TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 4

Instrucțiuni de bază 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: 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

1. 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--);
             }
      }
}
```

2. 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);
      }
}
```

3. 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);
             }
           //  System.out.println("i="+i);   eroare la compilare
             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
```

4. 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]);
         }
      }
}
```

5. 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

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**. Fişierul trebuie salvat la următoarea locaţie: **c:\JBulider7\jdk1.3.1\bin**
3. Compilarea mini-aplicaţiei Java se va face din linia de comandă. Se poate proceda astfel. Se deschide o fereastră MS-Dos: Start ->Run, se tipăreşte **command** în căsuţa de text şi se apasă butonul OK. Printr-o schimbare de directoare şi subdirectoare se trece la locaţia: **c:\JBulider7\jdk1.3.1\bin.** Sau, se lansează WindowsCommander. Se trece la locaţia **c:\JBulider7\jdk1.3.1\bin.** Se deschide o fereastră MS-Dos: Commander ->Run Dos.
4. Pentru compilare, se tipăreşte la prompter **javac nume_fişier_sursă.java** şi se apasă Enter. De ex., dacă fişierul se numeşte Test.java, se va scrie **javac Test.java**. În cazul în care programul conţine erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conţine erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Pentru rularea aplicaţiei Java, se lansează interpretorul Java. Se tipăreşte la prompter următoarea comandă **java nume_clasă_care_conţine_main** şi se apasă Enter. De ex., dacă clasa din program care conţine metoda *main( )* se numeşte Test, se va scrie **java Test**.

IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic (acolo unde este cazul).

2. Să se răspundă la următoarele întrebări grilă, explicând şi alegerea rezultatului.

1. 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
```

2. 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
```

3. 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);
```

3. Se vor testa practic răspunsurile la întrebările exercițiului anterior.

4. Scrieți un program Java care să efectueze umătoarele. Se determină numărul de parametri furnizați în linia de comandă la execuția programului. Dacă acest număr este 0, atunci programul afişează “0 argumente”; dacă în linia de comandă se furnizează un singur parametru atunci programul afişează “1 argument” precum şi valoarea acestui argument; altfel se va afişa textul “mai multe argumente” şi valorile acestora.

5. 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 afişează valoarea şirului de caractere **s** la care se concatenează şirul de caractere “< - - >” şi 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 şir de caractere oarecare (ales de programator) şi apoi pentru primul argument din linia de comandă dacă acesta există.

6. Scrieți un program Java ce constă din următoarele. Se construieşte un vector unidimensional cu elemente de tip float şi se inițializează elementele cu valori arbitrare alese de programator (atât pozitive cât şi negative). Într-o buclă iterativă se vor afişa valorile acestor elemente cu următoarea specificare: dacă se întâlneşte un element negativ nu se va afişa ci se va sări la următorul element din vector.

7. Aceeaşi problemă ca mai sus, dar dacă se întâlneşte un element negativ se doreşte ieşirea din bucla iterativă.