TD 8 – Les Piles

Structures de données (IF 122) – 2005

La classe Pile

Avec une structure de données de type pile, les seules opérations possibles sont *empiler, dépiler, lire la tête* et *test du vide*. Les piles permettent par exemple à la machine virtuelle Java ou aux microprocesseurs de gérer les appels de fonctions (récursives) : lorsqu'une méthode fait un appel, les valeurs de ses variables locales sont sauvegardées dans la pile.

On se restreint d'abord à des piles d'entiers dont la taille est bornée par `tailleMax`. On peut les implanter avec la classe suivante :

```
class Pile{
    int tailleMax; // taille maximale de la pile
    int[] contenu; // éléments contenus dans la pile
    int tete; // position de la tete de la pile dans le tableau
}
```

L'élément `contenu[tete]` est l'élément qui se trouve en haut de la pile. Par convention, la pile est vide si `tete` vaut -1.

Exercice 1 (TD/TP) Écrire un constructeur `Pile(int taille)` qui initialise une pile avec une taille maximale donnée.

Exercice 2 (TD/TP) Écrire les méthodes `empiler`, `depiler`, `lireTete` et `estVide`.

Calculateur à pile

On veut écrire un petit calculateur capable d'effectuer les opérations $+$, $-$, $*$, / (division entière), % (modulo) sur les nombres entiers. On utilisera une méthode analogue à celle utilisée dans les calculatrices HP.
Pour effectuer 56 + 23, on procède ainsi :

```
> 56
> 23
> +
> =
> 79
> q
```

Ainsi chaque nombre lu est placé dans la pile ; quand on lit un opérateur, on dépile le nombre d'opérandes nécessaires, on effectue l'opération et on remet le résultat sur la pile. Le signe = provoque l'affichage du nombre placé au sommet de la pile et le signe q permet de terminer le calcul en cours et de quitter le programme.

Exercice 3 (TP) Écrire la fonction
`void Applique(Pile p, char op)`
qui applique l'opération `op` à la pile `p`.

Exercice 4 (TD/TP) Écrire le programme du calculateur. On pourra utiliser `Deug.stringToInt` pour convertir une chaîne de caractères en entier.

Exercice 5 (TD/TP) Écrire une fonction qui permet d'afficher l'expression arithmétique correspondant au résultat affiché. Par exemple, si l'on entre successivement les éléments suivants :

```
56 23 + 8 7 * + =
```

le calculateur affichera le nombre 135 avec l'expression arithmétique correspondante :

$$(56 + 23) + (7 * 8) = 135.$$

Pour cela on pourra utiliser une pile de chaînes de caractères en plus de la pile d'entiers. Dans la pile de chaînes on ne conservera pas les résultats des opérations, mais les expressions intermédiaires.

Exercice 6 (TD) Comment implanter une pile avec une liste chaînée ?

Les files

Exercice 7 (TD) Comment implanter les files à l'aide d'un tableau ? Que doit-on vérifier ?

Exercice 8 (TD) Comment implanter les files avec des listes chaînées ? Comment réduire le nombre d'opérations ?

▶ **Exercice 9**

Exo 1, 2, 3, 4, 5 : voir code ci-dessous.

Exo 6 : empiler et dépiler en tête... normalement c'est une question de cours.

Exo 7 : on a besoin de deux indices : `debut` et `fin` de la file, qui sont incrémentés (modulo la taille du tableau) respectivement pour l'insertion et la suppression. Il faut vérifier que la fin ne dépasse pas le début.

Exo 8 : par exemple, insertion à la fin, et suppression en tête. C'est plus rapide si on garde un pointeur vers la fin de la liste.

```
import fr.jussieu.script.Deug;

class Pile{ // Pile d'entiers
    int tailleMax; // taille maximale de la pile
    int[] contenu; // éléments contenus dans la pile
    int tete; // position de la tete de la pile dans le tableau

    public Pile(int taille){
        tailleMax = taille;
        contenu = new int[tailleMax];
        tete = -1;
    }
    public static void empiler(Pile p, int e){
        // if (p.tete < p.tailleMax-1)
        p.tete++;
        p.contenu[p.tete]=e;
    }
    public static int depiler(Pile p){
        p.tete--;
        return p.contenu[p.tete+1];
    }
    public static boolean estVide(Pile p){
        return p.tete<0;
    }
    public static void lireTete(Pile p){
        Deug.print(p.contenu[p.tete]);
    }
}

class PileS{ // Pile de chaînes
    int tailleMax; // taille maximale de la pile
    String[] contenu; // éléments contenus dans la pile
    int tete; // position de la tete de la pile dans le tableau

    public PileS(int taille){
        tailleMax = taille;
        contenu = new String[tailleMax];
        tete = -1;
    }
    public static void empiler(PileS p, String e){
        // if (p.tete < p.tailleMax-1)
        p.tete++;
        p.contenu[p.tete]=e;
    }
    public static String depiler(PileS p){
        // if (p.tete>=0)
        p.tete--;
        return p.contenu[p.tete+1];
    }
    public static boolean estVide(PileS p){
        return p.tete<0;
    }
    public static void lireTete(PileS p){
        Deug.print(p.contenu[p.tete]);
    }
}

public class Calculatrice{
    static void Applique(Pile p, char op){
        int o2=Pile.depiler(p); // operande 2
```

```
        int o1=Pile.depiler(p); // operande 1
        switch(op){
        case '+' : {Pile.empiler(p,o1+o2); break;}
        case '-' : {Pile.empiler(p,o1-o2); break;}
        case '*' : {Pile.empiler(p,o1*o2); break;}
        case '/' : {Pile.empiler(p,o1/o2); break;}
        case '%' : {Pile.empiler(p,o1%o2); break;}
        default : {Deug.println("operation non valide");
        Pile.empiler(p,o2); Pile.empiler(p,o1); }
        }
    }
    static void AppliqueS(PileS ps, char op){ // Pour les piles de chaînes
        String s2=PileS.depiler(ps); // chaîne 2
        String s1=PileS.depiler(ps); // Chaîne 1
        switch(op){
        case '+' : {PileS.empiler(ps,"("+s1+" + "+s2+")"); break;}
        case '-' : {PileS.empiler(ps,"("+s1+" - "+s2+")"); break;}
        case '*' : {PileS.empiler(ps,"("+s1+" * "+s2+")"); break;}
        case '/' : {PileS.empiler(ps,"("+s1+" / "+s2+")"); break;}
        case '%' : {PileS.empiler(ps,"("+s1+" % "+s2+")"); break;}
        default : {Deug.println("operation non valide");
        PileS.empiler(ps,s2); PileS.empiler(ps,s1); }
        }
    }

    public static void main(String[] args) {
        Pile p = new Pile(15);
        PileS ps = new PileS(15);
        String entree;
        int num;
        char op;
        boolean encore = true;

        Deug.println("calculateur HP, entrez les nombres, puis l'operation");
        while(encore){
            entree = Deug.readString();
            num = Deug.stringToInt(entree);
            if (num!=0) { // si on a entré un nombre
                Pile.empiler(p,num);
                PileS.empiler(ps,Deug.intToString(num)); }
            else {
                op = Deug.charAt(entree,0);
                switch(op){
                case '0' : { // on peut aussi entrer zero
                    Pile.empiler(p,0);
                    PileS.empiler(ps,"0");
                    break;}
                case 'q' : {encore = false; break;}
                case '=' : {
                    PileS.lireTete(ps);
                    Deug.print(" = ");
                    Pile.lireTete(p);
                    Deug.println();
                    break;}
                default : {
                    Applique(p,op);
                    AppliqueS(ps,op);
                }
                }
            }
            // Pile.afficher(p); PileS.afficher(ps);
        }
    }
}
```