KeyTraining

Curs java pentru testare automata

JAVA

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de Key Talents Srl

KeyTraining

Cuprins curs:

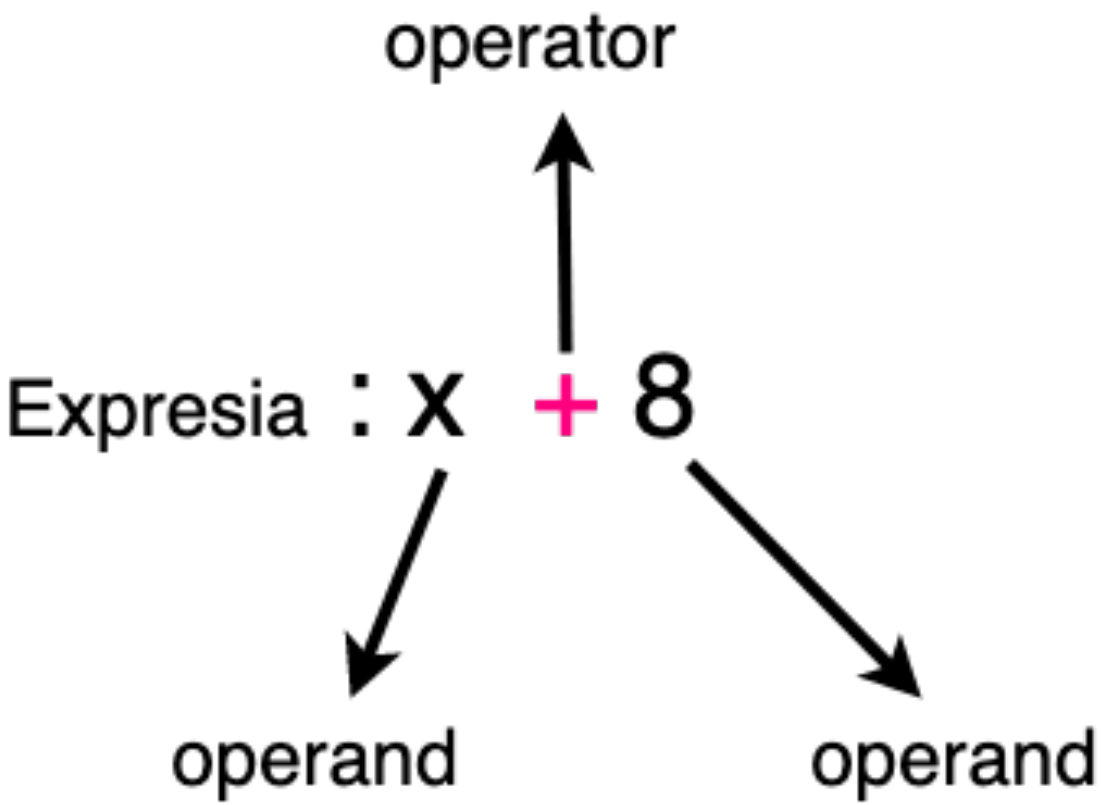
- Java Operators
- Increment/decrement
- For loop

Java Operators

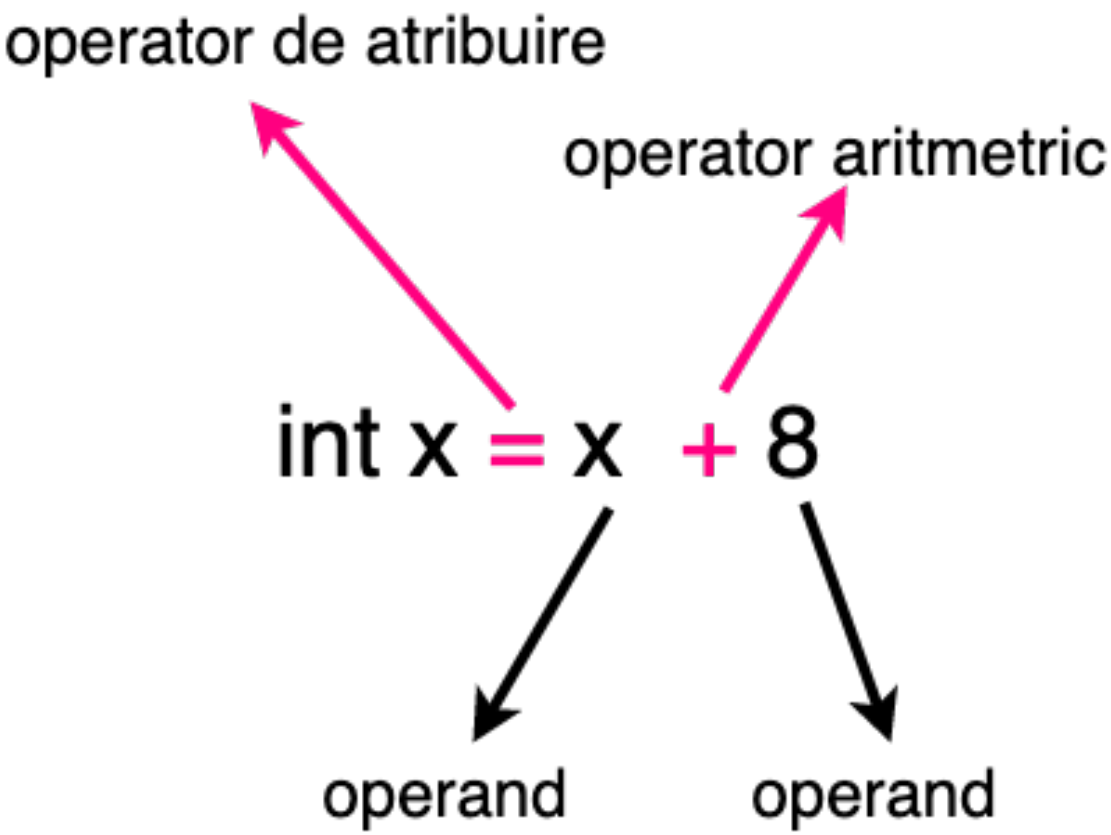
- În Java, o expresie are două părți: operand și operator. Variabilele sau constantele pe care acționează operatorii se numesc operanzi. Un operator în java este un simbol sau un cuvânt cheie care îi spune compilatorului să efectueze anumite operații matematice sau logice.
- Acestea sunt utilizate în programe pentru a manipula date și variabile.
- În general, formează expresii matematice sau logice.
- O expresie este o combinație de variabile, constante și operatori.

Java Operators

- De exemplu, o expresie este x + 8.
- Aici, operandul x este o variabilă, operandul 8 este o constantă și + este un operator care acționează asupra acestor doi operanzi și produce rezultatul dorit.



Java Operators



Java Operators

- Operator de atribuire compus

operator de atribuire compus

```
int x += 8
```

```
int x = x + 8
```

Java Operators

- **Tipuri de operatori:**

- 1. **Operator unar**: Operatorul care acționează asupra unui singur operand se numește operator unar. Un operator unar folosește o singură variabilă.
- 2. **Operator binar**: Operatorul care acționează asupra a doi operanzi se numește operator binar. Un operator binar folosește două variabile.
- 3. **Operator ternar**: Operatorul care acționează asupra a trei operanzi se numește operator ternar. Un operator ternar folosește trei variabile.

Java Operators

Clasificarea operatorilor:

1. Operator simbolic în Java:

- Operatori aritmetici: +, -, *, /, etc.
- Operatori relaționali: <,>, <=,> =, = =,! =.
- Operatori logici: &&, | |,!.
- Operatori de atribuire: =,
- Operatori de creștere și descreștere: + +, - -
- Operatori condiționati:?:
- Operatori bitwise: &,!, ^, ~, <<, >>, >>>
- Operatori de schimbare: <<, >>, >>>.

2. Operatori numițiîn Java : instanceOf operator

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Java Operators

OPERATORI ARITMETICI

Java Arithmetic Operators

- Operatorii aritmetici Java sunt obișnuiți să efectueze operații aritmetice fundamentale, cum ar fi adunarea, scăderea, înmulțirea și divizarea tipurilor de date numerice.
- Tipurile de date numerice pot fi char, byte, short, int, long, float și double.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Increment

- Operatorul care este utilizat pentru a crește valoarea unei variabile (operand) cu unu se numește operator de incrementare în java. Este reprezentat de (++). Operatorul ++ adaugă 1 la operand.
- Operatorul ++ poate fi utilizat în două forme:
  - **Pre increment** : Când scriem operatorul ++ înaintea unei variabile, se numește pre incrementare sau prefix. În pre-incrementare, creșterea se face mai întâi și apoi se va efectua orice altă operațiune.
  - **Post increment:** Când scriem operatorul ++ după o variabilă, se numește post incrementare sau postfix. În post-increment, toate celelalte operații sunt efectuate mai întâi, iar apoi la sfârșit, se face incrementul. Cu alte cuvinte, operatorul postfix stochează mai întâi valoarea în variabila din partea stângă și apoi operatorul adaugă 1 la un operand.

Decrement

- Operatorul care este folosit pentru a diminua valoarea unei variabile (operand) cu unu se numește operator de decrement în java. Este reprezentat de (--). Operatorul - scade 1 operandului.
- Operatorul -- poate fi utilizat în două forme:
  - **Pre decrement** : Când scriem -- operator înaintea unei variabile, se numește pre decrementare sau prefix. În pre-decrementare, decrementul se face mai întâi și apoi se va efectua orice altă operație. Cu alte cuvinte, operatorul de prefix scade mai întâi 1 unui operand, iar apoi rezultatul este stocat în variabila din partea stângă..
  - **Post decrement :** Când scriem operator -- după o variabilă, se numește post decrementare sau postfix. În post-decrement, toate celelalte operații sunt efectuate mai întâi, iar apoi la sfârșit se face numai decrementarea. Cu alte cuvinte, operatorul postfix stochează mai întâi valoarea în variabila din stânga și apoi operatorul scade 1 unui operand. Post decrement se efectuează după ce toate celelalte operațiuni sunt efectuate.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Java Operators

COMPARISON OPERATORS

Java Comparison Operators

- Operatorii relaționali/ de comparare din Java sunt acei operatori care sunt utilizați pentru a efectua comparația între două valori numerice sau două mărimi.
- Acești operatori determină relația dintre ei prin compararea operanzilor.
- De exemplu, să știți care este un număr mai mare, comparând vârsta a două persoane, comparând prețul a două articole etc.
- Aceste comparații pot fi efectuate cu ajutorul operatorilor relaționali.

Java Comparison Operators

- Java acceptă șase tipuri de operatori relaționali. Acestia sunt listati în tabelul de mai jos.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

- Rezultatul tuturor operatorilor relaționali este întotdeauna de tip boolean. Revine întotdeauna ca adevărat sau fals.

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Java Operators

LOGICAL OPERATORS

Java logical Operators

- Operatorii logici din Java sunt acei operatori care sunt utilizați pentru a forma condiții compuse prin combinarea a două sau mai multe condiții sau relații.
- Uneori, în Java, acești operatori sunt numiți și operatori booleani, deoarece returnează o valoare booleană.
- Un exemplu de operatori logici este : **x> y && x> z**
- Acest tip de expresie care combină două sau mai multe expresii relaționale se numește expresie logică sau expresie relațională compusă.
- Rezultatul expresiei logice este, de asemenea, o valoare adevărată sau falsă.

Java logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- AND Operator in Java &&
- În operatorul AND, două expresii (condiții) sunt combinate de operatorul &&.
- Dacă ambele condiții sunt adevărate, operatorul returnează adevărat.
- Dacă o expresie este falsă sau ambele expresii sunt false, atunci operatorul returnează fals.

Java logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- OR Operator in Java ||
- În operatorul OR, două sau mai multe expresii (condiții) sunt combinate cu | | (SAU) operator.
- Dacă oricare dintre condiții este adevărată, operatorul returnează adevărat.

Java logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- NOT Operator in Java !
- Operatorul NOT este utilizat pentru a inversa starea logică a operandului său. Dacă condiția este corectă, operatorul NU logic returnează fals.
- Dacă condiția este falsă, operatorul returnează adevărat.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Java Operators

ASSIGNEMENT OPERATORS

Java Assignement Operators

- Un operator care este utilizat pentru a stoca o valoare într-o anumită variabilă se numește operator de atribuire în java.
- În orice limbaj de programare, un operator de atribuire este cel mai frecvent utilizat pentru a atribui o valoare într-o variabilă.
- Există trei categorii de operații de atribuire în programarea java.
- Acestea sunt după cum urmează:
  a. Atribuire simplă
  b. Atribuirea compusă
  c. Atribuirea ca expresie

Java Assignement Operators

- O atribuire simplă poate fi utilizată în două moduri:
- Pentru a stoca sau a atribui o valoare unei variabile. Pentru a stoca o valoare a unei variabile într-o altă variabilă.
- Are următorul format general pentru a reprezenta o sarcină simplă :
- X = expresie;
- x: Este un nume variabil care reprezintă o locație de memorie în care poate fi stocată o valoare.
- expresie: poate fi o constantă, variabilă sau o combinație de constante, variabile și operatori.
- = este un operator de atribuire.

Java Assignement Operators

- Forma generală a atribuirii compuse este după cum urmează:
- x operator = expresie;
- operator : Este un operator binar Java. Poate fi + - * /% << >> etc.
- x și expresia sunt aceleași ca în explicația simplă de mai sus.
- De exemplu:
- x + = 8; // Este echivalent cu int x = x + 8;
- x - = 10; // Este echivalent cu int x = x - 10;
- a * = 100; // Echivalent cu int a = a * 100;

Java Assignement Operators

- In Java, o operație de atribuire este, de asemenea, considerată o expresie, deoarece operația are un rezultat.
- Rezultatul expresiei este o valoare stocată într-o variabilă.
- De exemplu:
- int x = y - z + 3; // Aici, expresia y - z + 3 este evaluată mai întâi și apoi rezultatul acesteia este stocat în variabila x.

Java Ternary Operator

- Perechea de caractere ? : se numește operator ternar în Java deoarece acționează asupra a trei variabile.
- Acest operator este, de asemenea, cunoscut sub numele de operator condițional în Java.
- Se folosește pentru a face o expresie condițională.
- În java are următoarea sintaxă:
- variabilă = expresie1**?** expresie2 **:** expresie3;

Java Ternary Operator

- variabilă = expresie1**?** expresie2 **:** expresie3;
- În primul rând, se evaluează expresia expresie1.
- Dacă este adevărata, expresia expresie2 va fi evaluată și valoarea expresie2 va fi stocată în variabilă.
- Dacă expresie1 este fals, expresie3 va fi evaluat și valoarea acestuia va fi stocată în variabilă.
- Luați în considerare următorul exemplu:
- int a = 60;
- int b = 30;
- int x = (a> b)? a: b;

Java Ternary Operator

- int a = 60;
- int b = 30;
- int x = (a> b)? a: b;
- Aici, primul (a> b) este evaluat. Dacă este adevărata, valoarea lui a este stocată în variabila x. Dacă este falsă, valoarea lui b este stocată în variabila x.
- Aici, (a> b) este adevărat deoarece valoarea lui a este mai mare decât valoarea lui b. Prin urmare, valoarea lui va fi atribuită în variabila x.
- Adică x = 60.
- În exemplul de mai sus, am folosit 3 variabile a, b și x.
- De aceea se numește operator ternar în java.

Java Ternary Operator

- int a = 60;
- int b = 30;
- int x = (a> b)? a: b;
- Acest exemplu poate fi realizat și prin utilizarea instrucțiunii if-else după cum urmează:

```
if (a> b) {
 x = a;
 }
else {
 x = b;
 }
```

Java bitwise Operator

- Un operator care acționează asupra biților individuali (0 sau 1) ai operanzilor se numește operator bit în Java.
- Acționează numai asupra tipuri de date întregi, cum ar fi byte, scurt, int și lung.
- Operatorii bitwise din java nu pot fi aplicați tipurilor de date float și double.
- Reprezentarea internă a numerelor în cazul operatorilor bitwise este reprezentată de sistemul numeric binar.
- Numărul binar este reprezentat de două cifre 0 sau 1. Prin urmare, acești operatori sunt utilizați în principal pentru a modifica tiparele de biți (reprezentare binară).

Java bitwise Operator

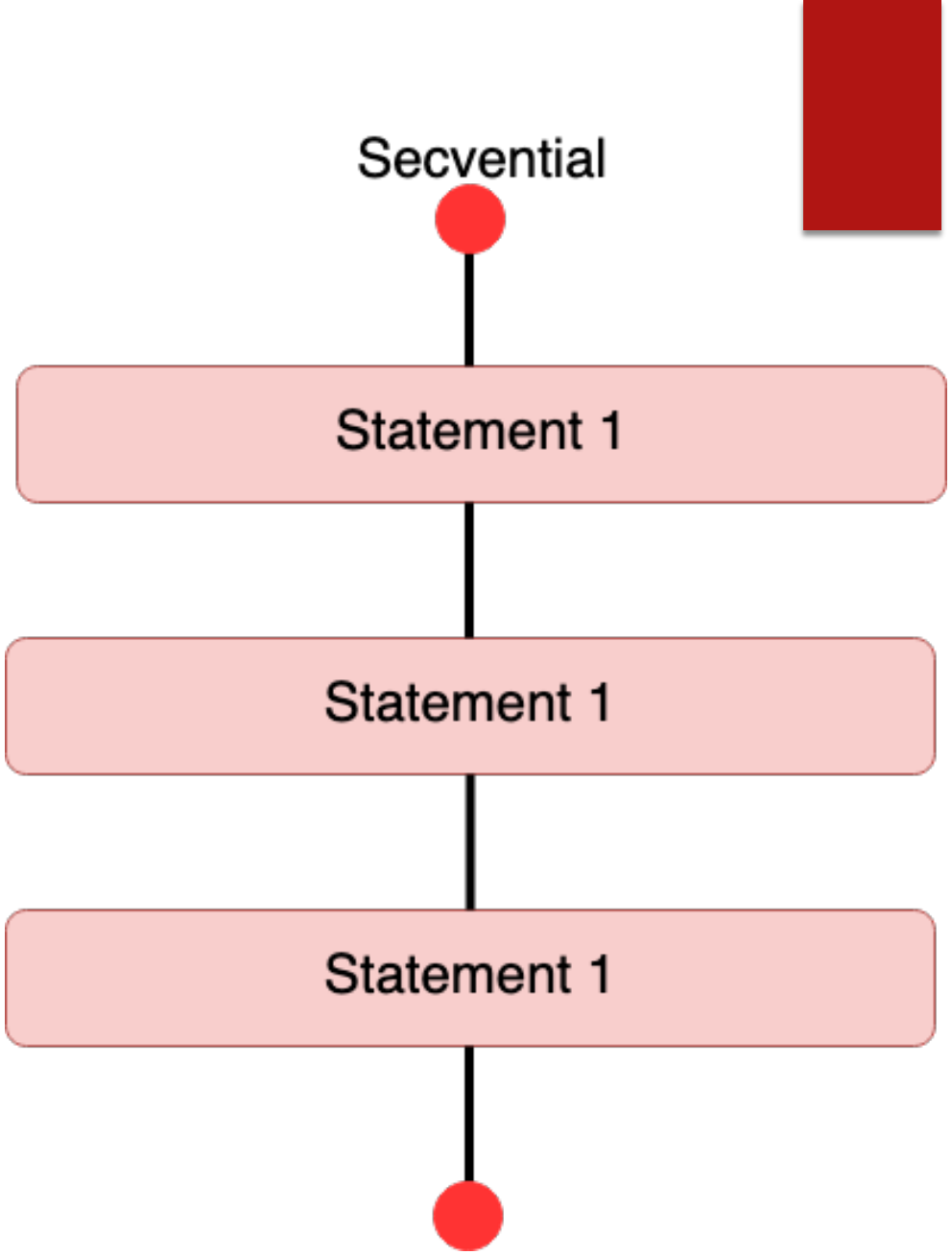
- În Java, există șapte tipuri de operatori bitwise
- 1. & bitwise AND
- 2. | bitwise OR
- 3. ^ bitwise exclusive OR
- 4. ~ one's complement (unary)
- 5. << shift left
- 6. >> shift right
- 7. >>> unsigned right shift

Conditional statements

- Există trei tipuri principale de flux de execuție (control) care apar în programare.
- 1. Secvențial: instrucțiunile se execută de sus în jos una câte una.
- 2. Condițional sau selecție: Din două instrucțiuni, doar una va fi executată cu succes pe baza condiției specificate. Acest lucru se datorează faptului că condiția generează rezultatul fie adevărat, fie fals.
- 3. Repetare sau buclă: Grupul de instrucțiuni se repetă ori de câte ori condiția specificată este adevărată.

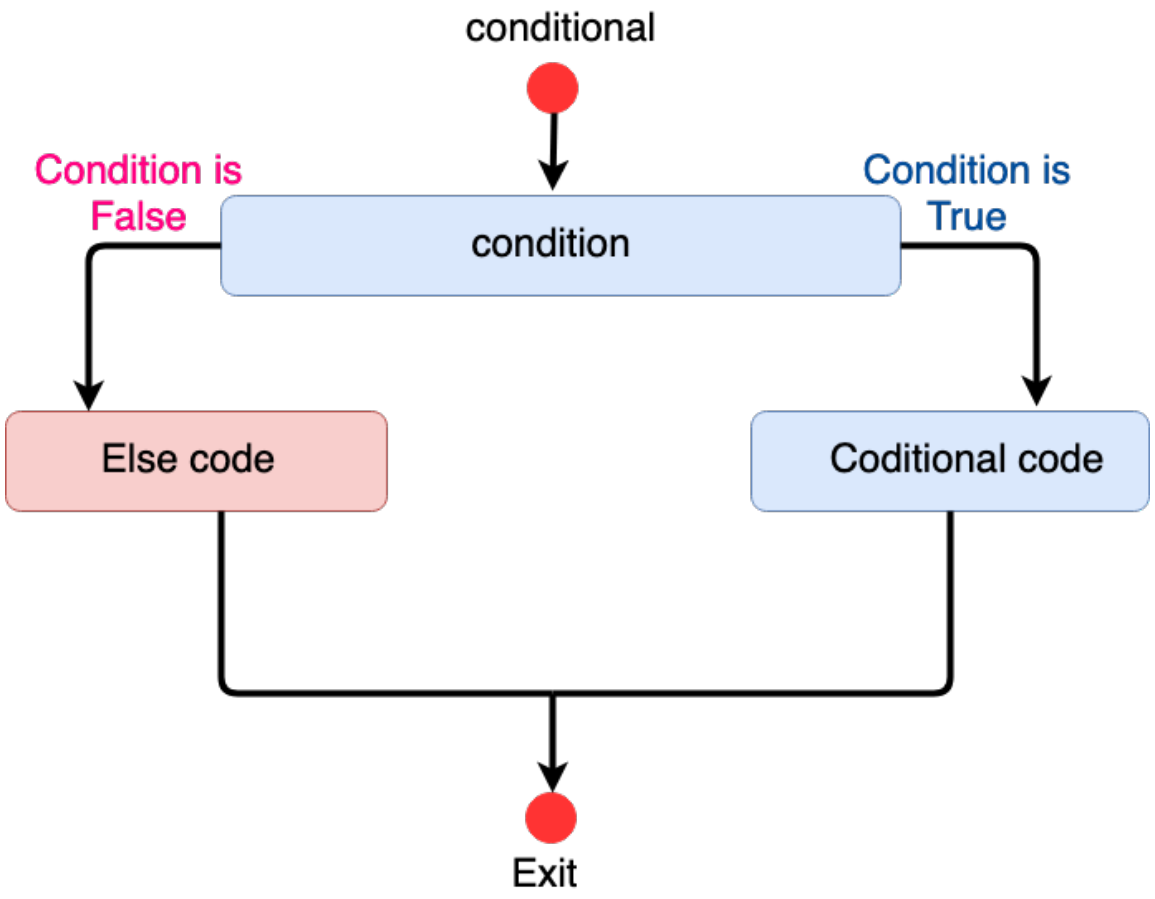
Conditional statements

- 1. Secvențial: instrucțiunile se execută de sus în jos una câte una.



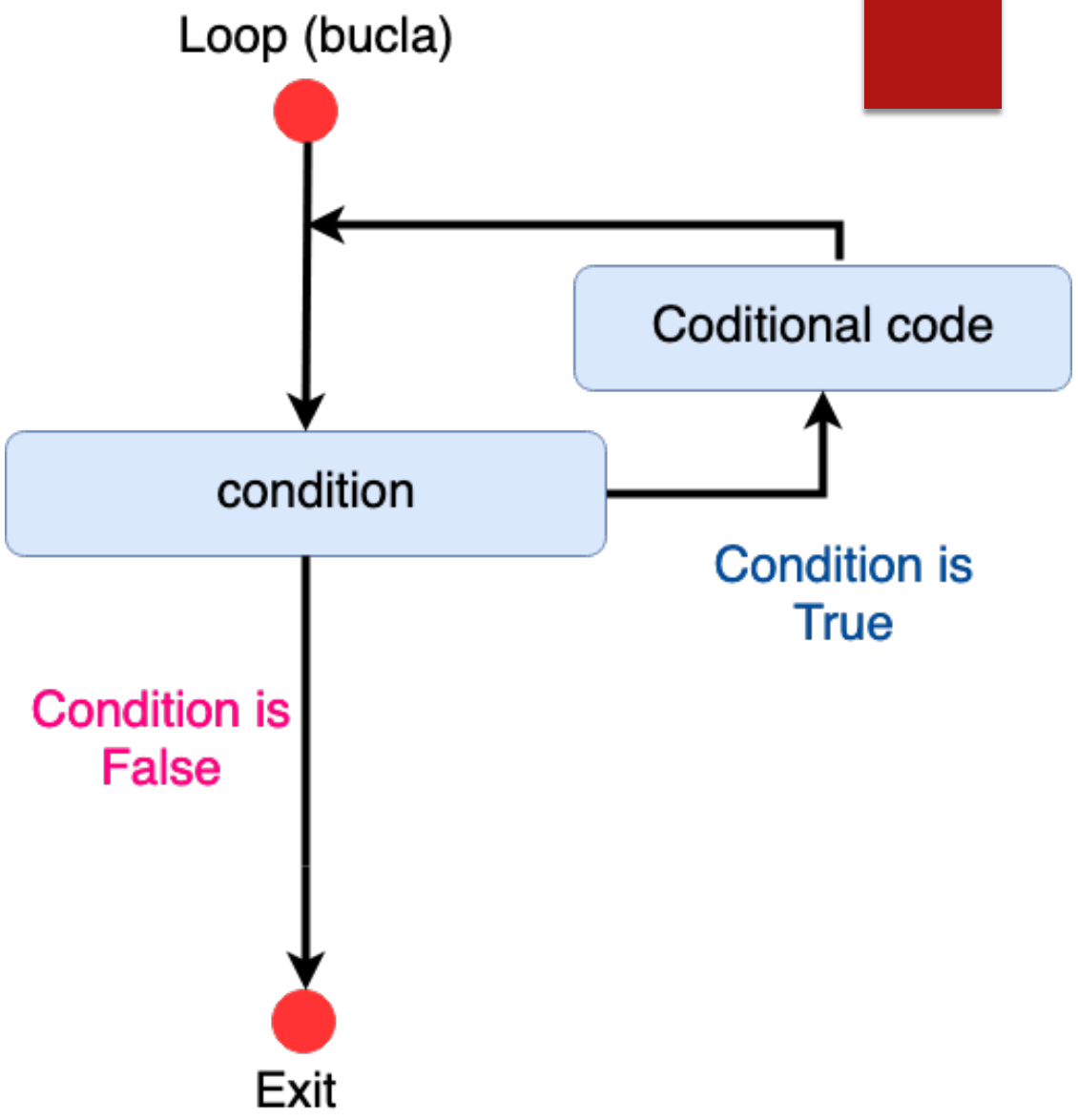
Conditional statements

- 2. Condițional sau selecție: Din două instrucțiuni, doar una va fi executată cu succes pe baza condiției specificate. Acest lucru se datorează faptului că condiția generează rezultatul fie adevărat, fie fals.



Conditional statements

- 3. Repetare sau buclă: Grupul de instrucțiuni se repetă ori de câte ori condiția specificată este adevărată.



Instrucțiunile de control

- Instrucțiunile de flux de control în Java sunt acele instrucțiuni care schimbă fluxul de execuție și oferă un control mai bun programatorului asupra fluxului de execuție în program.
- Aceste declarații sunt executate aleatoriu și în mod repetat. Instrucțiunile de control din programarea java sunt utilizate pentru a scrie programe mai bune și complexe.
- De exemplu, să presupunem că apare o situație în program, în care trebuie să schimbăm ordinea de execuție a instrucțiunilor pe baza unor condiții specifice sau să repetăm un grup de instrucțiuni până când sunt îndeplinite anumite condiții specifice.

Instrucțiunile de control

- Astfel de situații pot fi realizate cu ajutorul instrucțiunilor de flux de control sau pur și simplu numite instrucțiuni de control condițional în Java.
- Instrucțiunile de control sunt, de asemenea, numite structuri de control.
- Java acceptă două tipuri de instrucțiuni de control:
- condiționate
- necondiționate

Conditionate

- Un program care sparge fluxul secvențial și sare într-o altă parte a codului se numește ramificare în java.
- Când ramificarea se bazează pe o condiție specifică, atunci se numește ramificare condițională sau instrucțiuni condiționale în Java.
- Deoarece fluxul de execuție este transferat cu o anumită condiție dintr-o parte a codului în altă parte, se mai numește și execuție condiționată.

Conditionate

- Java acceptă următoarea instrucțiune de flux condiționat de control:

- if …else statement
- do …while statement
- while loop
- for loop
- for-each loop
- switch statement

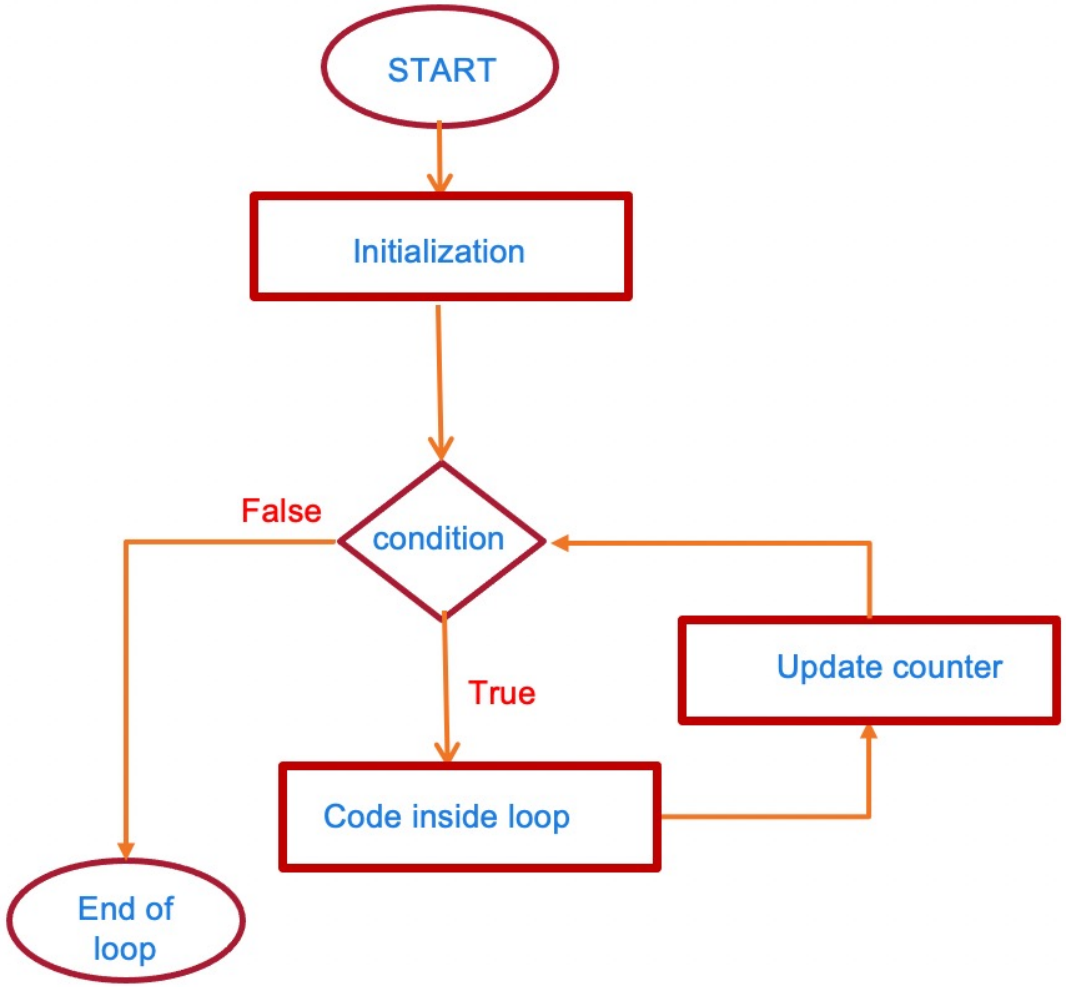
- Instrucțiunea fluxului de control condițional folosește expresia booleană pentru testele condiționale. Dacă există doar o singură instrucțiune în instrucțiunea fluxului de control condițional, acoladele sunt opționale.

Neconditionate

- Când fluxul de execuție sare la o altă parte a codului fără a efectua niciun test condiționat, se numește instrucțiuni necondiționate sau execuție necondiționată în Java.
- Următoarele declarații necondiționate sunt disponibile în Java:
- break statement
- continue statement
- return statement

For loop

- FOR LOOP din Java este o structură de tip buclă controlată care execută un set de instrucțiuni de un număr fix de ori.
- Instrucțiunea for oferă o sintaxă mai concisă pentru crearea de bucle. Execută un bloc de declarații atâta timp cât condiția este adevărată.

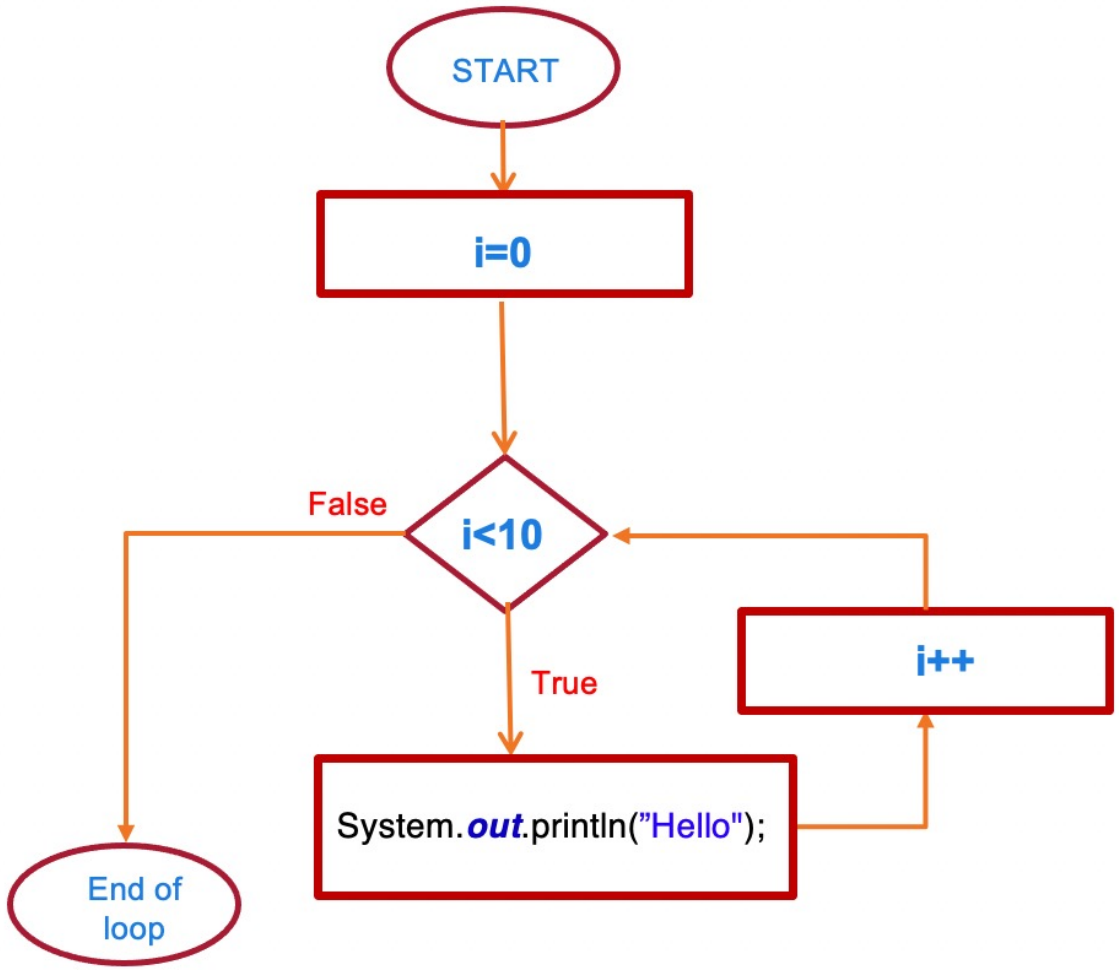

START
Initialization
False
condition
True
Update counter
Code inside loop
End of loop

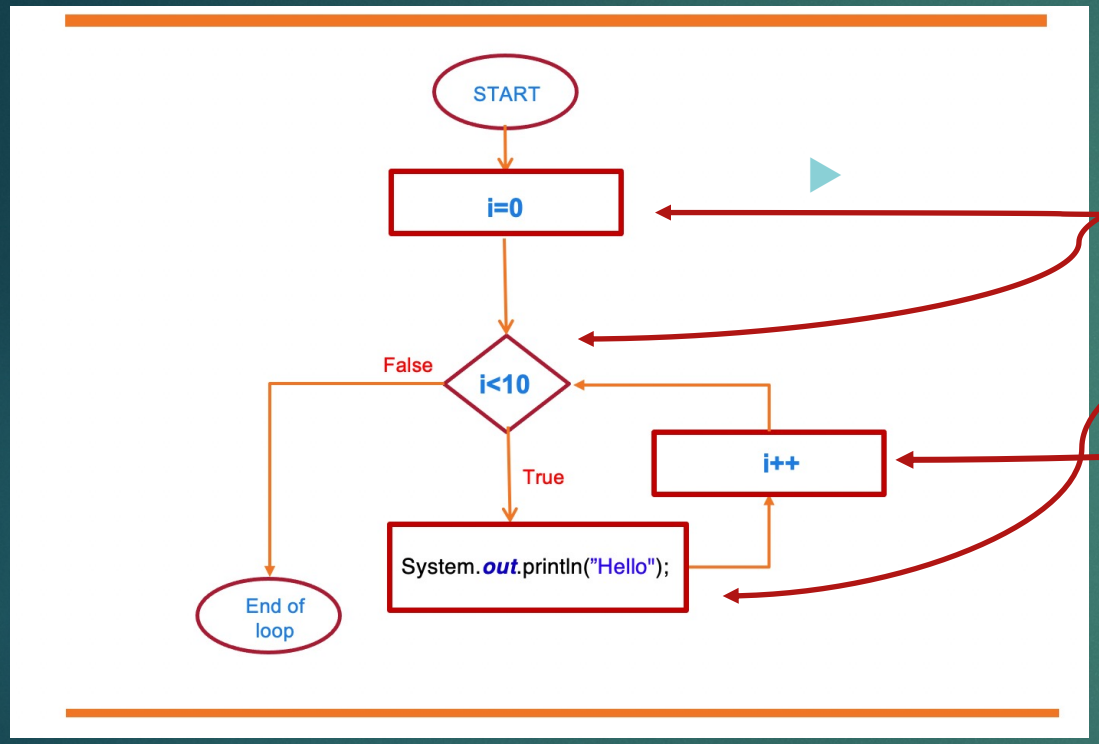
```
for (initialization, test-condition, iteration(increment/decrement)){

    //code statements
}
```

```java
for(int i=0; i< 10; i++) {

    System.out.println("Code inside loop : " + i);
}
```

```
for (initialization, test-condition, iteration(increment/decrement)){

    //code statements
}
```

```
for(int i=0; i< 10; i++) {

    System.out.println("Code inside loop : " + i);
}
```


START
i=0
i<10
False
True
i++
System.out.println("Hello");
End of loop




```
for(int i=0; i< 10; i++) {

    System.out.println("Code inside loop : " + i);
}
```

For Loop –ordinea executiei

- **Inițializarea variabilelor de control:** Un for loop utilizează în general o variabilă pentru a controla de câte ori corpul buclei va fi executat și când corpul buclei se termină.
- Această variabilă se numește variabilă de control. Se inițializează utilizând un operator de atribuire precum i = 1 și count = 0. Variabilele i și count sunt numite variabile de control buclă. Se efectuează o singură dată când se execută o instrucțiune for.

For Loop –ordinea executiei

- **Test-condition**: Valoarea variabilei de control este evaluată folosind condiția de testare.
- Condiția de testare este o expresie booleană, cum ar fi i <10 care determină când va ieși bucla. Este evaluat înainte de fiecare iterație de buclă (execuție).
- Când condiția este adevărată, corpul buclei este executat.
- Când este fals, bucla este terminată și execuția continuă la următoarea instrucțiune după închiderea unei acolade care urmează imediat bucla.

For Loop –ordinea executiei

- **Iteration**: Când corpul buclei este executat, controlul execuției este transferat înapoi la instrucțiune după executarea ultimei instrucțiuni din buclă.
- Variabila de control este incrementată sau decrementată folosind instrucțiuni de atribuire precum i ++ (adică i = i + 1) sau i— (adică i = i - 1), iar noua valoare a variabilei de control este evaluată din nou pentru a vedea dacă îndeplinește condiția sau nu.
- Când condiția este adevărată, corpul buclei este executat.
- Când este fals, bucla este terminată și execuția continuă la următoarea instrucțiune aflata după închiderea acoladei care închide bucla.