IPST-CNAM
Programmation JAVA
NFA 001
Mercredi 22 Juin 2011

Avec document
Durée : **2h30**
Enseignant : LAFORGUE Jacques

2ème Session NFA 001

(COURS)

La durée de l'examen est de 2 heures et 30 minutes. Les documents sont autorisés.. Le barème est indicatif.
La réponse doit être sur papier. Vous devez écrire les codes commentés en Java.

1. Probleme 1 [20mn] [2 points]

```
// Ajoute t2 dans t1 en gardant l'ordre croissant de t1
static public void ajouterTabOrd(int[] t1,
                    int nb1,
                    int[] t2,
                    int nb2) throws Exception
{
   // Si t1 n'est pas assez grand alors exception
   if (t1.length < nb1 + nb2)
      throw new Exception("t1 pas assez grand");

   // Pour chaque element de t2
   for(int p=0;p<nb2;p++)
      {
         // On recherche le 1er element immediatement superieur
         boolean ajoute=false;
         for(int i=0;i<nb1;i++)
            {
               if(t1[i]>t2[p])
                  {
                     // On décale pour insérer le nouvel élément
                     for(int k=nb1;k>=i;k--)
                        t1[k+1]=t1[k];
                     t1[i]=t2[p];
                     nb1++;
                     ajoute=true;
                     break;
                  }
            }
         // Si on n'a pas trouvé de 1er element immediatement superieur
         // alors il fat ajouter le nouvel element à la fin de t1
         if (!ajoute)
            {
               t1[nb1]=t2[p];
               nb1++;
            }
      }  }
```

2. Problème 2 [30mn] [4 points]

```
// Ajoute un nouveau mot dans tabT
static public void ajouter(String[][] tabT, String mot)
{
   // S'il n'existe pas déjà de mot de même longueur
   if (tabT[mot.length()-1]==null)
```

```
            {
               // Création du tableau devant contenir le mot
               tabT[mot.length()-1]=new String[200];
               String[] tabRn = tabT[mot.length()-1];
               // Ajout du mot
               tabRn[0]=mot;
            }
         else
            {
               // Le tableau contenant le mot est celui en N-1 de tabT
               String[] tabRn = tabT[mot.length()-1];
               // Recherche de la 1ere case null
               for(int i=0;i<tabRn.length;i++)
                  if (tabRn[i]==null)
                     {
                        // Ajout du mot
                        tabRn[i]=mot;
                        break; // On arrête la boucle
                     }
            }
      }

      // Recherche d'un mot dans tabT
      static public void chercher(String[][] tabT, String mot)
      {
         // Si le tableau des mots de longueur du mot à chercher est
         //  vide alors non trouvé
         if (tabT[mot.length()-1]==null)
            {
               System.out.println("Non trouvé");
            }
         else
            {
               // Le tableau dans lequel il faut rechercher
               String[] tabRn = tabT[mot.length()-1];
               // On recherche le mot dans le tableau des mots
               //  de longueur N
               boolean trouve=false;
               for(int i=0;i<tabRn.length;i++)
                  if (tabRn[i]!=null)
                     if (tabRn[i].equals(mot))
                        {
                           System.out.println("Trouve");
                           trouve=true;
                           break;
                        }
               if(!trouve)
                  System.out.println("Non trouvé");
            }
      }
```

Tournez la page ➔

3. Problème 3 [60mn] [8 points]

```
import java.util.*;

public class Probleme3
{
   public static void main(String[] args)
   {
      // Lecture du fichier et recupération de chacune des lignes dans un
      //  tableau de chaine
      StringBuffer sbf = Terminal.lireFichier("CompteBancaire.txt");
      String[] lignes = sbf.toString().split("\n");

      // Les valeurs à calculer
      double solde=0.0;      // Solde du compte
      double somme_cb=0.0;   // Somme des CB
      double somme_ch=0.0;   // Somme des chèques
      String[] origine_prlvt = new String[100];  // Le nom d'une origine de prlvt
      double[] valeur_prlvt  = new double[100];  // La somme des prlvt d'une origine
      int      nb_prlvt = 0;  // Nb d'origine de prlvt

      // Parcours des lignes
      for(int i=0;i<lignes.length;i++)
         {
            // Récupération de chacun des mots de la ligne dans une
            //   StringTokenizer
            StringTokenizer strtok=new StringTokenizer(lignes[i]);

            // Si 1ere ligne alors c'est le solde initial du compte
            if(i==0)
               {
                  strtok.nextToken();  // On saute le mot SOLDE
                  solde = Double.parseDouble(strtok.nextToken());
               }
            else
               {
                  // On récupère les 3 éléments de la ligne
                  String elt1 = strtok.nextToken();     String elt2 = strtok.nextToken();
                  String elt3 = strtok.nextToken();

                  // Valeur de la transaction
                  double val = Double.parseDouble(elt3);

                  // Si un DEBIT de CB ou Chèque
                  if (elt1.equals("DEBIT"))
                     {
                        if (elt2.equals("CB")) somme_cb=somme_cb-val;
                        else somme_ch=somme_ch-val;
                        solde=solde-val;
                     }
                  // Si un CREDIT de CB ou Chèque
                  if (elt1.equals("CREDIT"))
                     {
                        if (elt2.equals("CB")) somme_cb=somme_cb+val;
                        else somme_ch=somme_ch+val;
                        solde=solde+val;
                     }
                  // Si un Prélèvement
                  if (elt1.equals("PRLVT"))
                     {
```

```
                        // On recherche si on n'a pas déjà rencontré
                        //  cette origine de prlvt
                        int indice = -1;
                        for(int k=0;k<nb_prlvt;k++)
                           if (origine_prlvt[k].equals(elt2))  indice=k;
                        // Si non trouvé l'origine
                        if(indice==-1)
                           {
                              // On mémorise l'origine et sa somme initial
                              origine_prlvt[nb_prlvt]=elt2;
                              valeur_prlvt[nb_prlvt]=val;
                              nb_prlvt++;
                              solde=solde-val;
                           }
                        else
                           {
                              // On incrémente la somme de l'origine
                             valeur_prlvt[indice] = valeur_prlvt[indice]+val;
                             solde=solde-val;
                           }
                     }
                  }
               }
         // Affichage des informations calculées
         System.out.println("Solde final = "+solde);
         System.out.println("Detail CB   = "+somme_cb);
         System.out.println("Detail CH   = "+somme_ch);
         for(int i=0;i<nb_prlvt;i++)
            System.out.println("PLVT "+ origine_prlvt[i]+" "+ valeur_prlvt[i]);
      }
   }
```

---

4. Problème 4 [40mn] [6 points]

```
public class Probleme4
{
   public static void main(String[] args)
   {
      // Les numeros des tables sont saisis sur 1 seul ligne.
      // On décode et on les mets dans un tableau
      Terminal.ecrireString("Entrez les numeros de table:");
      String numeros = Terminal.lireString();
      StringTokenizer strtok = new StringTokenizer(numeros);
      int[] tables = new int[8];
      int nbt = 0;
      while(strtok.hasMoreElements())
         {
            int num = Integer.parseInt(strtok.nextToken());
            tables[nbt]=num;
            nbt++;
         }

      // Nombre de multiplication à trouver
      Terminal.ecrireString("Nombre de proposition:");
      int nbProposition = Terminal.lireInt();
      int nbJuste=0;  // Les nbre de réponses justes

      // On boucle sur le nombre de proposition
      for(int i=0;i<nbProposition;i++)
         {
```

```
            int i1 = aleat(8)+2;  // Tirage aléatoire de 2 à 9
            int i2 = aleat(nbt);  // Tirage aléatoire de l'indice
            // des numéros de tables

            int v1 = i1;
            int v2 = tables[i2];
            Terminal.ecrireString(v1 + " X " + v2 + " = ");
            int resultat = Terminal.lireInt();
            if (resultat==v1*v2)
               {
                  Terminal.ecrireStringln("JUSTE");
                  nbJuste++;
               }
            else
               {
                  Terminal.ecrireStringln("FAUX : "+v1*v2);
               }
         }
      // Affichage du resultat ramené à une note /20
      double note = 20 * ((double)nbJuste / (double)nbProposition);
      Terminal.ecrireStringln("Note : "+note+" /20");

   }
```