Operadores y Flujo de control

Programación y Laboratorio III

Agenda

- **Operadores**
  - Operadores de asignación
  - Operadores aritméticos
  - Operadores relacionales
  - Operadores lógicos
  - Precedencia de operadores
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Operadores

Tipo	Operadores	Propósito
Asignación	`=, +=, -=, *=, /=`	Asignar valor a una variable
Aritmético	`+, -, *, /, %, ++, --`	Sumar, restar, multiplicar, dividir y módulo de primitivas
Relacional	`<, <=, >, >=, ==, !=`	Comparan primitivas
Lógico	`!, &&, ||`	Aplicar NOT, AND y OR a primitivas

Agenda

- **Operadores**
  - **Operadores de asignación**
  - Operadores aritméticos
  - Operadores relacionales
  - Operadores lógicos
  - Precedencia de operadores
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Operadores de asignación

= → usado para inicializar variables con valores o reasignar nuevos valores.

```
a -= b → a = a - b

a += b → a = a + b

a *= b → a = a * b

a /= b → a = a - b

a %= b → a = a % b
```

Operadores de asignación - Ejemplos

```
double d = 10.2;
int a = 10;
int b = a;
float f = 10.2F;


b += a;   // b = 20
a = b = 10;
b -= a;  // b = 0


d = true;
boolean b = 0;
boolean bTrue = true;
boolean bFalse = false;
bTrue -= bFalse;

long num = 100976543356L;
int val = num;  // No se permite

int num1 = 1009;
long val1 = num1;  // Se permite
```

Agenda

- **Operadores**
  - ~~Operadores de asignación~~
  - **Operadores aritméticos**
  - Operadores relacionales
  - Operadores lógicos
  - Precedencia de operadores
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Operadores aritméticos

Operador	Propósito
+	Sumar
-	Restar
*	Multiplicar
/	Dividir
%	Resto en la división
++	Incrementa en 1
--	Decrementa en 1

Operadores aritméticos - Ejemplos

```java
char char1 = 'a';
System.out.println(char1 + char1); //194


byte age1 = 10;
byte age2 = 20;
short sum = age1 + age2;  //Tipos incompatibles


int a = 20;
int b = 10;
++a;
b++;
System.out.println(a);  // 21
System.out.println(b);  // 11


int a = 20;
int b = 10;
int c = a - ++b;
System.out.println(c);  // 9
System.out.println(b);  // 11


int a = 20;
int b = 10;
int c = a - b++;
System.out.println(c);  // 10
System.out.println(b);  // 11
```

Agenda

- **Operadores**
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - **Operadores relacionales**
  - Operadores lógicos
  - Precedencia de operadores
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Operadores relacionales

- Se usan para determinar si el valor de una primitiva es igual, menor o mayor al valor de otra.

Operador	Uso
`>, >=, <, <=`	Comparan mayor y menor.
`==, !=`	Comparan igualdad.

Operadores relacionales - Ejemplos

```java
int i1 = 10;
int i2 = 20;
System.out.println(i1 >= i2);   // false


long l1 = 10;
long l2 = 20;
System.out.println(l1 <= l2);  // true


int a = 10;
int b = 20;
System.out.println(a == b);  // false
System.out.println(a != b);  // true


boolean b1 = false;
System.out.println(b1 == true);  // false
System.out.println(b1 != false);  // false
System.out.println(b1 == false);  // true
```

Agenda

- **Operadores**
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - **Operadores lógicos**
  - Precedencia de operadores
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Operadores lógicos

- Se utilizan para evaluar una o más expresiones. La evaluazión retorna un valor `boolean`.

Operador	Uso
&&	AND
||	OR
!	NOT

Operadores lógicos - Ejemplos

```java
int a = 10;
int b = 20;
System.out.println(a > 20 && b < 10);  // false
System.out.println(a > 20 || b > 10);  // true
System.out.println(!(b > 10));  // false
System.out.println(!(a > 20));  // true


int marks = 8;
int total = 10;
System.out.println(total < marks && ++marks > 5);  // false
```

Agenda

- **Operadores**
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - ~~Operadores lógicos~~
  - **Precedencia de operadores**
- Control de flujo
  - `if, if-else`
  - `switch`
  - `for`
  - `while, do-while`

Precedencia de operadores

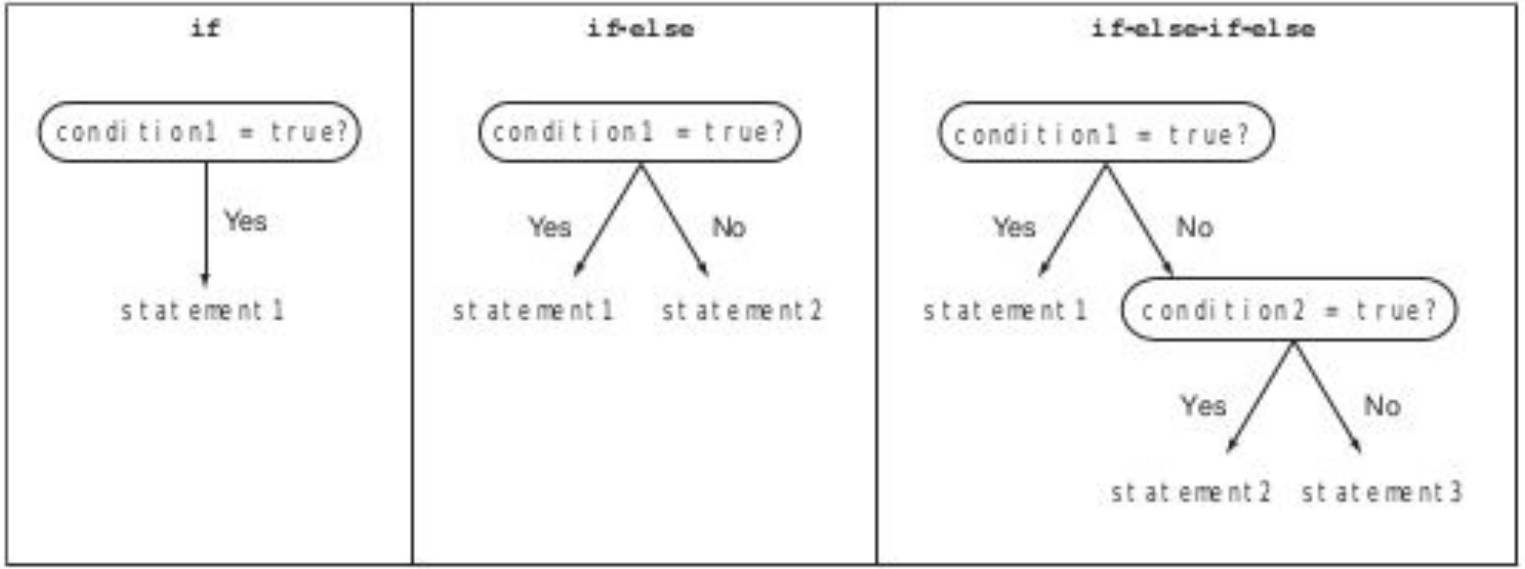
Operador
`expresion++, expresion--`
`++expresion, --expresion, +expresion, -expresion, !`
`*, /, %`
`+, -`
`<, >, <=, >=`
`==, !=`
`&&`
`| |`
`=, +=, -=, *=, /=, %=`

Agenda

- ~~Operadores~~
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - ~~Operadores lógicos~~
  - ~~Precedencia de operadores~~
- **Control de flujo**
  - **`if, if-else`**
  - `switch`
  - `for`
  - `while, do-while`

Control de flujo: `if`, `if-else`

- Nos permite ejecutar una serie de sentencias, según el resultado de una condición.
- El resultado de evaluar la condición debe ser `boolean` o `Boolean`.



if, if-else - Ejemplos

```
//if
if (b == true) {
    score = 200;
}


//if-else
if (b == true) {
    score = 200;
} else {
    score = 300;
}


if (score == 200) {
    result = 'A';
} else if (score == 300) {
    result = 'B';
} else {
    result = 'C';
}
```

Agenda

- ~~Operadores~~
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - ~~Operadores lógicos~~
  - ~~Precedencia de operadores~~
- **Control de flujo**
  - ~~if, if-else~~
  - **switch**
  - for
  - while, do-while

Control de flujo: `switch`

- Se utiliza cuando la variable a evaluar tiene múltiples valores.

```
switch (value) {
     case sth1 :
         statements;
         break;
     case sth2 :
         statements;
         break;
     default :
        statements;
        break;
}
```

switch - Ejemplo

```java
int marks = 20;

switch (marks) {
    case 10 : System.out.println(10);
        break;
    case 20 : System.out.println(20);
        break;
    case 30 : System.out.println(30);
        break;
    default : System.out.println("default");
        break;
}
```

Agenda

- ~~Operadores~~
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - ~~Operadores lógicos~~
  - ~~Precedencia de operadores~~
- **Control de flujo**
  - ~~if, if-else~~
  - ~~switch~~
  - **for**
  - while, do-while

Control de flujo: `for`

- Se utiliza cuando se necesita repetir la(s) misma(s) línea(s) de código múltiples veces.

```
for (initialization; condition; update) {
     statements;
}
```

for - Ejemplo

```java
int tableOf = 25;
for (int ctr = 1; ctr <=5; ctr++) {
    System.out.println(tableOf * ctr);
}


for (int hrs = 1; hrs <=6; hrs++) {
    for (int min = 1; min <= 60; min++) {
        System.out.println(hrs + ":" + min);
    }
}
```

Agenda

- ~~Operadores~~
  - ~~Operadores de asignación~~
  - ~~Operadores aritméticos~~
  - ~~Operadores relacionales~~
  - ~~Operadores lógicos~~
  - ~~Precedencia de operadores~~
- **Control de flujo**
  - ~~`if, if-else`~~
  - ~~`switch`~~
  - ~~`for`~~
  - `while, do-while`

Control de flujo: `while`, `do-while`

- Ejecutan una serie de sentencias hasta que la condición de corte sea igual a `true`.
- La principal diferencia entre ambos es que `while` chequea la condición antes de ejecutar el cuerpo, mientras que en `do-while` evalúa la condición después de ejecutar las sentencias definidas en el cuerpo.

```
while (condition) {
    statements;
}
```

```
do {
    statements;
} while (condition);
```

while, do-while - Ejemplos

```java
int num = 9;
boolean divisibleBy7 = false;

while (!divisibleBy7) {
    System.out.println(num);
    if (num % 7 == 0) {
        divisibleBy7 = true;
    }
    --num;
}
```

```java
int num = 9;
boolean divisibleBy7 = false;
do {
    System.out.println(num);
    if (num % 7 == 0) {
        divisibleBy7 = true;
    }
    num--;
} while (divisibleBy7 == false);
```