TD 1 – Rappels de Java
Structures, tableaux, chaînes de caractères

Structures de données (IF 122)

1 Rappels de syntaxe

Tous les programmes que vous ferez en TP seront écrits en Java. Si un programme s'appelle "MonProgramme", il devra être écrit dans un fichier nommé « `MonProgramme.java` » et avoir la forme suivante :

```
import fr.jussieu.script.Deug;
⟨définition de classes⟩
class MonProgramme {
    ⟨définition de fonctions⟩
    public static void main (String[] args) {
        ⟨programme principal⟩
    }
}
```

2 Chaînes de caractères

Une chaîne de caractères est constituée d'une suite de caractères. Ceux-ci sont représentés en interne par des codes définis par la norme Unicode. On peut faire sur les caractères certaines opérations comme on le fait pour les entiers : 'A' $+ 3$ est égal à 'D', on peut tester si 'A' $\leqslant$ 'a', etc.

Le type des caractères en Java est « `char` », le type des chaînes de caractères est « `String` ». Par exemple `"une chaîne"` est une chaîne de type « `String` ». Les chaînes sont munies des opérations suivantes :

- la concaténation : « `"c" + s1` » qui retourne une chaîne
- calcul de la longueur : « `Deug.length(s)` » retourne la longueur de « `s` »
- le n-ième caractère est donné par : « `Deug.charAt(s, n-1)` »
- pour extraire une sous-chaîne : « `Deug.subString(s, début, fin)` »
- test d'égalité : « `Deug.equals(s1, s2)` » renvoie un booléen

Attention, comme pour les tableaux, les positions utilisées dans une chaîne de caractères ne doivent pas dépasser la longueur de la chaîne !

Exercice 1 (TD) Exécuter à la main les lignes suivantes :

```
String s;    boolean b; int k;
...
s = "Bonjour";     s = s + " tout le monde !";
Deug.println(s + " contient " + Deug.length(s) + " caractères");
Deug.println(Deug.charAt(s, 4));
Deug.println(Deug.subString(s, 3, 6));
```

Exercice 2 (TD) Ecrire une fonction « `insert` » qui prend en argument une chaîne de caractères, une position dans cette chaîne, et une seconde chaîne, et qui retourne une chaîne resultant de l'insertion de la seconde dans la premiere à la position specifiée.

Exercice 3 (TP) Écrire une fonction qui prend comme paramètre un caractère et renvoie le caractère majuscule s'il s'agit d'une lettre ou le caractère initial sinon.

Exercice 4 (TP) Écrire un programme qui transforme une chaîne de caractères en un palindrome. Par exemple, « `taratata` » sera transformée en « `taratataatatarat` ».

Exercice 5 (TD) On peut remarquer que le plus petit palindrome qui étend « `abb` » est « `abba` ». On veut écrire un programme qui transforme une chaîne de caractères en plus petit palindrome qui l'étend. Pour cela :

1. Écrire une fonction « `boolean est_palindrome(String s)` »,
2. Écrire une fonction « `int position_plus_grand_suffixe_palindrome(String s)` »,
3. Utiliser les fonctions précédentes pour résoudre le problème.

3 Tableaux

Les tableaux permettent de regrouper en une seule entité des données de type identique, y compris des structures ou même des tableaux. On déclare un tableau en écrivant

```
int[] tableau;            /* déclaration d'un tableau d'entiers */
tableau = new int[10];    /* création d'un tableau de 10 entiers */
```

Si on veut par exemple initialiser chaque case du tableau à 3, on écrit :

```
int i;
for (i = 0; i < 10; i++) {
    tableau[i] = 3;
}
```

Remarquez que la numérotation des cases du tableau va de 0 à 9 (ce qui fait 10 éléments). Lorsque les tableaux sont passés en paramètre à une fonction, ils le sont toujours par référence (on en verra l'explication dans quelques cours).

Exercice 6 (TD+TP) — *Triangle de Pascal*

1. Écrire une fonction « `triangle(a,b)` » qui, supposant que le tableau « `a` » est une ligne du triangle de Pascal, calcule la ligne suivante en la plaçant dans « `b` ».
2. Écrire une fonction « `permute_g(a)` » qui permute circulairement les éléments d'un tableau « `a` » en les déplaçant d'une case vers la gauche, le premier élément devenant le dernier.
3. Idem avec « `permute_d` » qui permute les élément du tableau de la même façon, mais vers la droite.
4. Écrire une fonction « `triangle_direct(a)` » qui calcule la ligne suivante du triangle de Pascal, sans l'aide de tableaux auxiliaires.

Exercice 7 (TP) — *Fusion* Soient deux tableaux « `a` » et « `b` » de taille n dont les éléments sont déjà triés et un tableau « `c` » de taille $2n$ vide. Écrire une fonction « `fusion(a,b,c)` » qui fusionne les éléments des tableaux « `a` » et « `b` » dans le tableau « `c` » de telle sorte que les éléments du tableau « `c` » soient eux aussi triés.

▶ Exercice 8

Ca donne :

```
Bonjour tout le monde ! contient 23 caractères
o
jou
```

▶ Exercice 9

```
static String insert (String s, int p, String s2){
    // rajouter les cas d'erreurs
    s2=Deug.subString(s,0,p) + s2 + Deug.subString(s,p+1,Deug.length(s)-1);
    return s2;
}
```

▶ Exercice 10

Avec un cast.

```
static char minToMaj(char c){
    if (('a'<= c) && (c<='z'))
return (char)(c-'a'+'A');
    else return c;
}
```

Par contre je n'ai pas reussi a transformer toute une chaîne en lui appliquant minToMaj, est ce que tu sais comment on fait ?

▶ Exercice 11

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

        public static void main (String[] args) {
    String s,pal;
    s=Deug.readLine();
    pal=s;
    for (int i=Deug.length(s)-1 ; i>=0 ; i--){
pal= pal + Deug.subString(s,i,i+1);
    }
    Deug.println(s);
    Deug.println(pal);
        }
    }
```

▶ Exercice 12

```
 class test {

static boolean est_palindrome(String s){
    boolean res=true;
    int L=Deug.length(s);
    // Rq en exercice on peut donner une solution avec un while
    for(int i=0; i<=L/2-1 ; i++){
res=res && ( Deug.charAt(s,i)==Deug.charAt(s,L-1-i) );
    }
    return res;
}


static int position_plus_grand_suffixe_palindrome(String s){
    int i;
    String s1;
    i=0;
    s1=s;
    while ( i < Deug.length(s) && !(est_palindrome(s1)) ){
i++;
s1=Deug.subString(s,i,Deug.length(s));
    }
    return i;
}

static String donnePalindrome(String s){
    String pal=s;
    int i=position_plus_grand_suffixe_palindrome(s);
    Deug.println(i);
    //qq cas particulier
    if (i != Deug.length(s)-1 | Deug.length(s)==1) i--;
    for (;i>=0;i--){
pal += Deug.charAt(s,i);
    }
    return pal;
}


        public static void main (String[] args) {
    String s,pal;
    s=Deug.readLine();
    Deug.println(est_palindrome(s));
    pal=donnePalindrome(s);
    Deug.println(pal);
        }
```

▶ **Exercice 13**

Le triangle de Pascal est la matrice des C_n^p. Elle se calcule de bas en haut avec la règle $C_{n+1}^p = C_n^{p-1} + C_n^p$.

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

static void init(int t[]){
    t[0]=1; t[1]=1;
    for (int i=2;i<t.length; i++) t[i]=0;
}

static void affiche(int [] t){
    for (int i=0; i<t.length; i++){
if (t[i]==0) break;
Deug.print(t[i]);
    }
    Deug.println();
}

static void triangle(int []t1,int []t2){
    for (int i=1; i<t1.length;i++)
t2[i]=t1[i]+t1[i-1];
    t2[0]=1;
}

        public static void main (String[] args) {
    int[] t1,t2;
    t1=new int [20];
    t2=new int [20];
    init(t1);
    init(t2);
triangle(t1,t2);
affiche(t2);
/* Question suplementaire ! Que fait le code suivant ?
t1=t2;  // ... c'est pas une copie !
triangle(t1,t2);
affiche(t2);
*/
        }
```

▶ **Exercice 14**

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

static void init(int t[]){
    for (int i=0;i<t.length; i++) t[i]=i;
}

static void affiche(int [] t){
    for (int i=0; i<t.length; i++)
Deug.print(t[i]);

    Deug.println();
}

static void permute_g(int []t){
    int aux=t[0];
    for (int i=0;i<t.length-1;i++) t[i]=t[i+1];
    t[t.length-1]=aux;
}

static void permute_d(int []t){
    int aux=t[t.length-1];
    for (int i=t.length-1;i>0;i--) t[i]=t[i-1];
    t[0]=aux;
}

        public static void main (String[] args) {
    int[] t;
    t=new int [10];
    init(t);
    affiche(t);
    permute_g(t);
    affiche(t);
    permute_d(t);
    affiche(t);
        }

    }
```

▶ **Exercice 15**

```
voir au dessus
```

▶ **Exercice 16**

La seule remarque c'est que ça se fait directement, sans rien écraser si on lit de droite à gauche.

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

static void init(int t[]){
    t[0]=1; t[1]=1;
```

```
    for (int i=2;i<t.length; i++) t[i]=0;
}

static void affiche(int [] t){
    for (int i=0; i<t.length; i++){
if (t[i]==0) break;
Deug.print(t[i]);
    }
    Deug.println();
}

static void triangle(int []t){
       for(int i=t.length-1;i>0;i--){
t[i]=t[i-1]+t[i];
    }
    t[0]=1;
}

        public static void main (String[] args) {
    int[] t;
    t=new int [20];
    init(t);
    triangle(t);
    affiche(t);

    triangle(t);
    affiche(t);

    triangle(t);
    affiche(t);

        }

    }
```

▶ **Exercice 17**

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

static void init(int t[]){
    for (int i=0;i<t.length; i++) t[i]=i;
}

static void affiche(int [] t){
    for (int i=0; i<t.length; i++)
Deug.print(t[i]);

    Deug.println();
}

static void fusionne(int []a,int []b,int []c){
     int pa,pb;
     // je ne fais pas les cas d'erreurs sur la taille de c
     pa=0;
     pb=0;

     for(int i=0;i<c.length;i++){
 if ( (pb<b.length) && (pa<a.length)){
     if ( a[pa] <= b[pb] ) {
 c[i]=a[pa];
 pa++;
     }
     else {
 c[i]=b[pb];
 pb++;
     }
 } else if (pb >= b.length) {
     c[i]=a[pa];
     pa++;
 } else {
     c[i]=b[pb];
     pb++;
 }

     }
}
```