# Laborator 2:

## Instrucţiuni Java şi lucru cu şiruri de caractere

Întocmit de: Adina Neculai

Îndrumător: Asist. Drd. Gabriel Danciu

18 octombrie 2011

# I. NOŢIUNI TEORETICE

## A. Instrucţiuni condiţionale

### *1. Intrucţiunea if else*

Forma generală:

**if(expresie_condiţională)**

**secvenţă_if;**

**else**

**secvenţă_else;**

Dacă *expresie_condiţională* este evaluată la *true*, atunci se execută *secvenţă_if*. În caz contrar, se execută *secvenţă_else*. Este necesar ca *expresie_condiţională* să fie evaluată la o valoare booleană. În plus, prezenţa structurii *else* nu este obligatorie şi pot exista cazuri de folosire a instrucţiunii *if else* în cascadă.

### *2. Intrucţiunea switch*

Forma generală:

**switch(expresie_condiţie) {**

**case val1:**

**secvenţa1;**

**<break>;**

**. . . ;**

**case valN:**

**secvenţaN;**

**<break>;**

**<default:>**

**secvenţa_default;**

**}**

Pentru utilizarea acestei instrucţiuni este necesar ca atât tipul de date al lui *expresie_condiţie* cât şi cel al valorilor *val1, . . . , valN* să fie din categoria tipurilor numerice: byte, char, short,

int. Astfel că, primul pas constă în evaluarea valorii expresiei *expresie_condiţie*. Apoi se compară valoarea evaluată cu prima valoare a lui *case*, val1. Dacă această valoare este egală cu *val1* atunci se execută *secvenţa1* până la întâlnirea lui *break*. Dacă instrucţiunea *break* nu este prezentă, atunci se trece la execuţia celorlalte secvenţe de tipul *secvenţaN*, fără a mai testa celelalte valori din *case*.

În cazul în care valoarea din condiţie nu este egală cu nici o valoare din *case* atunci se execută *secvenţa_default* ce urmează lui *default:*. Cazul *default* permite executarea secvenţei de după el indiferent de valoarea expresiei de evaluat. Acesta poate lipsi.

## B. Instrucţiuni de ciclare

### *1. Intrucţiunea for*

Forma generală:

**for(<secvenţă_iniţializare>;<expresie_condiţie>;<secvenţă_incrementare>)**

**<secvenţă_repetată>;**

Intrucţiunea *for* face parte din instrucţiunile de ciclare cu test iniţial. De reţinut că *secvenţă_repetată* se execută înaintea *secvenţă_incrementare* şi că variabilele definite în *secvenţă_iniţializare* sunt valabile doar în interiorul for-ului.

Tot ce este între <> poate lipsi, astfel încât putem avea **for(;;)** pentru a putea realiza un ciclu infinit.

### *2. Intrucţiunea while*

Forma generală:

**while(expresie_condiţie)**

**secvenţă_repetată;**

În execuţia instrucţiunii *while*, la început, se evaluează *expresie_condiţie*. Dacă aceasta are valoarea *true* atunci se trece la execuţia *secvenţă_repetată*. Dacă valoarea este de la bun început *false*, secvenţa din cadrul buclei nu va mai fi deloc executată. Se observă că şi această instrucţiune face parte din cele de ciclare cu test iniţial.

#### 3. *Intrucţiunea do while*

Forma generală:

**do**

**secvenţă_repetată**

**while(expresie_condiţie);**

În execuţia instrucţiunii *do while*, întâi se execută *secvenţă_repetată* şi abia apoi se evaluează *expresie_condiţie*. Astfel, chiar dacă expresia de evaluat este falsă, *secvenţă_repetată* tot se execută măcar o dată. De aceea, instrucţiunea *do while* face parte din cele de ciclare cu test final.

### C. Instrucţiuni de salt

Intrucţiunea:

1. *break* este utilizată pentru întreruperea execuţiei instrucţiunilor de ciclare şi a celor switch;
2. *continue* poate fi folosită doar în interiorul instrucţiunilor de ciclare forţând trecerea la un nou ciclu;
3. *return* este utilizată pentru ieşirea forţată dintr-o metodă.

### D. Lucru cu şiruri de caractere

Cele mai cunoscute clase care lucrează cu şiruri de caractere sunt: *String, StringBuffer*.

#### 1. *Clasa String*

Cea mai importantă caracteristică a clasei *String* este că obiectele o dată iniţializate nu se mai pot modifica, fiecare dintre aceste obiecte indicând spre o zonă diferită de memorie.

O altă proprietate a obiectelor de tip *String* este că pot fi utilizate impreună cu operatorul '+', pentru concatenarea şirurilor. Prin concatenare se instanţiază un nou obiect de tip *String* care va referenţia un şir alcătuit din şirurile alipite cu ajutorul operatorului '+'.

Această clasă oferă o serie de metode pentru lucru cu şiruri de caractere. Aceste metode au în vedere compararea şirurilor, căutarea în şiruri, şamd. Se recomandă studierea API-ului.

### 2. *Clasa StringBuffer*

Un obiect *StringBuffer* reprezintă ca şi în cazul clasei *String* un şir de caractere. Diferenţa între cele două este că primul obiect poate suferi modificări. Acest lucru este posibil datorită metodelor *insert()* şi *append()* care permit inserarea, respectiv adăugarea unor şiruri de caractere. Pentru mai multe informaţii se recomandă studierea API-ului.

## II. PREZENTAREA LUCRĂRII DE LABORATOR

### A. Instrucţiuni Java

Codul sursă de mai jos exemplifică modul de utilizare al instrucţiunilor din secţiunile I A, I B, I C.

```
1  import java.util.Scanner;
2
3  public class TestIntructiuni {
4     public static void main(String args[]) {
5        Scanner s = new Scanner(System.in); //cu ajutorul acestei instructiuni se citeste text de la
             tastatura
6        System.out.println("Introduceti un numar: ");
7        int x = s.nextInt(); //in x se va retine numarul tastat
8        System.out.println("Rezultatul functiei este: " + test(x)); /*se apeleaza metoda test cu parametru x
             */
9     }
10
11    public static int test(int x) {
12       int suma = 0; //se initializeaza variabila suma cu 0
13       for (int i = 0; i < 4; i++) {
14          System.out.println("Am intrat in nivel 1");
15          int j = 0;
16          while (j++ < x) {//intai se efectueaza evaluarea expresiei j < x si abia apoi se incrementeaza
                variabila j
17             System.out.println("Am intrat in nivel 2");
18             System.out.println("i= " + i + "; j= " + j);
19             switch (i) {
20             //in cazul in care i=0, 1 sau 2 se sare la urmatorul pas si se ignora restul instructiunilor
                   de dupa continue din ciclul curent (while)
21             case 0:
22                continue;
23             case 1:
24                continue;
25             case 2:
26                continue;
27             case 3: //in cazul in care i=3 se actualizeaza suma si apoi se iese fortat din switch
28                suma += i+j;
29                break;
30             }
31          }
32          System.out.println("Am iesit din nivel 2");
```

```
33        }
34        System.out.println("Am iesit din nivel 1");
35        return suma; //se iese fortat din metoda test
36     }
37
38  }
```

### B. Lucru cu şiruri de caractere

Exemplul următor evidenţează caracterul imuabil pe care-l au obiectele de tip *String*. Se citeşte de la tastatură un şir de caractere şi se verifică dacă acesta coincide cu un alt şir de caractere.

```
1  import java.util.*;
2  public class CheckPassword {
3     public static void main(String args[]){
4        Scanner s = new Scanner(System.in);
5        String password = "java";
6        String userInput;
7        System.out.println("Care e parola?");
8        userInput= s.next();
9
10       System.out.println("Ai tastat: ");
11       System.out.println(userInput);
12       System.out.println("Dar parola este: ");
13       System.out.println(password);
14
15       //if (password.equals(userInput)){
16       if (password == userInput){
17          System.out.println("Ai trecut mai departe!");
18       }else{
19          System.out.println("NU ai trecut mai departe!");
20       }
21    }
22 }
```

Deşi variabila *userInput* ar conţine aceleaşi caractere ca variabila *password*, folosind operatorul '==' se va afişa textul de pe ramura *else* a instrucţiunii *if*. Acest lucru se întâmplă din cauza faptului că se compară adresele de memorie ale variabilelor şi nu conţinutul de la acele zone de memorie.

Decomentaţi linia 15 şi comentaţi linia 16. Observaţi ce se intâmplă dacă variabilele conţin acelaşi şir de caractere.

Următorul exemplu citeşte de la tastatură un şir de caractere şi înlocuieşte fiecare vocală întâlnită cu următorul caracter din alfabet. Acesta foloseşte clasa *StringBuffer*.

```java
import java.util.Scanner;
public class ReplaceVowel {
   public static void main(String args[]) {
      Scanner s = new Scanner(System.in);
      System.out.println("Introduceti cuvantul: ");
      String word = s.nextLine();
      System.out.println("Cuvantul rezultat este: "+replaceVowel(word.toLowerCase()));
   }

   public static boolean checkVowel(char c){
      c = Character.toLowerCase(c);
      return ("aeiou".indexOf(c) >= 0); /** se returneaza true daca metoda indexOf aplicata sirului de
                  caractere "aeiou" impreuna cu parametru c returneaza o valoare pozitiva*/
   }

   public static String replaceVowel(String word){
      StringBuffer sb = new StringBuffer(word);/**variabila sb este intializata cu valoarea
                        variabilei word*/
      for (int i=0; i<sb.length(); i++){/** sb se parcurge caracter cu caracter*/
         if (checkVowel(sb.charAt(i))){/** fiecare caracter al lui sb este verificat daca este vocala */
            sb.setCharAt(i, (char)(sb.charAt(i)+1)); /** se pune pe pozitia i in sb urmatorul caracter
                           din alfabet*/
         }
      }
      return sb.toString();
   }
}
```

## III. TEMĂ

1. Rulaţi programele din secţiunea II.

2. Citiţi un şir de caractere de la tastatură. Folosind intrucţiunea *switch*, realizaţi un meniu pentru următoarele cerinţe:

   (a) să se afişeze lungimea şirului de caractere;

   (b) să se returneze ultima poziţie pe care se întâlneşte caracterul 'a';

   (c) să se numere de câte ori apare în şirul de caractere secvenţa 'abc';

   (d) să se verifce dacă şirul de caractere este palindrom.

   (e) să se şteargă toate caracterele de pe poziţiile pare.

   Implementaţi cerinţele de la punctele 2a pana la 2e.