Université Paris XII
IUT de Sénart-Fontainebleau
Département Informatique
S3 - APL
2017–2018

PARTIEL 1

Seuls sont autorisés, à titre de documents, les programmes imprimés comprenant explicitement le nom imprimé de l'étudiant sur chaque page (et celui-ci seulement), un extrait de table Unicode et les notes manuscrites.

Les exercices sont indépendants mais ils doivent être présentés dans l'ordre (prévoir deux pages par exercice).

Exercice 1.- (**Envois différés**)

Une file d'attente *(*queue *en anglais) est une structure de données dynamique linéaire : on place les éléments un à un en fin de la file d'attente et on les récupère un à un en début de la file d'attente.*

Autrement dit on récupère le premier qui s'est placé dans la file (alors que dans une pile*, on récupère le dernier).*

On va considérer ici des files d'attente d'entiers naturels.

- 1^o) Concevoir une classe Java `QueueElement` ayant pour attributs une valeur entière et deux objets de la classe `QueueElement` : l'élément précédent (*previous*) et l'élément suivant (*next*) dans la file d'attente.

Les méthodes sont : le constructeur prenant en argument un entier (pour construire le premier élément d'une file d'attente vide), le constructeur prenant en argument un entier et un objet de la classe `QueuElement` (qui servira à ajouter un élément à une file d'attente non vide), les accesseurs `setNext(QueueElement)`, `getVal()` et `getNext()` dont les noms précisent les rôles.

- 2^o) Concevoir une classe Java `Queue` dont les attributs sont le premier élément (*begin*) de la pile et le dernier (*end*) et les méthodes sont : le constructeur par défaut (construction d'une file d'attente vide), `isEmpty()` pour tester si la file d'attente est vide, `push(int)` pour ajouter un élément en fin de la file d'attente, `pop()` pour récupérer la valeur du premier élément si la file d'attente n'est pas vide et `affiche()` qui affiche les valeurs des éléments de la file, du premier au dernier, un par ligne.

- 3^o) Écrire une application Java qui demande un certain nombre d'entiers naturels non nuls, qui affiche (envoie) ces entiers lorsqu'on entre 0 et qui termine la session (après avoir affiché les entiers qui sont encore dans le tampon) lorsqu'on entre - 1.

[*Un exemple de session est* :

```
>java envoi
n = 1
n = 7
n = 9
n = 0
1
7
9
n = 5
n = 12
n = 0
5
12
n = 3
n = 4
n = -1
3
4

>
```

]

- 4^o) a) Usuellement un tel *tampon* est implémenté par un tableau. Expliquer pourquoi cela n'est pas une bonne implémentation.

b) On aurait pu utiliser une *pile* pour entreposer les valeurs entrées : dans ce cas il faut, au moment de l'envoi différé, placer d'abord les éléments de la pile dans une pile auxiliaire puis afficher les éléments de la pile auxiliaire (pour les afficher dans le bon ordre). Expliquer pourquoi cette façon de faire est moins efficace que l'utilisation d'une file d'attente.

Exercice 2.- (**Inversion d'une chaîne de caractères**)

- 1^o) Écrire une méthode **récursive** :

`String inversion(String s)`

qui renvoie la chaîne de caractères écrite dans l'ordre inverse ('Bonjour' donne 'ruojnoB').

[*On remarquera que* :
inv(λ) = λ,
inv(aw) = inv(w) + a,
où λ *est le mot vide, a un caractère et w une chaîne de caractères.*]

- 2^o) Écrire une application Java qui demande une chaîne de caractères et qui affiche l'inverse de celle-ci.

[*Un exemple de session est* :

```
>java inverse
Entrez un mot :  Jean Dupont
Ce mot inversé est :  tnopuD naeJ
>
```

]

Remarque.- Voyez-vous comment on aurait pu répondre au problème de l'exercice 1 en utilisant une méthode récursive, sans utiliser de structure de données dynamique ?

Documentation
Complément sur les chaînes de caractères

Donnons deux méthodes de la classe `String` :

- `char charAt(int index)`

  renvoie le index-ième caractère, le premier caractère ayant pour index 0 ;

- `String substring(int indexbegin, int indexend)`

  renvoie la sous-chaîne de caractères constituée des caractères de la chaîne commençant par celui d'index `indexbegin` (inclus) jusqu'à celui d'index `indexend` (non inclus).