UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

INF6
ALGORITHME AVANCÉ

Manuele Kirsch Pinheiro

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Objectifs et organisation

- Objectifs
  - Introduction aux structures de données
  - Etude des principales structures de données et de leur algorithmes
- Evaluation
  - TPs à rendre
  - Interro surprises
  - Partiel

Bibliographie

- Bibliographie
  - Vincent Granet, « Algorithme et programmation en Java : cours et exercices corrigés », 4e édition, Dunod, 2014
  - Michel Divay, « Algorithmes et structures de données : cours et exercices corrigés en langage C », Dunod, 1999
  - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, « Introdiction to algorithms », 3rd edition, MIT Press, 2009
  - Plus des nombreux autres livres et sites Web… ☺

Contenu prévisionnel

- Piles et files
- Listes
- Récursivité
  - Récursivité dans le calcul
  - Récursivité structurelle
- Arbres binaires
  - Parcours en profondeur et en largeur
- Généralisation de la notion d'arbre
  - Insertion et suppression de nœuds
- Arbre de recherche
  - Recherche
  - Rééquilibrage

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

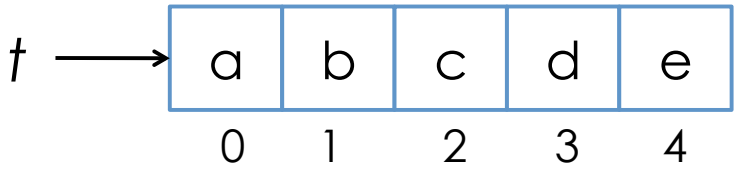
Structure de données

- **Structure de données**
  - Ensemble structuré des données présentant un comportement précis
  - *Structure logique destinée à contenir des données afin de leur fournir une organisation permettant leur traitement* (fr.wikipedia.org)
- Les structures des données sont au cœur des applications informatiques
  - Une **bonne organisation** des données permet un **traitement simple et efficace** de ces données

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Structure de données

- Exemple : **Tableaux**
  - Un tableau est une structure linéaire, agrégeant des **éléments** d'un **même type**, en **nombre fini**, **accessibles** sans un ordre particulier de **manière directe**, par un indice



t [1] == b

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

PILES

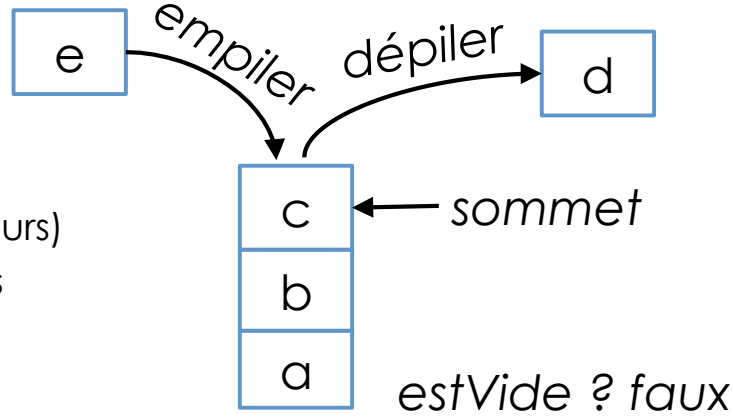
UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Piles

- **Pile**
  - Séquence d'**éléments accessibles** par une seule **extrémité** (**sommet**)
  - Les éléments sont organisés par leur **ordre d'arrivé**
  - Toutes les **opérations** s'appliquent au **sommet**

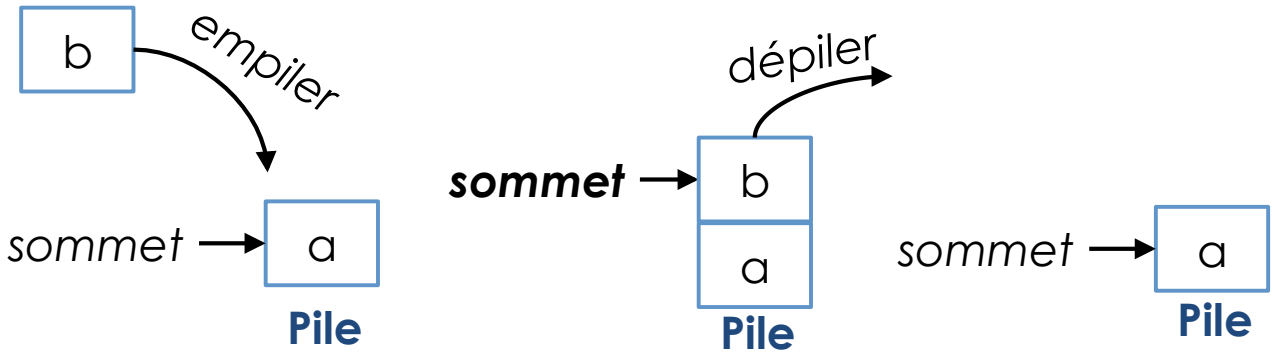
Exemples d'usage :

- pile d'exécution (SE)
- fonction undo (éditeurs)
- expressions post-fixés
  2 2 +



Piles

- **Piles**
  - Les éléments sont organisés en fonction de **leur ordre d'arrivé, suivant le modèle LIFO**
  - Les piles suivent le modèle **LIFO** (***Last In, First Out***)
    « *dernier entré, premier sorti* »



Piles

- **Opérations**

  Opérations abstraites définies sur les piles

  - **estVide** : vérifie si la pile est vide ou si elle contient des éléments
  - **empiler** : ajoute un nouvel élément au sommet de la pile
  - **dépiler** : supprime le sommet de la pile
  - **sommet** : consulte le sommet de la pile (sans le supprimer)

Pile

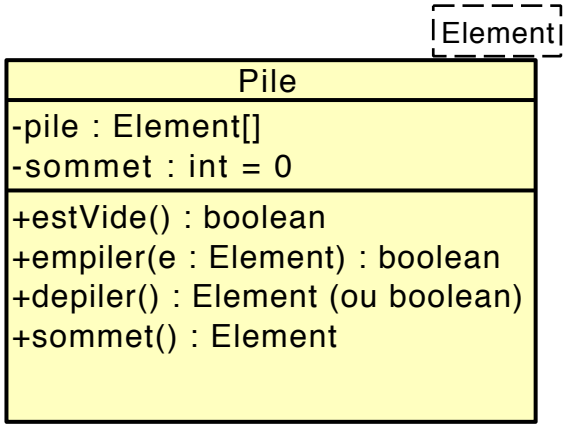
- **Opérations : conditions**
  - ***dépile*** enlève le dernier élément ajouté (LIFO)
  - ***sommet*** récupère le dernier élément ajouté (LIFO)
  - On ne peut pas dépiler une **pile vide**, ni avoir un sommet
    - soit ***p*** une pile et ***e*** un élément
      - $\nexists p$ *tel que* $p = dépiler(pileVide)$
      - $\nexists e$ *tel que* $e = sommet(pileVide)$

Exception Pile Vide

  - Les piles peuvent avoir une taille fixe et y être limitées, ou elles peuvent être extensibles
    - si taille fixe limitée, *empiler (pilePleine, e)* ⇒ *Exception* Pile Pleine

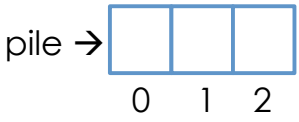
Pile

- Algorithmes



Taille = 3

sommet = 0



empiler (e) : booléen
entrée :
Elément e
sortie :
booléen réponse

Si sommet < pille.Taille
alors
pile [sommet] = e
sommet = sommet + 1
réponse = VRAI
sinon
réponse = FAUX
(ou exception)
fin Si
retourner réponse
Fin empiler

Pile

- Exercices
  1) simuler l'exécution de l'algorithme ***empiler*** pour l'ajout des éléments a, b, c et d dans la pile.
     a. quel est l'état final de la pile et du sommet ?
     b. quel est la réponse de l'opération ***empiler(d)*** ?
  2) simuler le comportement de l'opération ***dépiler*** pour les mêmes éléments (a, b, c, d).
  3) écrire les algorithmes pour les opérations ***estVide*** et ***dépiler***.

Pile

estVide ( ) : booléen
sortie :
booléen réponse
Si sommet > 0
alors
réponse = FAUX
sinon
réponse = VRAI
fin si
retourner réponse
Fin estVide

dépiler ( ) : Elément
sortie :
Elément s

Si estVide( )
alors
Lancer Exception Pile Vide
sinon
sommet = sommet -1
s = pile [sommet]
fin Si
retourner s
Fin dépiler

Pile

- Une implémentation par tableau est la seule possible en Java ?
  - Modéliser la notion de pile en Java de manière à maximiser la réutilisation et le couplage.

Opérations de base :
- estVide (*isEmpty*)
- empiler (*push*)
- dépiler (*pop*)
- sommet (*peek*)

Autres opérations utiles :
- taille (*size*)
- estPleine (*isFull*)
- éléments (*elements*)