



Java Flow Control (if, 4, then, break)

Mihai Iacob

Mai 2017



Despre mine:

- Java & BPM developer
- DevOps enthusiast
- Hobbies: moto, photo, drones, gadgets



Today's agenda:

1. Control Flow Statements
 1. If-then
 2. if-then-else
 3. Switch
2. Looping statements
 1. For
 2. While
 3. Do-while
3. Branching statements
 1. Break
 2. Continue
 3. Return
4. New Features



1. Control Flow Statements

1. if-then
2. if-then-else
3. Switch

- permite sa controlezi daca programul va executa o anumita sectiune de cod daca o conditie se evalueaza ca fiind adevarata sau falsa.

Structura standard:

```
if(conditie){  
    //cod care se executa daca conditia e adevarata  
}  
else {  
    //cod care se executa daca conditia e falsa  
}  
(a < 0) ? -a : a;
```



1. Control Flow Statements

1. If-then
2. if-then-else
3. Switch

- Spre deosebire de *if-then* si *if-then-else* block-ul switch poate avea mai multe cai de executie.
- Functioneaza doar cu tipuri de date primitive *byte*, *short*, *char* si *int* sau cu *Enum*-uri, clase *String*, sau obiecte care incapsuleaza anumite primitive: *Character*, *Byte*, *Short* si *Integer*.

Structura standard:

```
switch (variabila) {  
    case valoare_1: //cod care se executa 1  
        break;  
    case valoare_2: //cod care se executa 2  
        break;  
    default: //cod care se executa daca nu se executa alta ramura  
        break;  
}
```



2. Looping statements

1. for

2. While

3. Do-while

- Permite parcurgerea unei serie de valori

- Forma generala

```
for (cond initializare; cond finalizare; incrementare) {  
    //cod care se executa  
}
```

- *Cond initializare* – se executa o singura data la inceput
- *Cond finalizare* – cand se evalueaza la false bucla se opreste
- *Incrementarea* – este apelata la fiecare ciclu
- *Versiunea imbunatatita (for each)* – parcurgerea unei liste de valori (versiunea recomandata)



2. Looping statements

1. for

2. While

3. Do-while

- Este folosita pentru executia continua a unui bloc de cod cat timp o conditie este adevarata
- Asemnator cu *for* doar ca initializarea si incrementarea se fac independent

Structura standard:

```
while (expresie) {  
    //cod care se executa  
}
```



2. Looping statements

1. for
2. While
3. Do-while

- Asemănător cu *while* doar că expresia se evaluează la final – se execută cel puțin o dată

Structura standard:

```
do {  
    //cod care se execută  
} while (expresie);
```



3. Branching statements

1. Break

2. Continue

3. return

- Are 2 forme: cu eticheta si fara eticheta, a fost folosit la *switch*.
- Poate fi folosit pentru a intrerupe executia unui *for*, *while* sau *do-while*.
- Cu eticheta:
 - Este folosit pentru a intrerupe executia unei anumite comenzi, executia se continua dupa comanda search:

```
for (int i = 0; i < arrayOfInts.length; i++) {  
    for (int j = 0; j < arrayOfInts[i].length; j++) {  
        if (arrayOfInts[i][j] == searchfor) {  
            break search;  
        }  
    }  
}
```



3. Branching statements

1. Break
2. Continue
3. return

- Este folosita pentru a “sari” iteratia curenta a unei bucle *for*, *while* sau *do-while*
- La fel ca si *break* poate avea si versiunea cu eticheta care va “sari” executia buclei care e etichetata

```
for (int i = 0; i < max; i++) {  
    if (conditie)  
        continue;  
    //cod care se executa daca nu se indeplineste conditia  
}
```



3. Branching statements

1. Break
2. Continue
3. `return`

- Este folosit pentru a iesi din metoda curenta, iar executia va continua din punctual de unde a fost apelata metoda
- Are 2 forme: una care intoarce o valoare si una care nu intoarce valoare
`return ++count;`
`return;`



4. New Features

- Incepand cu Java 8 s-a introdus metode noi care se pot aplica pe obiecte de tip *List*, *Map*, *Set*; printer metodele adaugate se numara si *foreach*.
 - primeste ca parametru o functie
 - poate fi paralelizata executia foarte usor
 - se pot aplica filtrari pe lista de obiecte inainte sa se faca parcurgerea

```
<listOfValues>.forEach(i -> {  
    System.out.println(i + "-");  
});
```