# TEHNICI AVANSATE DE PROGRAMARE

LUCRARE DE LABORATOR 2

# Instrucțiuni ale limbajului Java

## I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu câteva instrucțiuni de bază ale limbajului Java: (if( )/else, switch( )), construcțiile iterative while( ), do/while( ), for( ), instrucțiunile de salt break, continue; cu importanța şi situațiile de utilizare a acestora.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie programe Java în care să folosească instrucțiunile limbajului menționate anterior.

## II. NOȚIUNI TEORETICE

Observație: Vezi capitolul 1 din curs.

### 1. Instrucțiunea de selecție if( )/else

Instrucțiunea **if( )/else** primește un argument de tip boolean, ca bază a alegerii. Adesea se utilizează o expresie de comparare pentru a furniza argumentul. Exemplu:

```
1. if (x>=10) {
2.       System.out.println("x este mai mare decat 10");
3. }
```

Linia 2 a codului se execută dacă testul (x>=10) din linia 1 returnează adevărat.

Putem furniza cod care să se execute atunci când testul returnează fals în partea **else** a instrucțiunii. Exemplu:

```
1. if (x>=10) {
2.       System.out.println("x este mai mare sau egal decat 10");
3. }
4. else {
5.       System.out.println("x este mai mic decat 10");
6. }
```

Construcția **if( )/else** efectuează un test numai între două posibile căi de execuție, dar se pot utiliza intrucțiuni **if( )/else** imbricate pentru a selecta între mai multe posibilități. Exemplu:

```
class Selectie {
   public static void main(String args[]) {
      int x=7,y=20,z=2;
      if ((x>10) && (y>6)){
               System.out.println(x+y);
      }


      else if(x<12){
         if(y>50){
```

```
            x++;
         }
         else {
            y--;
         }
         z++;y--;
         System.out.println("z="+z+" y="+y);

      }
      else {
         System.out.println(x-y);
      }
   }
}
```

## 2. Instrucțiunea de selecție switch( )

Dacă este nevoie să se aleagă între mai multe căi de execuție alternative, şi dacă alegerea se poate baza pe o valoare **int**, atunci se poate utiliza instrucțiunea **switch( )**. Exemplu:

```
switch(x) {
     case 1:
                System.out.println("1");
     case 2:    System.out.println("2");
     case 3:
                System.out.println("3");
                break;
     default:
                System.out.println(" altceva in afara de 1,2 sau 3");
                break;
}
```

**Observații:**

• Variabila **x** poate să fie numai de tipul **byte, short, char** sau **int** (sau pe scurt, valoarea lui **x** trebuie să fie compatibilă la asignare cu tipul **int**).

• Compararea valorilor care urmează după etichetele **case** cu valoarea expresiei furnizate de argumentul lui **switch( )** determină calea de execuție. Argumentele etichetelor **case** trebuie să fie constante, sau cel puțin expresii constante care să poată fi complet evaluate la momentul compilării (nu se pot utiliza variabile sau expresii ce utilizează variabile).

• Fiecare etichetă **case** primeşte un singur argument, dar când execuția sare la una dintre aceste etichete, continuă în jos până când atinge o instrucțiune **break**.

• Eticheta **default** este comparabilă ca efect cu partea **else** a unei instrucțiuni **if( )/else**. Execuția sare la eticheta **default** dacă nici unul dintre argumentele etichetelor **case** nu se potrivește cu argumentul furnizat de **switch( )**. Deşi se obişnuieşte ca eticheta **default** să fie plasată la sfârşitul blocului **switch( )**, nu există nici o regulă care să impună acest lucru.

Exemplu:

```
class TestSwitch {
      public static void main(String args[ ]) {
             int i=7;
             switch(i) {
                case 1:
                        System.out.println(i+1);
                        break;
                case 2+5:
```

```
                        i=i+3;
                        System.out.println(i);
                case 8:
                        System.out.println(i+4);
                        break;
                default:
                        System.out.println(i+10);
                        break;
                case 10:
                        System.out.println(i+2);
            }
        }
}
```

## 3. Instrucțiunea de ciclare *while( )*

Vezi Curs 3, secțiunea 2.10.1.

Este o instrucțiune de ciclare cu test inițial şi are număr necunoscut de paşi.

Observație:

Se recomandă utilizarea unui bloc pentru a conține codul corpului buclei iterative (chiar dacă acesta constă dintr-o singură instrucțiune; în general este puțin probabil să rămână aşa, iar lipsa acoladelor este o eroare greu de depistat într-un program mare). Observația este valabilă şi pentru celelalte intrucțiuni.

Exemple:

```
class TestW1 {
      public static void main(String args[ ]) {
            char c='a';
            while(++c <= 'm')
                  System.out.print(c+"   ");
            System.out.println("\n");
            while(c-- > 'd')
            {
                  System.out.print(c+"   ");
            }
       }
}
```

```
class TestW2 {
      public static void main(String args[ ]) {
            double d1=34.8,d2=7;
            while(d1 > 0)
            {
                  d1-=d2;
                  System.out.println(d1);
            }
            System.out.println("\n");
            while(d2<5.3)
            {
                  System.out.println(d2--);
            }
```

```
    }
}
```

## 4. Instrucțiunea de ciclare *do/while( )*

Vezi Curs 3, secțiunea 2.10.2.

Este o instrucțiune de ciclare cu test final (corpul se execută cel puțin o dată) şi are număr necunoscut de paşi.

Exemplu:

```
class TestDo {
      public static void main(String args[ ]) {
             char c='a';
             int j=9;
             do {
                  System.out.println(c++);
             } while(++c <= 'i');
             j=c;
             System.out.println("\n j="+j);
             do {
                  System.out.println(j--);
             }while(j<95);
      }
}
```

## 5. Instrucțiunea de ciclare *for( )*

Vezi Curs 3, secțiunea 2.10.3.

Permite efectuarea de iterații peste un interval de valori. Este o instrucțiune de ciclare cu test inițial şi are număr cunoscut de paşi.

Exemple:

```
class TestFor1 {
      public static void main(String args[ ]) {
             int i;
             for(i=1;i<=5;i+=2)
                System.out.println("i="+i);
      }
}
class TestFor2 {
      public static void main(String args[ ]) {
             float f;
             char c;
             for(f=1.1f;f<=10.5;f++){
                System.out.println("f="+f);
             }
             for(c='m';c>'a';c-=2){
                System.out.println("c="+c);
             }
      }
}
```

Variabila cu rol de contor se poate declara chiar în cadrul intrucțiunii **for( )**. O astfel de variabilă va avea domeniul de valabilitate restricționat la blocul instrucțiunii **for( )**. Acest fapt protejează împotriva interferenței variabilelor contor şi a reutilizării accidentale a acestora.

```
for(int i=0;i<10;i++){
       System.out.println("i="+i);
}
```

Codul echivalent implementat folosind intrucțiune **while( )** arată astfel:

```
{
   int i=0;
   while(i<10) {
       System.out.println("i="+i);
       i++;
   }
}
```

Astfel, se poate mai uşor observa că: scopul variabilei **i** din cadrul instrucțiunii **for( )** este restricționat la blocul corespunzător lui **for( )**, incrementarea contorului se face după executarea corpului principal al intrucțiunii **for( )**, iar apoi se efectuează din nou testul.

```
class TestFor3 {
      public static void main(String args[ ]) {

             for(int i=0;i<10;i++){
                System.out.println("i="+i);
             }
           +i);   eroare la compilare
//  System.out.println("i="             for(int i=0;i<10;i++){
                System.out.println("i="+i);
             }
      }
}
```

Intrucțiunea **for( )** permite utilizarea separatorului virgulă într-un mod special. Exemple:

```
int j,k;
for(j=0,k=1; j+k<20; j++,k+=3) {
   System.out.println("j="+j+"  k="+k);
}

for(int i=7,j=0; i<15; j++) {
   System.out.println("i="+(i++)+"  j="+j);
}
```

Dar nu se poate utiliza separatorul virgulă în orice mod. Astfel următoarele utilizări sunt ilegale:

```
int i=7;
for(i++,int j=0; i<10; j++) { } // ilegal


for(int i=7, long j=0; i<10; j++) { } // ilegal
```

## 6. Instrucțiunile de salt *break, continue*

Instrucțiunile **break** şi **continue** se utlizează atunci când este nevoie să se abandoneze execuția corpului unei instrucțiuni de ciclare, sau poate a unui număr de bucle iterative imbricate.

Exemple:

```
class TestCont1
{
    public static void main(String args[ ])
    {
        char array[][]=new char[3][5];
        for(int i=0;i<array.length;i+=2)
        {
            for(int j=0;j<array[i].length;j++)
            {
               array[i][j]='a';
            }
        }
        mainLoop: for(int i=0; i<array.length; i++)
        {
            for(int j=0; j<array[i].length; j++)
            {
               if(array[i][j]== '\u0000')
                    continue mainLoop;
               System.out.println(array[i][j]);
            }
            System.out.println("\n");
        }
    }
}
```

Adevărata putere a instrucțiunii **continue** este aceea că permite ieşirea din corpul unor bucle iterative imbricate.

Să observăm şi utilizarea etichetei (label) **mainLoop** care a fost aplicată instrucțiunii **for( )** de la linia 13. De obicei etichetele se aplică la începutul unor instrucțiuni de ciclare **while( ), do/while( ), for( )**.

Când procesarea vectorului bidimensional de caractere ajunge la o valoare zero, aceasta este abandonată şi se sare la efectuarea incrementării **i++** din cadrul instrucțiunii **for( )** corespunzătoare etichetei **mainLoop** de la linia 13.

```
class TestCont2
{
      public static void main(String args[ ])
      {
         int vec[]=new int[10];
         for(int i=0;i<vec.length;i+=2)
             vec[i]=i*i+20;
         for(int i=0;i<vec.length;i++)
         {
             if (vec[i]==0)
                     continue;
             System.out.println(vec[i]);
         }
      }
}


class TestB1
```

```
{
      public static void main(String args[ ])
      {
         int vec[]=new int[10];
         for(int i=0;i<vec.length;i+=2)
             vec[i]=i*i+20;
         for(int i=0;i<vec.length;i++)
         {
             if (vec[i]==0)
                     break;
             System.out.println(vec[i]);
         }
      }
}
```

Utilizarea instrucțiunii **break** cauzează abandonarea întregului ciclu. În acest caz, în loc de a se sări peste procesarea lui **vec[i]** şi de a se continua cu procesarea lui **vec[i+1]** aşa cum se întâmplă în cazul utilizării instrucțiunii **continue**, se abandonează întregul ciclu **for( )** de îndată ce este întâlnit un element cu valoarea 0.

Se pot utiliza etichete şi în cazul instrucțiunilor **break**.

**Observație:**

Etichetele pot fi aplicate oricărei instrucțiuni, iar **break** poate fi utilizat pentru a se ieşi din orice bloc etichetat, chiar dacă blocul este corpul unei instrucțiuni de ciclare sau nu.

```
class TestB2 {
      public static void main(String args[ ]) {
         int vec[]={1,4,6,8,12,56,77,2,5};
         for(int i=0;i<vec.length;i++) {
             et: {
                   System.out.println(++vec[i]);
                   if(i%2==0) break et;
                   System.out.println(--vec[i]);
                 }
             System.out.println(vec[i]);
         }
      }
}
```

## 7. Alte exemple utile

```
class Ex1 {
    static char f(int n) {
        System.out.print(n);
        if(n>=0) return '+';
        return  '-';
    }
    public static void main(String args[ ]) {
        System.out.print("  "+f(1)+"\n");
        System.out.print("  "+f(-1));
   }
}
```

# III. MODUL DE LUCRU

- Clasic:

1. Se editează codul sursă al programului Java folosind un editor de text disponibil (de ex., se poate utiliza Notepad).
2. Se salvează fişierul cu extensia **.java**.
3. Compilarea mini-aplicaţiei Java se va face din linia de comandă:
   **javac nume_fişier_sursă.java**
   În cazul în care programul conţine erori acestea vor fi semnalate şi afişate.
4. Pentru rularea aplicaţiei Java, se lansează interpretorul Java:
   **java nume_clasă_care_conţine_main**

- Se foloseşte utilitarul disponibil în laborator J2SDK Net Beans.

# IV. TEMĂ

- Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic (acolo unde este cazul).
- Să se răspundă la următoarele întrebări grilă, explicând şi alegerea rezultatului.

1. Ce se va afişa după executarea fragmentului de cod de mai jos?

```
int x=0, y=4, z=5;
if (x>2) {
    if (y<5){
       System.out.println("mesaj unu");
    }
    else {
       System.out.println("mesaj doi");
    }
}
else if (z>5) {
    System.out.println("mesaj trei");
}
else {
    System.out.println("mesaj patru");
}
```

A. mesaj unu
B. mesaj doi
C. mesaj trei
D. mesaj patru

2. Care afirmaţie despre următorul fragment de cod este adevărată?

```
1.  int j=2;
2.  switch(j) {
3.      case 2:
```

```
4.              System.out.println("valoarea este doi");
5.      case 1+2:
6.              System.out.println("valoarea este trei");
7.              break;
8.      default:
9.              System.out.println("valoarea este "+j);
10.             break;
11. }
```

**A. Codul este ilegal datorită expresiei de la linia 5**
**B. Tipurile acceptate pentru variabila j, care este argument al construcției switch(), pot fi byte, short, int sau long.**
**C. Output-ul va fi:** *valoarea este doi*
**D. Output-ul va fi:** *valoarea este doi*, **urmat de** *valoarea este trei*
**E. Output-ul va fi:** *valoarea este doi*, **urmat de** *valoarea este trei*, **urmat de** *valoarea este 2*

3. Să considerăm următorul fragment de cod:

```
for(int i=0;i<2;i++) {
   for(int j=0;j<3;j++) {
      if(i==j) {
         continue;
      }
      System.out.println("i="+i+" j="+j);
   }
}
```

Care linii vor face parte din output?

**A. i=0 j=0**
**B. i=0 j=1**
**C. i=0 j=2**
**D. i=1 j=0**
**E. i=1 j=1**
**F. i=1 j=2**

4. Să considerăm următorul fragment de cod:

```
outer: for(int i=0;i<2;i++) {
    for(int j=0;j<3;j++) {
          if(i==j) {
                continue outer;
          }
          System.out.println("i="+i+" j="+j);
    }
}
```

Care linii vor face parte din output?

**A. i=0 j=0**
**B. i=0 j=1**
**C. i=0 j=2**
**D. i=1 j=0**
**E. i=1 j=1**
**F. i=1 j=2**

5. Care dintre următoarele construcții de ciclare sunt legale? (Alegeți una sau mai multe.)

```
A.
while(int i<7) {
  i++;
  System.out.println("i="+i);
}

B.
int i=3;
while(i) {
  System.out.println("i="+i);
}

C.
int j=0;
for(int k=0;j+k !=10; j++,k++) {
    System.out.println("j="+j+" k="+k);
}

D.
int j=0;
do {
    System.out.println("j="+j);
    if(j==3)  { continue loop; }
}while(j<10);
```

- Scrieți un program Java pentru a rezolva ecuația de gradul doi.
- Scrieți un program Java care să determine numărul de parametri furnizați în linia de comandă la execuția programului. Dacă acest număr este 0, atunci programul afișează "0 argumente"; dacă în linia de comandă se furnizează un singur parametru atunci programul afișează "1 argument" precum și valoarea acestui argument; altfel se va afișa textul "mai multe argumente" și valorile acestora.
- Scrieți un program Java ce constă dintr-o clasă care la rândul ei conține două funcții. Prima este o funcție ***static String f(String s)*** care afișează valoarea șirului de caractere **s** la care se concatenează șirul de caractere "< - - >" și apoi returnează **s.** A doua este metoda **main** în cadrul cărei se apelează funcția **f** de mai sus în cadrul unor comenzi **System.out.println**; funcția **f** se va apela o dată pentru un șir de caractere oarecare (ales de programator) și apoi pentru primul argument din linia de comandă dacă acesta există.
- Scrieți un program Java ce constă din următoarele: Se construiește un vector unidimensional cu elemente de tip float și se inițializează elementele cu valori arbitrare alese de programator (atât pozitive cât și negative). Într-o buclă iterativă se vor afișa valorile acestor elemente cu următoarea specificare: dacă se întâlnește un element negativ nu se va afișa ci se va sări la următorul element din vector.