Lucrare de laborator nr. 6
Şiruri de caractere (I)

1. Scopul lucrării

În această lucrare se studiază şirurile de caractere. Se prezintă modul de instanţiere al obiectelor String şi câteva metode de bază din clasa String.

2. Breviar teroretic

În limbajul C, un string este un vector de caractere.

În limbajul Java, un string este un obiect instanţiat din **clasa String**, clasă definită în pachetul **java.lang**.

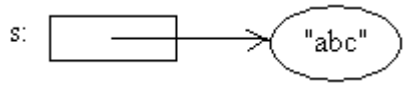
Instanţierea unui string se face în două moduri:

a) cu ajutorul operatorului **new**:

String s=new String(“abc”);

Variabila s este o variabilă referinţă ce ţine adresa obiectului string propriu-zis.

Obiectul string este alocat în memoria dinamică.



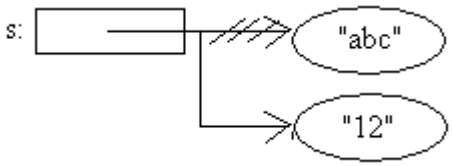
b) Al doilea mod de instanţiere: prin **atribuire directă.** Exemplu:

String s1=”1234”;

Stringurile sunt obiecte **imutabile** (nu se pot modifica).

Exemplu:

```
String s=”abc”;
    .....
s=”12”;
```



În limbajul C++, există noţiunea de **destructor** ca şi noţiune complementară constructorului. Un destructor în limbajul C++ este apelat în mod explicit pentru a elibera, în general, memoria dinamică atunci când un obiect nu mai este folosit.

În limbajul Java, nu există noţiunea de destructor, eliberarea spaţiului de memorie dinamică atunci când un obiect nu mai este necesar, se face automat de către un proces (fir de execuţie) denumit **garbage collector** (colectorul de gunoi). Acest garbage collector sesizează când un obiect nu mai este necesar şi dezalocă în mod automat spaţiul de memorie ocupat de acesta. În exemplul nostru, obiectul String inițial ”abc”, va fi automat dezalocat de către garbage collector, atunci când acesta sesisează că nu mai este referit.

În cazul stringurilor asupra cărora trebuie făcute multe modificări, se recomandă să se folosească în loc de clasa String, clasa **StringBuffer** din pachetul **java.util**, stringurile instanţiate din această clasă fiind **mutabile** (pot fi modificate).

Metode de baza din clasa String.

Cea mai importantă metodă este **metoda length( )** cu semnătura:

public int length( )

care returnează lungimea şirului.

O altă metodă importantă din clasa String este **metoda charAt( )** cu semnătura:

public char charAt(int index)

care returnează caracterul de la poziţia index.

ATENŢIE: in limbajul **C:** s[i]
In limbajul **Java:** s.charAt(i)

OBS: Pentru a citi şirul de la tastatură se foloseşte **metoda showInputDialog( )** din **clasa JOptionPane** care face parte din pachetul **javax.swing**.

Concatenarea a două şiruri:

Pentru a concatena două şiruri se foloseşte operatorul de concatenare + .

Exemplu:
String s;
String s1=”abc”;
String s2=”12”;
s=s1+s2; //s←“abc12”.

Operatorul de concatenare + este un operator binar în care unul dintre operanzi trebuie să fie în mod obligatoriu String, cel de-al doilea poate să fie şi diferit de String, dar va fi convertit în mod automat, dacă este posibil, la tipul String.
Exemplu:
String s=”abc”;
int nr=15;
String rezultat=s+nr; // ”abc15”.

Compararea a două şiruri.

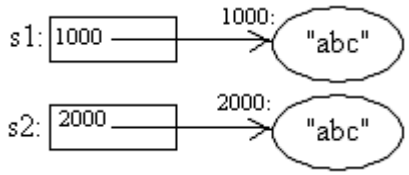
Sunt trei posibiltăţi de a compara două şiruri:

a) cu ajutorul operatorului = =
b) cu ajutorul metodei **equals( )**
c) cu ajutorul metodei **compareTo( )**

a) Compararea a două şiruri cu ajutorul operatorului = =

Se compară referințele, nu conținutul !
Exemplu:

```
String s1=”abc”;
String s2= new String(”abc”);
if (s1= =s2) System.out.println(”DA”);
else System.out.println(”NU”);//Programul afişează: NU
```



Am opresupus că obiectul s1 este memorat la adresa 1000, iar s2 la adresa 2000.

Pentru a compara două stringuri ca şi conţinut se foloseşte metoda **equals( )** sau metoda **compareTo( )**.

b) Compararea a două şiruri cu ajutorul metodei equals()

Un șir este obiectul cu ajutorul căruia se apelează metoda, iar celălalt șir este dat ca parametru. Metoda returnează *true* dacă cele două șiruri au același conținut.

Exemplu:

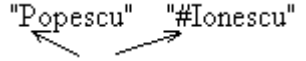
```
String s1=”abc”;
String s2= new String(”abc”);
if (s1.equals(s2)= =true) System.out.println(”DA”);
else System.out.println(”NU”);//Programul afişează: DA
```

c) Compararea a două şiruri cu ajutorul metodei compareTo( )

Metoda **compareTo( )** este definită în clasa String şi are următoarea semnătură:

public int compareTo(String s)

Compară şirul curent cu şirul dat ca parametru din punct de vedere al ordinii lexicografice. Ordinea lexicografică este o generalizare a ordinii alfabetice, în care două şiruri sunt comparate caracter cu caracter, pe baza codurilor **Unicode** ale caracterelor.



Metoda **compareTo( )** scoate un rezultat 0, dacă cele două şiruri sunt egale(au acelaşi conţinut). Dacă şirul curent este mai mare decât şirul dat ca parametru, în sensul ordinii lexicografice, metoda compareTo( ) scoate un rezultat mai mare ca 0. Dacă şirul curent este mai mic decât şirul dat ca parametru, metoda scoate un rezultat mai mic ca 0.

Exemplu:

```
String s1=”Abc”;
String s2=”Aa”;
int rez=s1.compareTo(s2); // >0
```

3. Probleme rezolvate

Problema 1

Se citesc trei şiruri de la tastatură ce reprezintă ora, minutul şi secunda orei curente. Să se construiască un şir rezultat în care timpul curent este memorat în formatul *hh:mm:*ss . Să se afişeze acest timp.

Exemplu:

Daca h=12, min=10, sec=20, atunci timpul rezultat este:
rezultat=”12:10:20”

```
import javax.swing.*;
class Concatenare
{
 public static void main(String args[])
{
  String h,min,sec;
  h=JOptionPane.showInputDialog("ora = ");
  min=JOptionPane.showInputDialog("minute = ");
  sec=JOptionPane.showInputDialog("secunde = ");
  String rezultat="";
  rezultat=h+":"+min+":"+sec;
  System.out.println("Timpul = "+rezultat);
 }
}
```

Problema 2

Se citeşte un şir. Să se afişeze dacă primul caracter este egal cu ultimul caracter.

```
import javax.swing.*;
class PrimUltimEgale
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
    if(s.charAt(0)==s.charAt(s.length()-1))
    System.out.println("Primul caracter este egal cu ultimul caracter");
    else System.out.println
                        ("Primul caracter este diferit de ultimul caracter");
 }
}
```

Problema 3

Se citeşte un şir. Să se afişeze dacă are toate caracterele egale între ele. Se va folosi o metodă separată: *areToateCarEgale()*.
Exemplu:

Şirul s=”aaaaaa” are toate caracterele egale între ele.

Şirul s= ”abbb” nu le are .

```
import javax.swing.*;
class ToateCarEgale
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
  if(areToateCarEgale(s)==true)
   System.out.println("Are toate caracterele egale");
  else System.out.println("Nu are toate caracterele egale");
 }
 private static boolean areToateCarEgale(String s)
 {
  //Le comparam pe toate, cu primul caracter:
   for(int i=1;i<s.length();i++)
    if(s.charAt(i)!=s.charAt(0))return false;
  return true;
 }
}
```

Problema 4

Se citeşte un şir. Să se afişeze dacă are toate caracterele diferite între ele. Se va folosi o metodă separată: *suntToateDiferite().*
Exemplu:

Şirul s=”abcd” are toate caracterele diferite între ele.

Şirul s= ”aba” nu le are .

```
import javax.swing.*;
class ToateCarDiferite
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
  if(suntToateDiferite(s)==true)
   System.out.println("Are toate caracterele diferite");
  else System.out.println("Nu are toate caracterele diferite");
 }
 private static boolean suntToateDiferite(String s)
 {
```

```
 //Comparam caracterul curent, cu toate caracterele de dupa el:
 for(int i=0;i<s.length()-1;i++)
   for(int j=i+1;j<s.length();j++)
      if(s.charAt(i)==s.charAt(j))return false;
 return true;
 }
}
```

Problema 5

Se citeşte un şir. Să se afişeze dacă şirul conţine numai vocale. Se va folosi o metodă separată: *suntNumaiVocale().*

```
import javax.swing.*;
class SuntNumaiVocale
{
 public static void main(String args[])
 {
 String s=JOptionPane.showInputDialog("sir = ");
 System.out.println("are numai vocale = "+suntNumaiVocale(s));
 }
 private static boolean suntNumaiVocale(String s)
 {
  for(int i=0;i<s.length();i++)
    if(esteVocala(s.charAt(i))==false)return false;
  return true;
 }
 private static boolean esteVocala(char ch)
 {
  if((ch=='a')||(ch=='A')||
    (ch=='e')||(ch=='E')||
    (ch=='i')||(ch=='I')||
    (ch=='o')||(ch=='O')||
    (ch=='u')||(ch=='U'))return true;
  else return false;
 }
}
```

Problema 6

Se citeşte sub formă de şir de caractere un număr natural foarte lung. Să se afişeze dacă este divizibil cu 3 sau nu.

Se va scrie o metodă separată ce are ca parametru de intrare şirul de caractere şi care scoate ca rezultat valoarea *true* dacă numărul este divizibil cu 3.

Exemplu:

Şirul s=”12332196” reprezintă un număr divizibil cu 3.

Şirul s=”124” nu reprezintă un număr natural divizibil cu 3.

```
import javax.swing.*;
class EsteDivizibilCu3
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
  boolean este=esteDivizibil(s);
 if(este)System.out.println("este");
 else System.out.println(" nu este");
}

private static boolean esteDivizibil(String s)
{
 //Calculam suma cifrelor numarului:
 int suma=0;
 int i;
 for(i=0;i<s.length();i++)
   if(!esteCifra(s.charAt(i)))return false;
   else suma=suma+(int)s.charAt(i)-(int)'0';
 if(suma%3==0)return true;
 else return false;
}
 private static boolean esteCifra(char ch)
 {
int cod=(int)ch;
if( (cod>=(int)'0') && ( cod<=(int)'9') )
        return true;
else return false;
 }
}
//Observatie: se putea folosi si metoda statica isDigit() din
// clasa Character
```

Problema 7

Se citeşte de la tastatură un şir de caractere. Să se copieze caracterele şirului într-o matrice pătrată de dimensiune corespunzător calculată. Poziţiile neocupate din matrice (dacă există) se vor completa cu caracterul * (asterisc). Să se afişeze matricea.

Exemplu:

Dacă şirul este ”carte”, are lungimea 5, deci matricea trebuie să aibă dimensiunea 3x3, şi se va obţine:

c a r
t e *
* * *

```
import javax.swing.*;
class Copiere
{
 public static void main(String args[])
 {
  String s=JOptionPane.showInputDialog("sir = ");
  int L=s.length();//lungime sir
  int N;
  //calcul N, dimensiune matrice patrata in care copiem sirul:
  //este L patrat perfect?
  int radical=(int)Math.sqrt(L);
  if(radical*radical==L)
        N=radical;//este patrat perfect
  else N=radical+1;
  //instantiere matrice:
  char a[][]=new char[N][N];
  int i,j;
  int k=0;//index in ;ir
  for(i=0;i<N;i++)
        for(j=0;j<N;j++)
          //s-a terminat sirul de copiat?
          if(k>=L)a[i][j]='*';
          else {
                a[i][j]=s.charAt(k);
```

```
                k++;
    }
 //afisare matrice:
 for(i=0;i<N;i++){
        for(j=0;j<N;j++)
          System.out.print(a[i][j]+" ");
        System.out.println();
 }
}
}
```

Problema 8

Se citeşte un număr natural N. Se citesc N nume de elevi. Care este numele elevului care este ultimul la catalog ?

```
import javax.swing.*;
class UltimLaCatalog
{
 public static void main(String args[])
 {
  int N=Integer.parseInt(JOptionPane.showInputDialog("N = "));
  String numeCrt;
  //Citim separat primul nume si cu el il initializam pe ultimul:
  String numeUltimLaCatalog=JOptionPane.showInputDialog(
                                  "nume = ");
  //citim restul numerelor:
  int i;
  for(i=1;i<N;i++){
   numeCrt=JOptionPane.showInputDialog("nume = ");
   //comparam numele curent cu numeUltimLaCatalog:
   if(numeCrt.compareTo(numeUltimLaCatalog)>0)
     //numele curent este dupa numeUltimLaCatalog,
     // deci il memorez pe numeCrt:
     numeUltimLaCatalog=numeCrt;
  }
  //Afisam rezultatul:
  System.out.println("Numele ultim = "+numeUltimLaCatalog);
 }
}
```

3. Probleme propuse

Problema 1

Se citeşte un şir. Să se afişeze dacă este palindrom (simetric faţă de mijloc).

Exemplu:

Şirul ”cojoc” este palindrom.

Şirul ”casa” nu este.

Problema 2

Citim un şir de caractere. Să se afişeze care vocală apare de cele mai multe ori în şir.

Exemplu:

Dacă şirul este ”toamna” , vocala a apare de cele mai multe ori.

Problema 3

Se citește de la tastatură un șir de caractere s1. Să se construiască șirul s2, ce conține toate vocalele din șirul s1.

Exemplu:

Dacă șirul s1=”orar” , atunci șirul s2=”oa” .

Problema 4

Se citeşte un număr natural N. Se citesc N nume de elevi şi pentru fiecare elev se citeşte şi câte o notă. Care este numele elevului ce are cea mai mare notă ? Dacă sunt mai mulţi elevi ce au aceeaşi notă maximă, se va afişa doar numele primului.

Problema 5

Se citeşte un şir de caractere s1. Să se construiască şirul s2 care reprezintă criptarea cu metoda Caesar a şirului s1.

Se criptează doar literele, restul caracterelor se lasă neschimbate. Fiecare literă se va substitui cu litera aflată la distanţă de 3 litere de ea. Astfel litera A se substituie cu litera D, B cu E, etc. Pentru ultimele 3 litere ale alfabetului avem substituţiile următoare: X cu A, Y cu B şi Z cu C.

Exemplu:

Daca s1=”A+B=Y” atunci s2=”D+E=B” .

Problema 6

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