université PARIS DIDEROT PARIS 7

PARTIEL | CI2 | 2/3/2016 8^{30}–10^{00}

Aucun document ou support autre que le sujet ou les copies d'examen n'est autorisé.
(la copie ou les brouillons du voisin ne sont pas des supports autorisés).
Éteignez impérativement vos mobiles.

Lorsque des calculs sont nécessaires, il est impératif de les présenter sur la feuille d'examen. Il est aussi nécessaire de **justifier** ses réponses.

1 Exercice

(En mathématiques) On appelle **matrice triangulaire supérieure** une matrice carrée dont les valeurs en dessous de la diagonale principale sont nulles, comme ici :

$$\begin{pmatrix} a_{11} & a_{21} & a_{31} & ... & a_{n1} \\ 0 & a_{22} & a_{32} & ... & a_{n2} \\ 0 & 0 & a_{33} & ... & a_{n3} \\ \vdots & ... & \ddots & \ddots & \vdots \\ 0 & 0 & ... & 0 & a_{nn} \end{pmatrix}.$$

1. On souhaite représenter en Java une telle structure de données :
   (a) Quel peut être le type Java d'une telle structure ? Donner un exemple de code permettant de créer et initialiser la matrice triangulaire supérieure $\begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}$.
   (b) Dessiner l'état de la mémoire correspondant à l'objet précédemment créé (en plaçant les différentes choses dans la pile ou le tas).
2. Par souci d'économies on souhaite se passer de la place occupée par les 0 situés sous la diagonale en ne stockant qu'une structure semblable à $\begin{matrix} 1 & 2 \\ 3 & \end{matrix}$.
   (a) Quel peut être le type Java d'une telle structure ? Donner un exemple de code permettant de créer (sans en initialiser le contenu) une telle structure.
   (b) Écrire une fonction `initialiser(`T_1 `t1, `T_2 `t2)` qui étant donné une structure du type initial de la question (1a) (ici nommé T_1) permet d'initialiser une structure «économe» du type de la question (2a) (ici du type nommé T_2) avec les valeurs correspondantes.
   (c) Écrire une fonction `afficher(`T_2 `t)` qui étant donné une structure du type de la question (2a), permet de l'afficher comme une matrice du premier type (question (1a)), c'est-à-dire avec les zéros qui manquent...

2 Exercice

Rappel : une pile d'entiers en Java est du type `Stack<Integer>` et possède les opérations : `Integer pop()` qui dépile et renvoie l'élément en sommet de pile (et provoque une erreur si la pile est vide), `push(Integer)` qui ajoute un élément en sommet de pile, `boolean isEmpty()` qui permet de déterminer si la pile contient des éléments ou non. La construction d'une pile vide s'effectue avec l'instruction `new Stack<Integer>()`.
On définit une **liste d'entiers** comme une structure de données permettant de contenir des entiers ordonnés par leur position dans la liste. Une liste possède aussi des méthodes permettant de la manipuler comme : `void ajouterALaFin(Integer)`, `void ajouterAuDebut(Integer)`, `boolean estVide()`, `Integer enleverLePremier()`, `Integer enleverLeDernier()`, `Integer premier()`, `Integer suivant()`...
On se propose d'utiliser deux piles pour réaliser une liste, voici comment. Étant donnée une liste constituée des n éléments e_1, e_2, e_3, ..., e_p, e_{p+1}, ..., e_n, on peut ranger ses éléments à l'aide de deux piles :

| e_1 | e_2 | e_3 | ... | e_p |

| e_{p+1} | ... | e_n |

La première contenant les éléments numérotés de 1 à p (pour un certain p) et la seconde contenant en ordre inverse les éléments numérotés de $p+1$ à n. On notera qu'il est possible de modifier le rangement dans les deux piles sans modifier l'ordre des éléments en dépilant d'un côté et rempilant de l'autre :

e_1	e_2	e_3	...	e_{p-1}

e_p	e_{p+1}	...	e_n

On notera que la même liste a plusieurs représentations dans ces deux piles. On supposera qu'il existe deux attributs correspondant à deux piles déjà créés et nommées `pileGauche` et `pileDroite` et du type `Stack<Integer>`. Le code devra être fourni dans le langage Java.

1. Écrire une fonction `boolean estVide()` permettant de déterminer si une liste est vide ou non.
2. Écrire une fonction `boolean deplacerADroite()` permettant, si la pile de gauche n'est pas vide, de déplacer son élément en sommet sur la pile de droite et de renvoyer `true`. Si la pile de gauche était initialement vide de renvoyer `false`. Écrire la fonction `void deplacerToutADroite()` qui déplace tous les éléments de la pile de gauche vers la pile de droite. Pour la suite, on supposera qu'il existe les méthodes symétriques : `boolean deplacerAGauche()` et `void deplacerToutAGauche()`.
3. Écrire une fonction `void ajouterAuDebut(Integer v)` qui permet d'ajouter en début de liste la valeur `v`. On supposera qu'il existe une méthode symétrique `void ajouterALafin(Integer v)`.
4. Écrire une fonction `afficherListe()` permettant d'afficher tous les éléments contenus dans la liste et dans leur ordre (du début à la fin).
5. Écrire une fonction `int longueur()` permettant de déterminer le nombre d'éléments contenus dans la liste.

3 Exercice

Soit le programme Java suivant :

```
public class Prog {
  public static int i;
  public static int j;
  public static void main(String []a) {
    j = 0;
    i = 0;
    while (i<5) {
      j = i*i;
      i++;
      System.out.println(j);
    }
  }
}
```

1. Qu'affiche le programme si on l'exécute ?
2. De combien de variables ce programme a t-il besoin pour fonctionner ? (On ne comptera pas le paramètre de la fonction `main`).
3. Traduire, tel qu'en cours et en tds, ce programme en un programme ne contenant qu'un tableau d'entiers appelé `memoire`, un entier appelé `instruction`, qu'une seule boucle `while` et une seule instruction `switch`.
4. Qu'est-ce que cela changerait à la traduction si dans le programme original on déplaçait l'instruction `System.out.println(j);` après la boucle `while` ?