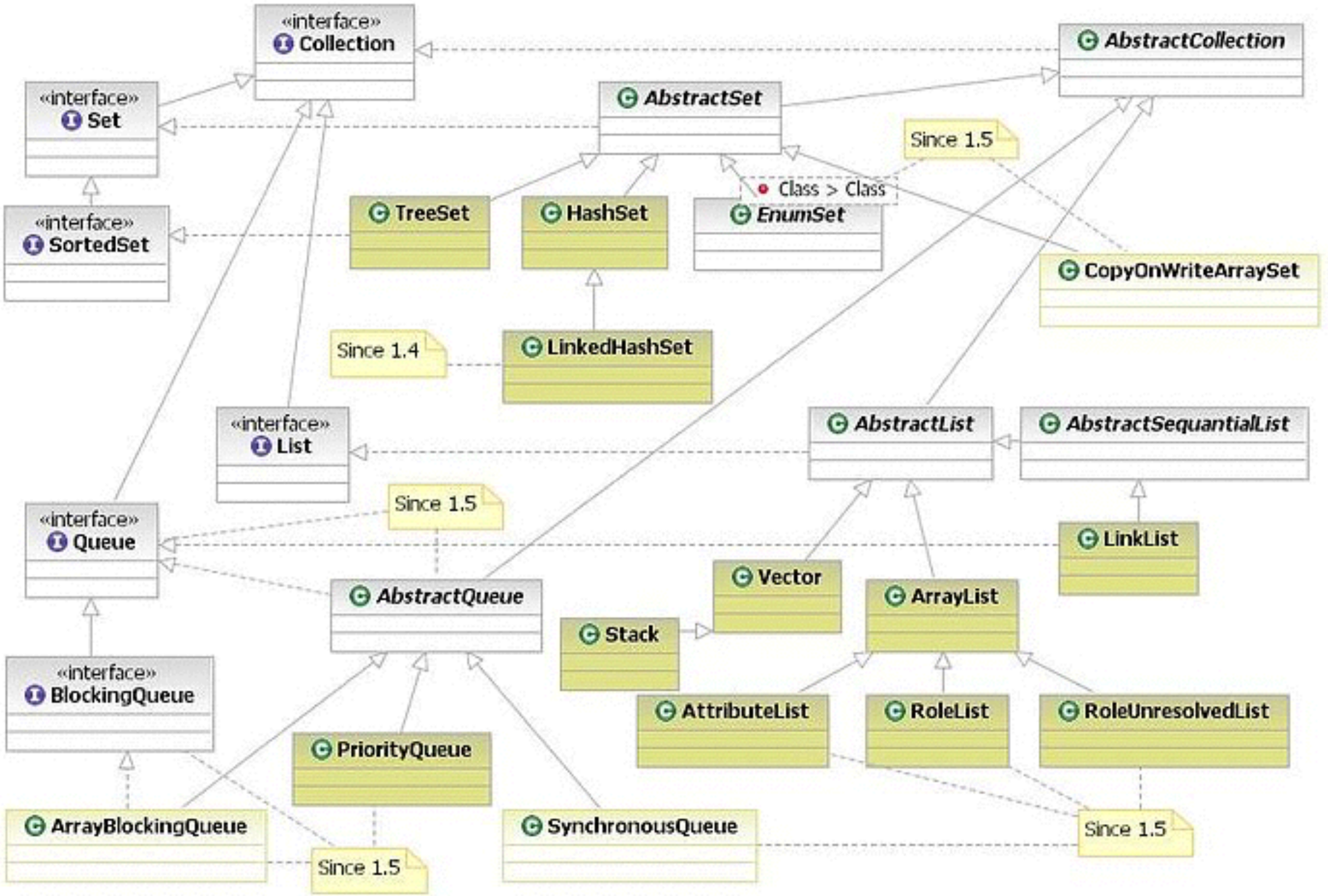
Programmation orientée objets

- Métaphore de la taxonomie:



- Regroupement de l'état (données) et du comportement (traitements) au sein d'un **objet**
- Relation est-un (héritage) et constitué-de (agrégation)
- Organisation de la complexité par des classes (le plus souvent)
- Langages: smalltalk, C++, Java, C#, …

Schéma UML représentant les structures de données définies dans la librairie standard de Java



Programmation fonctionnelle

- Métaphore … programmer par composition de fonctions ?

- Concepts: fonction et liste, récursivité, immutabilité

- La référence: LISP (J. Mc Carthy)

- Revient en force avec l'explosion du parallélisme (avec erlang par exemple)

- Langages: LISP, Scheme, Haskell, JavaScript, OCaml …

Définition d'une fonction calculant la factorielle d'un nombre en programmation impérative (C)

Rappel: **0! = 1 et n! = n*(n-1)!**

```
(defun factorial (n)
  (if (= n 0)
      1
      (* n (factorial (- n 1))) ) )
```

traduction directe de la définition mathématique (récursivité)

```
(defun factorial (n)
  (reduce #'*
      (loop for i from 1 to n collect i))
```

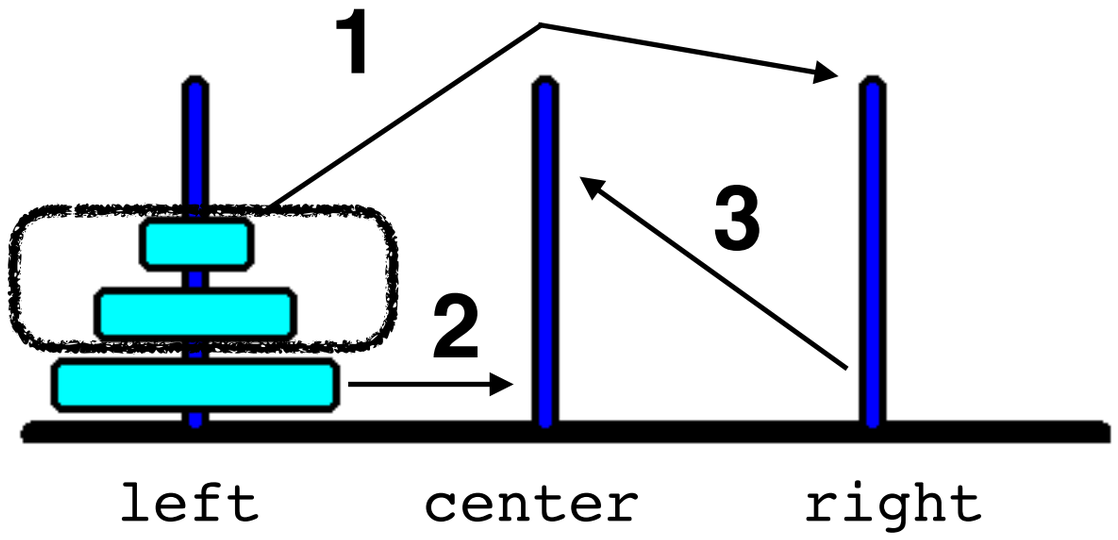
création de l'ensemble [1..n]

réduction de l'ensemble avec un produit

Programmation logique

- Métaphore de l'intelligence artificielle :)
- Principe: modélisation de faits et description de règles, ensuite un moteur de déduction tente de résoudre votre problème !
- Exemple classique: placement des 8 reines
- Langages: **Prolog**, Datalog, Oz, …

Le problème des tours de Hanoi




```
move(1,X,Y,_) :-
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
move(N,X,Y,Z) :-
    N>1,
    M is N-1,
    move(M,X,Z,Y),    1
    move(1,X,Y,_),    2
    move(M,Z,Y,X).    3
```

```
?-  move(3,left,right,center).
Move top disk from left to right
Move top disk from left to center
Move top disk from right to center
Move top disk from left to right
Move top disk from center to left
Move top disk from center to right
Move top disk from left to right

yes
```

Algorithmes & langages

- Lego/crêpier: illustration du processus de résolution et de la difficulté de son expression de manière non ambigu dans un langage formel

- Importance du paradigme de programmation et de la représentation des informations du problème

- Programmer, c'est créer une théorie comme en philosophie, mais exécutable et opérationnelle (mais généralement restreinte ;))

- Possibilité d'expérimenter la programmation dans des environnements simples comme LOGO ou Scratch / studio.code.org pour la programmation impérative et/ou fonctionnelle, ou python !