Université Paris Cité

L1 informatique - Année 2023–2024

TD de Concepts Informatique n° 1

Nous utilisons la classe `Stack` et certaines de ses variantes comme `Stack<Integer>`, `Stack<String>`, `Stack<String[]>`, etc.
Rappelons que pour une pile `p`, nous avons les opérations suivantes

- `p.push(x)` pour ajouter l'élément `x` au sommet de la pile
- `p.pop()` qui enlève et retourne l'élément au sommet de la pile
- `p.empty()` qui vérifie si la pile est vide ou non

Exercice 1. .
Pour chaque question, commencer par faire un ou des schémas avant d'écrire le code.

1. Écrire une fonction de prototype `void afficher(Stack<Integer>)` qui affiche verticalement, fond de pile vers le bas, le contenu de la pile passée en paramètre ; celle-ci doit être intacte après l'exécution.
2. Écrire une fonction de prototype `void transvaser(Stack<Integer>, Stack<Integer>)` qui déplace le contenu de la première pile vers la deuxième de sorte que l'ordre soit inversé.
3. Écrire une fonction de prototype `void deplacer(Stack<Integer>, Stack<Integer>)` qui déplace le contenu de la première pile vers la deuxième de sorte que l'ordre soit maintenu.
4. Écrire une fonction de prototype `void deplacerPairImpair(Stack<Integer>, Stack<Integer>)` qui déplace le contenu de la première pile vers la deuxième de sorte que tous les nombres pairs soient au-dessous des nombres impairs (l'ordre des pairs entre eux et des impairs entre eux n'importe pas ici).
5. Écrire une fonction de prototype `void copierPairs(Stack<Integer>, Stack<Integer>)` qui copie les entiers pairs de la première pile vers la deuxième de sorte que la première pile soit intacte (après l'exécution) et que l'ordre des pairs entre eux soit le même dans les deux piles.

Exercice 2. *Pile vs. tas.*
En distinguant pile et tas, faire un schéma pour illustrer l'évolution du contenu de la mémoire lors de l'exécution de chacun des extraits de programme suivants (en détaillant les étapes les plus importantes des constructions opérées).

```
1 int i = 64;
2 long k = 129;
3 k = i;
```

```
1 int[] t = new int[4];
2 for(int i=0; i<4; i++)
3     t[i] = i+1;
```

Exercice 3. *Piles et expressions arithmétiques.*
Considérons l'expression

```
( 5 / ( 3 - 1 ) ) * ( 2 + 4 ).
```

1. Dessiner son arbre syntaxique.
2. Donner sa forme préfixe
3. Donner sa forme postfixe.
4. Décrire l'évolution du contenu de la pile (chaque `push` et chaque `pop`) lors de l'évaluation de sa forme postfixe.
5. Donnez un algorithme pour évaluer une expression sous forme préfixe en utilisant une pile. Décrivez l'évolution de la pile lors de l'exécution de cet algorithme sur l'exemple ci-dessus.
6. Donnez un algorithme pour transformer une expression sous forme infixe en forme postfixe en utilisant une pile. Décrivez l'évolution de la pile lors de l'exécution de cet algorithme sur l'exemple ci-dessus.

Exercice 4. *Conversion.*
Écrire toutes les fonctions de conversion entre les formes infixe, préfixe, et postfixe.