Algorithmique et programmation avec Java : notions de base (NFA001)

Corrigé 2ème session

23 avril 2009

Tous les documents sont autorisés. Le barème est donné à titre indicatif.

Exercice 1 (4 pts)

Soit le programme suivant :

```
public class Exo2 {
   static boolean alphN(char c){
     return (c>='a' && c <= 'z') || (c>='A' && c <= 'Z') || (c>='0' && c <= '9');
   }
   static void a(int n, char c){
      for (int i = 1; i <= n; i++)  {
         Terminal.ecrireChar(c);
      }
      Terminal.sautDeLigne();
   }

   static void p(char [] t){
      for (int i = 0; i < t.length; i++)  {
          if (alphN(t[i])) {
              a(i+1, t[i]);
          } else {
              Terminal.ecrireCharln( t[i]);
          }
      }
   }

    public static void main (String args[]) {
        char [] tab = {'6', '+', 'a', 'B', '*', '4','f' };
        p(tab);
    }
}
```

1. Que fait la méthode `alphN` ?
2. Que fait la méthode `a` ?
3. Qu'affiche ce programme ?
4. Que fait ce programme pour un tableau quelconque ?

Corrigé :

1 point par bonne réponse

1. Que fait la méthode `alphN` ?
   Elle teste si un caractère est alphanumérique (lettre majuscule, minuscule ou chiffre).
2. Que fait la méthode `a` ?
   Elle affiche n fois le caractère c passé en paramètre, puis va à la ligne.
3. Qu'affiche ce programme ?

```
6
+
aaa
BBBB
*
444444
fffffff
```

4. Que fait ce programme pour un tableau quelconque ?
   Il affiche sur autant de lignes que le tableau à d'éléments : soit le caractère de la case i du tableau est un caractère alphanumérique, et dans ce cas, il est affiché i fois sur une même ligne ; sinon il est affiché une seule fois.

Exercice 2 (4 points)

Le but du programme suivant est de tester si un tableau d'entiers saisit au clavier est trié dans l'ordre croissant.

```java
public class TableauTrie {
    static void estTrie(int [] t){
        int x = t[1];
        for (int j=1; j<= t.length; j++){
            if (x > t[j]) {
               return false;
            } else {return true};
        }
        return true;
    }
    public static void main (String args[]) {
        int n;
        int [] tab;
        Terminal.ecrireString("Taille du tableau? ");
        n = Terminal.lireInt();

        for (int i = 0; i <= tab.length -1; i++)  {
             Terminal.ecrireString("Un entier? ");
             tab[i] = Terminal.lireInt();
        }
        boolean b = estTrie();
        if (b) {
            Terminal.ecrireStringln("Tableau trie ");
        }
        Terminal.ecrireStringln("Tableau non trie ");
```

```
26        }
27  }
```

Ce programme contient 3 erreurs détectés à la compilation, ainsi que plusieurs autres erreurs de conception. Ces dernières font que le programme (une fois les erreurs de compilation corrigées) ne se comporte pas correctement.

1. Trouver et corriger les erreurs de compilation : donnez la ligne de chaque erreur de compilation, une description succinte de l'erreur, et une correction qui permet au programme de compiler.
2. Trouver et corriger les erreurs de conception : donnez la, ou les lignes de chaque erreur de conception, une description succinte de l'erreur, et une correction pour que le comportement du programme soit correct.

Corrigé

(0.5 pt) par erreur trouvée et corrigéé. Indulgence sur la distinction entre erreur de compilation et de conception, pourvu qu'il les aient trouvé et corrigé.

1. ligne 2 : mauvais type de retour. Correction :
   `static boolean estTrie(int [] t){`
2. ligne 3 : mauvaise initialisation de x. Correction :
   `int x = t[0];`
3. ligne 4 : dépassement de l'indice le plus grand du tableau. Correction :
   `for (int j=1; j<t.length; j++)`
4. ligne 7 : il est incorrect de retourner true. Correction : éliminer
   `else {return true;}`
5. ligne 8 : avant la fin de la boucle, on doit mettre à jour la valeur de x. Correction :
   `x = t[j];`
6. ligne 16 : manque l'initialisation du tableau. Correction :
   `tab = new int[n];`
7. ligne 21 : il manque le paramètre. Correction :
   `boolean b = estTrie(tab);`
8. ligne 24 : manque le "else". Correction :
   `else { Terminal.ecrireStringln("Tableau non trie ");}`

Exercice 3 (5 points)

On veut écrire une fonction qui prend deux tableaux d'entiers, que l'on suppose tous les deux triés dans l'ordre croissant. On souhaite écrire une fonction `merge` qui doit renvoyer la réunion de deux tableaux triée dans l'ordre croissant. Par exemple si les tableaux contiennent chacun t = 3,5,25,30 et r= 1,2,3,4,8,28, la fonction `merge(t,r)` doit renvoyer le tableau constitué de 1,2,3,3,4,5,8,25,28,30.

Indication : Pour cela, il suffit de parcourir les deux tableaux au même temps, en insérant dans un troisième tableau, l'élément le plus petit parmi les deux cases comparées pour les deux tableaux. Attention à bien gérer les indices de chaque tableau.

Correction

- Entête correcte de la méthode + initialisations tableaux et indices : 1.5 pt.
- boucle principale : 2 pts.
- boucles de parcours des derniers éléments + return : 1.5 pts.

```java
public class MergeSort {
    static int []  merge(int [] t, int [] r){
        int [] res = new int [t.length + r.length];
        int i =0;
        int j=0;
        int k=0;

        while (i < t.length && j < r.length){
            if (t[i]<= r[j]) {
                res[k] = t[i];
                i++;
            } else {
                res[k] = r[j];
                j++;
            }
            k++;
        }
        for (int p = i; p < t.length; p++, k++)  {
            res[k] = t[p];
        }
        for (int p = j; p < r.length; p++, k++)  {
            res[k] = r[p];
        }
        return res;
    }
    public static void main (String args[]) {
        int [] t1 = {1,4,7, 20};
        int [] t2 = {1,2,3,4,10, 27, 28};
        int [] t3;
        t3 = merge(t1,t2);
        for (int i = 0; i <= t3.length -1; i++)  {
            Terminal.ecrireIntln(t3[i]);
        }
    }
}
```

Exercice 4 *(6 pts)*

On souhaite représenter les notes d'un ensemble d'étudiants par deux tableaux de même taille, l'un contenant leurs noms et l'autre contenant leurs notes. Le tableau des noms s'appellera tNoms et celui des notes tNotes. Exemple :

Tableau de noms :

Bernard	Dupont	Zidane	Henry	Barthez

Tableau de notes :

12,5	5	10	7	4

Les noms et les notes sont dans le même ordre. On sait ainsi que Zidane et sa note ont le même indice dans les tableaux respectifs et on peut déduire que la note de Zidane est 10, de même, la note de Barthez est 4, etc.

Écrire un programme avec les méthodes et actions suivantes :

1. Une fonction `trouveIndice` qui étant donnée un nom et un tableau de noms retourne l'indice de ce nom dans le tableau. Si le nom ne figure pas dans le tableau, la fonction doit renvoyer -1. Afin de comparer les chaînes de caractères, vous utiliserez la fonction prédéfinie `equals` qui renvoie un boolean, et qui étant donnés 2 chaînes `s1` et `s2`, permet de les comparer par `s1.equals(s2)`.
2. Ecrire une fonction `trouveNote` qui étant donnés un nom, un tableau de noms et un tableau de notes, retourne la note correspondant au nom donné. Cette fonction doit obligatoirement utiliser la fonction `trouveIndice`. Si le nom n'est pas trouvé, on doit renvoyer -1 (qui n'est pas une valeur possible de note).
3. Ecrire un programme principal qui déclare et initialise les tableaux tNotes et tNoms comme dans l'exemple. Ensuite le programme demande à l'utilisateur le nom d'un étudiant et lui fournit sa note. Utiliser obligatoirement la fonction `trouveNote`. Ce programme principal doit traiter le cas d'un nom d'étudiant ne figurant pas sur la liste en affichant un message à l'écran. Vous utiliserez obligatoirement le fait que la fonction `trouveNote` renvoie -1 dans ce cas là.