Université Paris XII
IUT de Fontainebleau
Département informatique
FI-2
2002-2003

PARTIEL 1

Seuls sont autorisés, à titre de documents, les listings comprenant explicitement le nom de l'étudiant (et celui-ci seulement) et les notes manuscrites.

Exercice 1.- (**Pile**)

Une **pile** *est une structure de donnée dynamique définie récursivement : on part d'un nœud appelé* **sommet**, *chaque nœud comprend une donnée et un* `precedent`, *celui-ci étant vide ou étant lui-même un nœud. On ne peut accéder à une pile qu'en lui ajoutant une valeur au sommet (à travers un nœud, on dit qu'on* **empile***) ou en récupérant la dernière valeur ajoutée (on* **dépile***) si la pile n'est pas vide.*

- 1^o) Définir la classe `Noeud` des éléments d'une pile d'entiers en Java.

La classe `Noeud` comprendra deux données (un entier, qui est sa valeur, et un nœud) et quatre méthodes : le constructeur par défaut (de valeur zéro et de fils nul), le constructeur à deux arguments (un entier et un nœud), la méthode `getVal()` qui renvoie la valeur et la méthode `getPrecedent()` qui renvoie le nœud précédent.

- 2^o) Définir la classe `Pile` d'entiers en Java.

La classe `Pile` comprendra une donnée privée (le sommet qui est un nœud) et quatre méthodes : le constructeur par défaut (qui construit une **pile vide**, c'est-à-dire sans nœud), la méthode `empile()` à un argument entier (qui ajoute un nœud, sa valeur étant cet entier), la méthode booléenne sans argument `vide()` (qui renvoie vrai si la pile est vide), la méthode `depile()` (sans argument, qui renvoie le dernier entier ajouté si la pile n'est pas vide).

Les piles servent aux compilateurs pour faciliter le travail d'évaluation des expressions. L'homme écrit généralement des expressions telles :

```
(6 + 2) * 5 - 8 / 4
```

en **notation infixe**. *La* **notation postfixe** *fait apparaître les opérateurs binaires à droite des deux opérandes :*

```
6 2 + 5 * 8 4 / -
```

dans laquelle aucune parenthèse n'est nécessaire.

- 3^o) Écrire la classe `EvaluateurPostfixe`, qui demande de saisir une expression postfixe (constituée de nombres entiers à un chiffre et des quatre opérations arithmétiques) au clavier et qui affiche la valeur de cette expression.

L'algorithme d'évaluation de l'expression est le suivant (en supposant que l'expression est bien formée) :

- a) créer un pile vide
- b) lire l'expression de gauche à droite.
  - si le caractère courant est un chiffre, l'empiler ;
  - sinon le caractère courant est un opérateur `op` :
    - dépiler un élément `x` du sommet de la pile ;
    - dépiler un élément `y` du sommet de la pile ;
    - calculer `y op x` ;
    - empiler le résultat du calcul sur la pile ;
- c) quand on est arrivé à la fin de l'expression, dépiler la valeur du sommet de la pile qui est le résultat.

Exercice 2.- (**Ascenseur**)

Écrire une application Java faisant apparaître un cadre fermant sur lequel apparaît un ascenseur, un texte indiquant la valeur (entière) d'un diamètre et un disque du diamètre indiqué sur l'ascenseur à chaque fois que l'on déplace le curseur de celui-ci.

DOCUMENTATION

1 Compléments sur les chaînes de caractères

La méthode :

```
int length()
```

de la classe `String` permet d'obtenir la longueur d'une chaîne de caractères.

La méthode :

```
char charAt(int)
```

de la classe `String` permet d'obtenir le caractère d'index donné d'une chaîne de caractères. Les index commencent à 0.

2 Tracé d'une ovale

La méthode :

```
void fillOval(int, int, int, int)
```

de la classe `Graphics` permet de dessiner une plaque ovale dont les coordonnées du centre sont les deux premiers paramètres, les demi-axes horizontal et vertical les deux paramètres suivants.

3 Ascenseur

Les composants de la classe :

```
Scrollbar
```

du paquetage `java.awt` sont des objets que manipulent l'utilisateur pour choisir une valeur dans une gamme de valeurs entières. Le composant présente des traits de graduation et un curseur (que l'on déplace avec la souris ou les touches flèche), ainsi que deux flèches vers le bas et vers le haut.

Le constructeur par défaut crée un ascenseur vertical horizontale avec des graduations de 0 à 100 et une valeur initiale de 0.

La méthode :

```
int getValue()
```

de cette classe permet de récupérer la valeur choisie par l'utilisateur.

Les objets de la classe `Scrollbar` génèrent des événements de la classe :

```
AdjustmentEvent
```

du paquetage `java.awt.event` quand l'utilisateur déplace le curseur de l'ascenseur. Pour répondre à ces changements, on doit implémenter l'interface :

```
AdjustmentListener
```

en surchargeant sa méthode :

```
public void adjustmentValueChanged(AdjustmentEvent e)
```

et l'associer grâce à la méthode :

```
addAdjustmentListener(AdjustmentListener)
```

de `Component`.