C5

Instrucțiunile (1) limbajului Java

- Categorii: secventa, decizie, ciclu, salt
- Secventa cu instructiuni
  - simple
  - compuse

Decizia cu if

- prezentare, functionare, probleme

Decizia cu switch

- prezentare, functionare, probleme

Obiective

Dupa parcurgerea acestui curs ar trebuie sa puteti:

- face diferenta intre instructiuni si date;
- defini categoriile de instructiuni din Java
- inteleg si scrie cod din cateogria de:
  - Secventa cu instructiuni: simple, compuse
  - Ramificatie (decizia) simpla sau multipla cu if
  - Ramificarea multipla cu switch
- identifica si corecta erorile comune

Rularea programelor

Programele sunt alcătuite din instrucţiuni.

Implicit, instrucţiunile se rulează secvenţial.

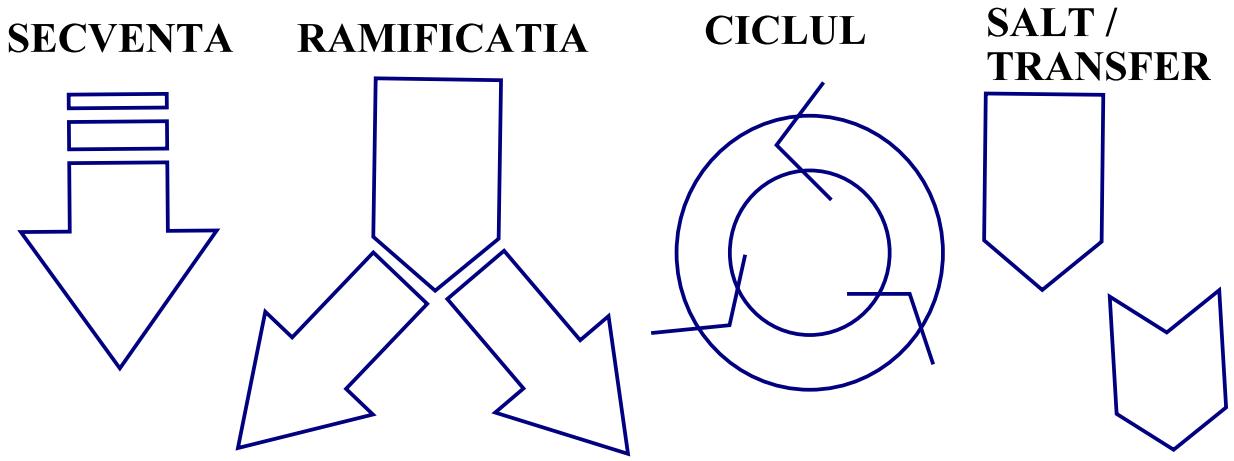
Există situaţii în care se fac devieri de la rularea secvenţială:

- Decizia sau ramificarea: if, switch
- Ciclul: do, for, while
- Saltul sau transferul: break, continue, apel de metodă

O aplicaţie Java este formată din instrucţiuni. Procesul prin care JVM îndeplineşte o instrucţiune se numeşte rulare sau execuţie. În Java terminarea rulării unei instrucţiuni se poate face cu succes, cu excepţie sau cu eroare.

O instrucţiune este formată din una sau mai multe expresii care se rulează ca o singură acţiune. De exemplu `x = 5` este expresie, în timp ce `x = 5;` este deja o instrucţiune. O expresie terminată în caraterul ; se numeşte instrucţiune, iar caracterul ; se numeşte terminator de instrucţiune. Tot instrucţiuni sunt: `x = 5.3*(4.1/Math.cos(0.2*y));` respectiv `System.out.println(x);`. Instrucţiuni sunt inclusiv declaraţiile simple sau multiple, cu sau fără iniţializare. Instrucţiunile vor fi plasate în clase pentru alcătuirea aplicaţiei funcţionale.

Implicit, aplicaţia Java începe din metoda `main`, iar instrucţiunile se rulează secvenţial, adică de sus în jos, în ordinea scrierii lor în aplicaţie pâna la terminarrea rularii tutror instrucţiunilor. În situaţia unei aplicaţii complexe rularea poate devia de la prelucrarea secvenţială astfel, există posibilitatea ramificării la nivelul rulării unor porţiuni de cod pe baza unor condiţii (`if`, `switch`), repetarea unui grup de instrucţiuni atânt timp cât o expresie de control este adevărată (`while`, `do while`, `for`), saltul de la o instrucţine la o alta în program (`break`, `continue`).



SECVENŢA

Secvenţa este alcătuită din:

- instruncţiuni simple
- blocuri numite şi instrucţiuni compuse

Instrucţiune simplă: `expresie;`

Blocul grup de instruncţiuni cuprinse între acolade `{ }`

Instruncţiunea simplă
Instrucţiune simplă este orice expresie terminată în caracterul ;, iată câteva exemple:

```
int x = 1, y = 2 , z;
z = x+ y;
x=y++;
z++;
```

Instrucţiunea vidă
Instrucţiunea vidă constă într-un `;`. Ea nu are efect la nivelul execuţiei instrucţiunilor aplicaţiei. De exemplu, dacă considerăm instrucţiunea simplă **`expresie;`** iar partea de **`expresie`** lipseşte, se obţine o instrucţiune vidă. În practică, se foloseşte pentru a da posibilitatea ca aplicaţia să poată fie completată, de exemplu, cu instrucţiuni în locuri nu eram siguri că va fi nevoie de acestea. Un alt avantaj este cel de evitare al unor erori datorate scrierii "mecanice" a lui `;`, fie secvenţa de cod:

```
if (x < 0) { x = -x; };
```

Caracterul ; de după } este legal, însă compilatorul îl consideră o instrucţiune vidă. Un alt motiv al instrucţiunii vide este situaţia în care dorim să reprezentăm, în aplicaţie, starea de "fă nimic".

Instruncţiunea compusă sau blocul
Mai multe instrucţiuni pot fi grupate cu ajutorul acoladelor într-o instrucţiune compusă, numită şi bloc.

```
{
    int x = 1;
    System.out.println(”x = ” + x);
    ++x;
}
```

Instrucţiunea compusă este sintactic echivalentă cu instrucţiunea simplă. Nu se scrie ; după acolada de închidere.

Orice variabilă declarată într-un bloc are presistenţa limitată la interiorul blocului în care s-a declarat. În afara blocului variabilă respectivă încetează să mai existe.

DECIZIA - if

- `if` se foloseşte pentru programarea unei decizii şi are forma generală:

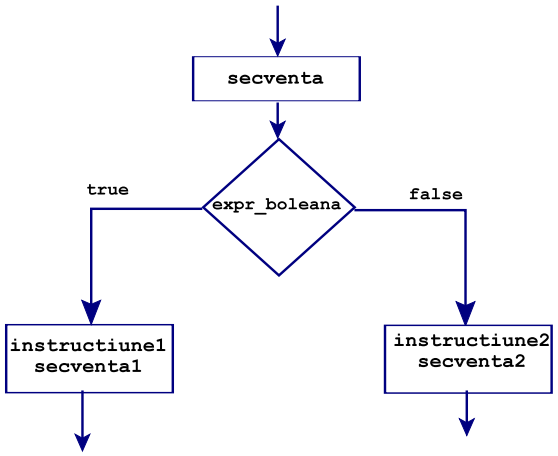
```
if (expr_boleana)
    instructiune1
[else
    instructiune2]
```

- `expr_boleana` trebuie să fie scrisă între paranteze rotunde şi să aibă un rezultat boolean
- dacă este nevoie de mai multe instrucţiuni trebuie folosită instrucţiunea compusă în locul celei simple

Instrucţiunea if

Permite ramificarea rulării în aplicaţie pe baza valorii luate de `expr_boleana`. Expresia boleană determină care dintre ramuri este rulată după cu urmează:

- dacă valoarea expresiei boleene este `true` se rulează instrucţiune simplă `instructiune1`
- dacă valoarea expresiei boleene este `false` se rulează instrucţiunea simplă `instructiune2`



Porţiune cu `else` este opţională, dacă este omisă atunci nu se va rula nimic în situaţia în care `expr_boleana` ia valoarea `false`

Dacă se doreşte ca instrucţunea `if` să aibă efect asupra unui grup de instrucţiuni acestea trebuie puse înt-un bloc. Blocul ne permite ca în locul unei instrucţiuni simple Java să avem voie să scriem un grup de instrucţiuni Java.

Exemplu:

```
if (i>0)
   System.out.println("i -= " + i);
```

```
if (i>0)
{
   System.out.println("i -= " + i);
   i=i%2;
```

}

if-uri imbricate

- apar atunci când una dintre instrucţiunile simple din if este la rândul ei un if

```
if (expr_boleana1)
    if (expr_bolenan2)
        instructiune1
    [else
        instructiune2]
[else
    instructiune3]
```

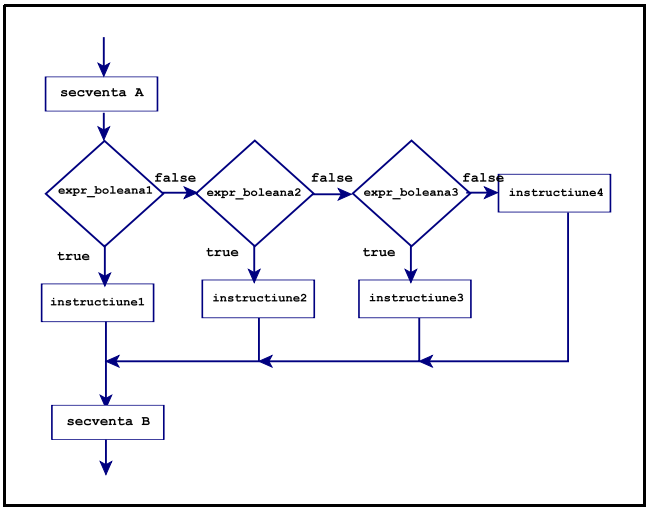
Dacă este cazul instrucţiunile `if` pot fi imbricate (cuprinse una în alta). Deoarece partea de `else` este opţională se poate ajunge la situaţia în care pune problema aparteneţei lui `else`.

```
if (expr_boleana1)
if (expr_bolenan2)
    instructiune1
else
    instructiune3
```

Regula este aceea că `else` se asociază cu cel mai apropiat `if`. Dacă acestă asociere implicită nu este convenabilă atunci se pot folosi acoladele pentru a o modifica.

Când trebuie să luam o singură decizie dintre mai multe avem de a face cu o multidecizie. Aceasta se poate programa cu o secvenţă de if-uri după cum urmează. Dacă `expr_boleana`**`i`** este `true` atunci se rulează `instructiune`**`i`** (i = 1, 2, 3), altfel se ajunge la `instructiune`**`4`**.

```
secventa A
if (expr_boleana1)
    instructiune1
else if (expr_boleana2)
    instructiune2
```



```
else if (expr_boleana3)
    instructiune3
else
    instructiune4
secventa B
```

Operatorul condiţional ? :

- este o alternativă la `if else`
- este ternar (are 3 operanzi) având forma:

  `expr_boleana ? expr1 : expr2`

- dacă `expr_boleana` este `true` ia valoarea lui `expr1`, altfel pe a lui `expr2`

Cei trei operanzi formează o expresie condiţională. Primul operand (`expr_boleana`) trebuie să fie o expresie boleană (de exemplu o condiţie), al doilea operand (`expr1`) este valoarea pe care expresia condiţională o întoarce dacă expresia boleană ia valoarea `true`. Al treilea operand (`expr2`) este valoarea expresiei condiţionale dacă expresia boleană ia valoarea `false`.

Fie linia de cod:

```
System.out.println(nota >=5 ? ”Admis.” : ”Respins.”);
```

Dacă valoarea din variabila `nota` este mai mare sau egală cu 5, atunci expresia condiţională ia valoarea `”Admis.”` , atfle ia valoarea `”Respins.”`.

O aplicaţie este determinarea minimului dintre două valori:

```
int a = 5, b =3;
int minim;

minim = (a > b) ? b : a;
System.out.println(”Minimul este: ” + minim);
```

Erori specifice lui if

❶
```
int a = 3, b = 2;
if (a > 0)
    if (b < a)
        System.out.println(”b < a”);
else
    System.out.println(”a < 0”);
```

❷
```
int a = 3;
if (a = 5)
    System.out.println(”a este 5”);
```

❸
```
int a = 3;
if (a%2 == 1) ;
    System.out.println(”a este impar”);
```

Eroarea ❶
Asocierea dintre if-uri şi else este greşită, secvenţa de cod fiind echivalentă cu:

```
if (a > 0) {
  if (b < a)
    System.out.println(”b < a”);
  else
    System.out.println(”a < 0”);
}
```

adică else-ul ţine de al doilea if. Pentru ca acesta să ţină de primul trebuie să folosim acoladele astfel:

```
if (a > 0) {
  if (b < a)
    System.out.println(”b < a”);
  }
else
  System.out.println(”a < 0”);
```

Eroarea ❷
Aici în locul opreratorului de testare a egalităţii (= =) s-a folosit cel de atribuire (=). Java va da eroare la compilare deoarece expresia de testat trebuie să fie boleană.

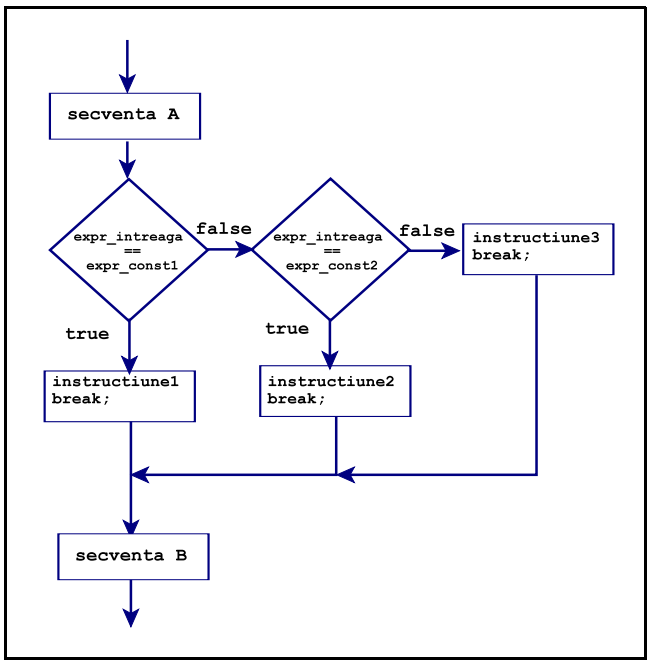
Eroarea ❸
S-a pus caracterul `;` după testul din if. Compilatorul nu dă eroare deoarece tratează if-ul având drept corp o instrucţiune vidă.

DECIZIA - switch

- `switch` se foloseşte pentru programarea unei multidecizii, are forma generală:

```
switch (expr_intreaga) {
    case expr_const1:
        instructiune1
        break;
    case expr_const2:
        instructiune2
        break;
    [default:]
        instructiune3
        break;]
}
```

Instrucţiunea de decizie `switch` se utilizează atunci când avem de rulat o secvenţă dintre mai multe posibile. Valoarea expresiei de test (`expr_intreaga`) trebuie să fie de tipurile: `byte`, `char`, `short` sau `int`. Tipurile `long`, `float`, `double` sau alte tipuri de obiecte (`String` etc.) sunt interzise. Valoarea expresiei de test se compară, secvenţial, cu expresiile constante (`expr_const`**`i`**) care etichetează instrucţiunile `instructiune`**`i`**. Dacă s-a găsit egalitate rularea începe de la instrucţiunea următoare (cea care vine imediat după cele `:` ale lui `case`) şi ţine până la întâlnirea unui `break`, moment în care rularea se conţinuă cu `secventa B`. Dacă nu se găseşte nici o egalitate rularea va continua cu instrucţiunile următoare etichetei `default`. Eticheta `default` este opţională, dacă aceasta lipseşte switch-ul nu va face nimic în lipsa găsirii unei egalităţi.



Etichetele trebuie să fie expresii constante (au valori cunoscute în momentul compilării). În acest scop se pot folosi literali sau variabile `final`.

```
import java.util.*;
public class Switch {
    public static void main(String[] args) {
     Scanner intrare = new Scanner(System.in);
     System.out.print("Selecteaza optiunea (1,2,...,7): ");
     int optiune = intrare.nextInt();

     switch (optiune) {
         case 1:
         case 2:
         case 3:
         case 4:
         case 5:
             System.out.println("La munca cu tine");
             break;
         case 6:
             System.out.println("La chef cu tine");
             break;
         case 7:
             System.out.println("Relaxeaza-te sau revino-ti!");
             break;
         default:
            System.out.println("Aici, pe Terra, avem numai 7 zile!");
             break;
     } //terminare switch

    } //terminare main

} // terminare clasa Switch
```

Rezultate:

```
Selecteaza optiunea (1,2,...,7): 5
La munca cu tine
```

C5- Întrebari

1. Explicaţi conceptele de secventa si ramificatie
2. Cate categorii de instructiuni simple sunt?
3. Ce este instructiunea vida?
4. Cum se realizeaza decizia multipla cu if?
5. Care sunt erorile comune la utilizarea lui if?
6. Ce limitari are switch-ul in comparatie cu if?
7. Care sunt erorile comune la utilizarea lui switch?

BIBLIOGRAFIE

1. http://www.oracle.com/technetwork/java/javase/documentation/index.html

2. http://docs.oracle.com/javase/6/docs/

3. Stefan Tanasa, Cristian Olaru, Stefan Andrei, Java de la 0 la expert, Polirom, 2003, ISBN: 973-681-201-4.

4. Herber Schild, Java 2 - The Complete Reference, Fourth Edition, Osborne, 2001, ISBN: 0-07-213084-9.

5. Deitel H.M., Deitel P. J., Java - How to programm, Fith Edition, Prentice Hall, 2003, ISBN: 0-13-120236-7.

6. http://www.east.utcluj.ro/mb/mep/antal/downloads.html