Lucrare de laborator nr. 7
Şiruri de caractere (II)

1. Scopul lucrării

În această lucrare se studiază vectorii de şiruri şi se studiază modul de extragere pentru cuvintele dintr-o propoziţie, folosind clasa *StringTokenizer.*

2. Breviar teroretic

Vectorii de şiruri se declară la fel ca si vectorii de primitive. Astfel, vectorul s[ ] de obiecte *String,* ce conţine trei nume, se poate instanţia astfel:

String s[]={"Ionescu Ion","Popescu George","Vasile Marius"};

Pentru a extrage atomii lexicali dintr-un string, se foloseşte clasa **StringTokenizer,** definită în pachetul **java.util**.

Constructori:

public StringTokenizer(String s)

Este folosit atunci când separatorul implicit dintre atomi este caracterul spaţiu.

public StringTokenizer(String s, String separator)

Acest constructor se folosește când utilizăm alt caracter ca separator (diferit de spațiu), caracter ce va fi dat ca parametru.

Exemplu:

În stringul "23**:**20**:**15", caracterul de separație este **:**

Metode:

a) **public int countTokens( )**

Returnează numărul de atomi din string.

b) **public String nextToken( )**

Returnează atomul curent din string.

c) **public boolean hasMoreTokens( )**

Returnează *true* dacă mai sunt atomi neextraşi din şir.

3. Probleme rezolvate

Problema 1

Se citeşte un număr natural N. Se citesc de la tastatură N cuvinte, într-un vector de şiruri. Să se afişeze dacă toate cuvintele sunt diferite între ele sau nu.

```
import javax.swing.*;
class VectorCuvinteDiferite
{
 public static void main(String args[])
 {
  int N=Integer.parseInt(JOptionPane.showInputDialog("N = "));
  String s[]=new String[N];
  int i,j;
  for(i=0;i<N;i++)
    s[i]=JOptionPane.showInputDialog("cuvant = ");
  //Sunt diferite?
  for(i=0;i<N-1;i++)
     for(j=i+1;j<N;j++)
        if(s[i].compareTo(s[j])==0){
           System.out.println("Nu sunt toate diferite . ");
           return;
        }
   System.out.println("Sunt toate diferite ");
 }
}
```

Problema 2

Se citeşte un număr natural N. Se citesc de la tastatură N nume de elevi şi nota fiecăruia dintre ei. Să se afişeze clasamentul acestor elevi (sortare crescătoare după note).

```
import javax.swing.*;
class SortareDupaNote
{
 public static void main(String args[])
 {
  int N=Integer.parseInt(JOptionPane.showInputDialog("N = "));
  String nume[]=new String[N];
```

```
 int nota[]=new int[N];
 int i,j;
 for(i=0;i<N;i++){
  nume[i]=JOptionPane.showInputDialog("nume elev "+(i+1)+"=");
  nota[i]=Integer.parseInt(JOptionPane.showInputDialog(
            "nota elev "+(i+1)+" = "));
 }
 //Sortarea prin interschimbare a notelor:
 for(i=0;i<N-1;i++)
   for(j=i+1;j<N;j++)
     if(nota[i]>nota[j]){
       //comutam pe nota[i] cu nota[j]:
       int temp=nota[i];
       nota[i]=nota[j];
       nota[j]=temp;
       //comutam si numele:
       String aux=nume[i];
       nume[i]=nume[j];
       nume[j]=aux;
     }
 //Afisare vector sortat:
 for(i=0;i<N;i++)
   System.out.println(nume[i]+" : "+nota[i]);
}
}
```

Problema 3

Se citesc de la tastatură sub formă de şiruri de caractere doi timpi, în formatul hh:mm:ss (ore: minute: secunde). Să se afişeze care timp este mai mare.

Exemplu:

T1= 5:35:42

T2= 5:18:50

Se va afişa: T2 > T1

```
import javax.swing.*;
import java.util.*;
class ComparaTimpi
{
 public static void main(String args[])
```

```
{
 String timp1=JOptionPane.showInputDialog("timp1 (hh:mm:ss)=");
 String timp2=JOptionPane.showInputDialog("timp2 (hh:mm:ss)=");
 //Extragem din fiecare timp, orele, minutele si secundele.
 // Folosim clasa StringTokenizer pentru a extrage acesti atomi.
 StringTokenizer tk=new StringTokenizer(timp1,":");
 int ore1=Integer.parseInt(tk.nextToken());
 int min1=Integer.parseInt(tk.nextToken());
 int sec1=Integer.parseInt(tk.nextToken());
 //Calculam primul timp, in secunde:
 int T1=3600*ore1+60*min1+sec1;
 //Similar, pentru timp2 :
 tk=new StringTokenizer(timp2,":");
 int ore2=Integer.parseInt(tk.nextToken());
 int min2=Integer.parseInt(tk.nextToken());
 int sec2=Integer.parseInt(tk.nextToken());
 int T2=3600*ore2+60*min2+sec2;
 if(T1>T2)System.out.println("timp1 > timp2");
 else if(T1==T2)System.out.println("timp1 = timp2");
 else System.out.println("timp1 < timp2");
 }
}
```

Problema 4

Se citeşte de la tastatură un şir care conţine numai biţi (valori 0 şi 1), fiecare bit separat de următorul prin unul sau mai multe spaţii. Să se afişeze dacă sunt mai mulţi biţi de 0 sau mai multi biţi de 1.
Exemplu: s=”1 0 0 0 1”
Sunt mai multe valori de 0.

```
import javax.swing.*;
import java.util.*;
class Extrage0si1
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
  StringTokenizer tk=new StringTokenizer(s);
  int n=tk.countTokens();
  int n_0=0;
```

```
 int i;
 for(i=0;i<n;i++){
        int bit=Integer.parseInt(tk.nextToken());
        if(bit==0)n_0++;
 }
 int n_1=n-n_0;
 if(n_0 > n_1)System.out.println("Sunt mai multe valori de 0");
 else if(n_0 < n_1)System.out.println("Sunt mai multe valori de 1");
 else System.out.println("Sunt numar egal de valori de 0 si de 1");
 }
}
```

Problema 5

Se citeşte de la tastatură o propoziţie. Să se afişeze cuvântul de lungime maximă din propoziţie.

```
import javax.swing.*;
import java.util.*;
class Cuvinte_1
{
 public static void main(String args[])
 {
 String s=JOptionPane.showInputDialog("Introduceti propozitia : ");
 String cuvantMax="";//cuvantul de lungime maxima (initializare)
 int lMax=0;//lungimea maxima (initializare)
 //Extragem fiecare cuvant. Folosim clasa StringTokenizer.
 StringTokenizer tk=new StringTokenizer(s);
 int n=tk.countTokens();//numarul de cuvinte
 //Extragem fiecare cuvant si ii comparam lungimea cu lMax:
 int i;
 for(i=0;i<n;i++){
        String cuvantCrt=tk.nextToken();
        int lCrt=cuvantCrt.length();
        if(lCrt>lMax){
                cuvantMax=cuvantCrt;
                lMax=lCrt;}
 }
 System.out.println("cuvant de lungime maxima : "+cuvantMax);
 }
}
```

Problema 6

Se citeşte de la tastatură , ca String, un şir de numere întregi, separate între ele prin spaţii. Să se calculeze maximul dintre aceste numere.

```
import javax.swing.*;
import java.util.*;
class NumarMaxim
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog(
              "Introduceti un sir de numere separate prin spatii: ");
  //Extragem fiecare numar. Folosim clasa StringTokenizer.
  StringTokenizer tk=new StringTokenizer(s);
  int n=tk.countTokens();//numarul de atomi
  //Extragem primul numar si initializam
  // maximul cu el:
  int max=Integer.parseInt(tk.nextToken());
  //Extragem restul numerelor si le comparam cu max:
  for(int i=1;i<n;i++){
         int nrCrt=Integer.parseInt(tk.nextToken());
         if(nrCrt>max)max=nrCrt;
  }
    System.out.println("maxim : "+max);
 }
}
```

Problema 7

Se citeşte de la tastatură o propoziţie. Să se afişeze cuvântul din propoziţie care apare de cele mai multe ori.

Exemplu:

”El a cumparat si mere si pere si a cumparat si alte fructe.”

Cuvântul cel mai frecvent este *”si”*. Apare de 4 ori.

```
import javax.swing.*;
import java.util.*;
class Cuvinte_2
{
 public static void main(String args[])
```

```
{
String s=JOptionPane.showInputDialog("Introduceti propozitia : ");
//Extragem fiecare cuvant. Folosim clasa StringTokenizer.
StringTokenizer tk=new StringTokenizer(s);
int n=tk.countTokens();//numarul de cuvinte
int contor[]=new int[n];
int i,j;
for(i=0;i<n;i++)contor[i]=0;
//Extragem cuvintele si le memoram intr-un vector de String:
String cuvinte[]=new String[n];
for(i=0;i<n;i++)
        cuvinte[i]=tk.nextToken();
//Parcurgem vectorul de cuvinte si comparam cuvantul curent
 // (daca nu a mai fost gasit identic cu altul), cu toate de dupa el.
 // Daca gasim un alt cuvant identic, incrementam contorul si
// marcam cuvantul identic astfel incat sa nu-l mai analizam
// ulterior (vom face contorul asociat lui -1).
for(i=0;i<n-1;i++){
        if(contor[i]==-1)//a mai fost analizat cuvantul cuvinte[i]
          continue;//trece la urmatorul cuvant
        //nu a mai fost analizat cuvinte[i]:
        contor[i]=1;
        //cautam pe cuvinte[i] in restul vectorului:
        for(j=i+1;j<n;j++)
                if(cuvinte[i].compareTo(cuvinte[j])==0){
                        contor[i]++;
                        contor[j]=-1;//pentru a nu mai analiza
                                        // din nou pe cuvinte[j]
                }
}
//Calculam maximul din vectorul contor[]
// ( de fapt, indexul lui lui in vector)
int indexMaxim=0;
int contorMaxim=contor[0];
for(i=1;i<n;i++)
        if(contor[i]>contorMaxim){
                contorMaxim=contor[i];
                indexMaxim=i;
        }
System.out.println("Cel mai frecvent: " +cuvinte[indexMaxim]);
```

```
  System.out.println("numarul de aparitii = "+contorMaxim);
 }
}
```

3. Probleme propuse

Problema 1

Se citeşte un număr natural N. Se citesc de la tastatură N cuvinte, într-un vector de şiruri. Să se afişeze cuvântul de lungime maximă.

Problema 2

Se citesc de la tastatură două propoziţii în două şiruri s1 şi s2. Să se calculeze şi afişeze care sunt cuvintele comune în cele două propoziţii.
Exemplu:
”Afara este cald dar bate vantul.”
”Vara este prea cald afara.”
Sunt doua cuvinte comune: ”este” si ”cald” .

Problema 3

Se citeşte de la tastatură o propoziţie. Să se copieze toate cuvintele care încep cu o vocală, într-un vector de stringuri, şi să se afişeze acest vector.

Problema 4

Citim o propoziţie de la tastatură. Să afişăm dacă toate cuvintele sunt diferite între ele sau nu.
Exemplu:
”Ana are mere .” – sunt toate diferite.
”Ana are si mere si pere .” – nu sunt toate diferite.

Problema 5

Se citeşte de la tastatură , ca String, un şir de numere întregi, separate între ele prin spaţii. Să se calculeze maximul dintre aceste numere.