TP 5: Boucles et tableaux. Corrigé

Informatique Fondamentale (IF121)

16 Novembre 2003

1 Suites numériques : la suite

Exercice 1 (Puissance d'un nombre)
Il n'y a pas de difficultés particulières. La boucle `for` sert à itérer n fois le produit par l'entier dont on souhaite calculer la puissance. Pour les puissance négatives, on remarque que $x^{-n} = \left(\frac{1}{x}\right)^n$. D'où le code suivant :

```
import fr.jussieu.script.Deug;
class Puissance{
    static double puissance (double nombre, int k){
        double p=1;
        if (k >= 0){
            for (int i=1; i <= k; i++){
                p = p*nombre;
            }
        }
        else{
            for (int i=1; i <= ((-1)*k); i++){
                p = p/nombre;
            }
        }
        return p;
    }
    public static void main (String[] args){
        Deug.print("Nombre? ");
        double nombre = Deug.readDouble();
        Deug.print("Exposant? ");
        int puiss = Deug.readInt();
        Deug.println(nombre+" ^ "+puiss+" vaut: "+puissance(nombre, puiss));
    }
}
```

Exercice 2 (Une série)
La seule difficulté est la gestion des nombres impairs. Le n-ième nombre impair étant $2n - 1$, on a le code suivant :

```
import fr.jussieu.script.Deug;
class Serie2{
    static int sommeCarrésImpairs (int n){
        int somme = 0;
        for (int i=1; i <= n; i++){
            somme = somme + (2*i-1) * (2*i -1);
        }
        return somme;
    }

    public static void main (String[] args){
        Deug.print("n? ");
        int n = Deug.readInt();
        Deug.println("La somme des "+n+" premiers carrés impairs vaut: "+sommeCarrésImpairs(n));
    }
}
```

Exercice 3 (Une suite)
On calcule les termes successifs de la suite (dans la variable `x` du code ci-dessous) :

```
import fr.jussieu.script.Deug;
class Suite{
    static void suite (int a, int n){
        int x = a;
        Deug.println("x(0)="+a);
        for (int i=1; i <= n; i++){
            x = 4*x*(1-x);
            Deug.println("x(" + i + ")=" + x);
        }
    }

    public static void main (String[] args){
        Deug.print("a? ");
        int a = Deug.readInt();
        Deug.print("n? ");
        int n = Deug.readInt();
        suite(a,n);
    }
}
```

2 Affichage

Exercice 4 (Compter)

```
import fr.jussieu.script.Deug;
class Compter{
    public static void main(String[] args){
    int n=Deug.readInt();
    for (int i=1;i<=n;i++){
        Deug.println(i);
        }
    }
}
```

Exercice 5 (Un rectangle)

```
import fr.jussieu.script.Deug;
class Rectangle{

    static void repetePrint(int n,String c){
        for (int i=1;i<=n;i++){
            Deug.print(c);
        }
    }

    public static void main(String[] args){
        int L=Deug.readInt(),H=Deug.readInt();
        Deug.print("+");
        repetePrint(L,"-");
        Deug.println("+");
        for (int j=1;j<=H;j++) {
            Deug.print("+");
            repetePrint(L," ");
            Deug.println("+");
        }
        Deug.print("+");
        repetePrint(L,"-");
        Deug.println("+");
    }
}
```

3 Traitement de données

Exercice 6 (La ligne la plus longue)

```
import fr.jussieu.script.Deug;
class Compter{
    public static void main(String[] args){
        Deug.println("Combien de lignes ?");
        int n=Deug.readInt();
        String x;
        String resultat="";
        for (int j=1;j<=n;j++) {
            x=Deug.readString();
            if (Deug.length(x)>Deug.length(resultat)){
                resultat=x;
                }
            }
        Deug.println("Ligne la plus longue : "+resultat);
    }
}
```

En cas de longueurs égales, le programme retourne la première chaîne de cette longueur. En remplaçant > par >=, ce serait la dernière.

Exercice 7 (Compter les voyelles)

```
public static int compte(String t){
    int resultat=0;
    char c;
    for (int j=0;j<Deug.length(t);j++) {
        c=Deug.charAt(t,j);
        if ((c=='a')||(c=='e')||(c=='i')||(c=='o')||(c=='u')||(c=='y')||
            (c=='A')||(c=='E')||(c=='I')||(c=='O')||(c=='U')||(c=='Y')){resultat++;}
        }
    return resultat;
}
```

Exercice 8 (Arguments d'un programme)

```
import fr.jussieu.script.Deug;
class Echo{
    public static void main(String[] args){
        for (int j=0;j<args.length;j++) {
            Deug.print(args[j]);
            Deug.print(" ");
        }
    }
}
```

Le tableau nommé `args` (vous pourriez d'ailleurs lui donner un autre nom) est créé automatiquement par l'interpréteur java lors de l'exécution de la commande
`java Echo truc machin chose bidule`

Dans cet exemple, l'interpéteur crée un tableau de String `args` qui a 4 cases, respectivement remplies par `truc`, `machin`, `chose`, `bidule`

Bien entendu, lors de l'exécution de la commande
`java Echo`
le tableau `args` est vide, de taille 0.

4 Tableaux de nombres

Dans toute cette partie, à défaut de pouvoir faire des `print` ou `println` sur des tableaux, on peut tester la correction des programmes en utilisant la méthode suivante pour afficher un tableau :

```
static void affiche(double[] t){
        for (int i=0; i < t.length; i++){
            Deug.print(" | "+ t[i]);
        }
        Deug.println("|");
    }
```

Exercice 9 (Lecture d'un tableau de nombres)

```
static double[] tableau (){
        int n= Deug.readInt();
        double[] t = new double[n];
        for (int i=0; i < n; i++){
            Deug.print("Réel n°"+(i+1)+" = ");
            t[i] = Deug.readDouble();
        }
        return t;
    }
```

Exercice 10 (Concaténation)
Il faut tout d'abord récupérer les tailles `n1` et `n2` des deux tableaux à concaténer. Ensuite, on crée un tableau de taille égale à la somme des tailles précédentes. Puis on parcourt le premier tableau en recopiant les éléments dans le nouveau tableau. Enfin on parcourt le second tableau et on recopie les éléments dans le nouveau tableau à la suite des autres. Cela donne alors :

```
static double[] concatene ( double[] t1, double[] t2){
        int n1 = t1.length;
        int n2 = t2.length;
        double[] concatene = new double[n1+n2];
        for (int i=0; i < n1; i++){
            concatene[i] = t1[i];
        }
        for (int i = 0; i < n2; i++){
            concatene[n1+i] = t2[i];
        }
        return concatene;
    }
```

Exercice 11 (Vérification de tri)
Il suffit de comparer chaque élément du tableau à l'élément qui le suit jusqu'à détecter éventuellement une erreur. C'est pour cela que je préfère ici une boucle `while` (à quoi bon parcourir le tableau si dès les deux premiers éléments on détecte qu'il n'est pas trié ?). Il faut faire attention également à ne pas sortir du tableau.

```
static boolean verif (double[] t){
        boolean b = true;
        int i = 0;
        while ((i<t.length-1)&& b){
            b = (t[i] < t[i+1]);
            i++;
        }
        return b;
    }
```

Notez que la méthode renvoie `true` si le tableau est trié, sinon `false`.

Exercice 12 (Fusion de deux tableaux triés)
Le premier élément du tableau fusionné est le premier du premier ou du second tableau. Le second est le second élément du tableau d'où est issu le premier élément ou le premier élément de l'autre tableau. Et ainsi de suite. Ainsi, on maintient deux compteurs `i` et `j` qui parcourent respectivement le premier et le second tableau. Ils désignent le premier élément de chaque tableau non encore mis dans le tableau fusionné. C'est donc parmi ces deux éléments qu'il faudra chercher le nouvel élément à fusionner. Voici le code qui s'en suit (ne pas hésiter à me demander de plus amples détails) :

```
static double[] fusion (double[] x, double[] y) {
    int n = x.length + y.length;
    double[] t = new double[n];
    int i = 0; //compteur pour le premier tableau
    int j = 0; //compteur pour le second tableau
    for (int k = 0; k < n; k++) {
        if (i == x.length) { // on a épuisé le premier tableau
            t[k] = y[j];
             j++;
        }
        else if (j == y.length) { // on a épuisé le second tableau
            t[k] = x[i];
            i++;
        }
        else if (x[i] <= y[j]) { // le prochain élément est dans le premier tab.
        t[k] = x[i];
            i++; // on avance dans le premier tableau.
        }
        else { // le prochain élément est dans le second tab.
            t[k] = y[j];
            j++; //on avance dans le second tableau.
        }
    }
    return t;
}
```

Exercice 13 (Norme du sup)
Pour la norme sup, il faut prévoir une variable dans laquelle on stocke le plus grand élément déjà vu et l'actualiser en parcourant le tableau. On passe par une méthode auxiliaire pour calculer la valeur absolue d'un nombre. Cela donne le code suivant :

```
static double abs (double x){
        if (x < 0)return (-x);
        else return x;
    }

static double normeSup (double[] x){
        double max = abs(x[0]);
        int n = x.length;
        for (int i = 0; i<n; i++){
            if (max < abs(x[i])){
                max = abs(x[i]);
            }
        }
        return max;
    }
}
```