## Operatori

In Java, gli operatori producono sempre un risultato, e alcuni possono anche modificare uno degli operandi. Alcuni operatori sono rappresentati simbolicamente con un singolo carattere, altri con due o tre caratteri senza spazi separatori.

Gli operatori si dividono in:

* Operatori di assegnamento
* Operatori aritmetici
* Operatori relazionali
* Operatori logici
* Operatori sui bit
* Operatori su stringhe e oggetti.

**Operatori di assegnamento**

*L’operatore di assegnazione* o *assegnamento*, è il carattere “=”. *L’assegnazione* avviene secondo la sintassi:

|  |
| --- |
| <**variabile**> **=** <**espressione**>**;** |

Utilizzando *l’operatore di assegnazione,* l’*espressione* alla destra del carattere ”=” viene valutata e il risultato viene assegnato alla variabile posta alla sinistra del “=”.

|  |
| --- |
| **Un esempio:**    int a;  int b;  a = 10; // assegno alla variabile a il valore 10  b = 0; // assegno alla variabile b il valore 0  b = 2 \* a; // assegno alla variabile b il doppio del valore di a  /\* ora la variabile b vale 20 \*/ |

In alcuni casi particolari, può essere utile assegnare ad una variabile un’espressione che già contiene un’assegnazione:

|  |
| --- |
| **Un esempio:**    int a;  int b;  int c;  a = 10; // assegno alla variabile a il valore 10  c = (b = 2 \* a); // assegno alla variabile b il doppio del  // valore di a, e alla variabile c il valore di b    /\* ora la variabile b vale 20 e la variabile c pure \*/ |

## Operatori aritmetici

Gli operatori aritmetici si dividono in *Operatori Binari* e *Operatori Unari.* I primi, si applicano a due operandi mentre i secondi si applicano ad un operando solo.

**Operatori Binari**

Gli operatori binari matematici si applicano a due operandi di cui non cambiano il valore, e producono un risultato. Si utilizzano nella forma:

|  |
| --- |
| <**operando 1**> <**operatore**> <**operando 2**>**;** |

In Java esistono i seguenti operatori binari:

|  |  |  |
| --- | --- | --- |
| **Operatore** | **Simbolo** | **Esempio** |
| Addizione | + | Somma due operandi o concatena due stringhe |
| Sottrazione | - | Sottrae il secondo operando al primo |
| Prodotto | \* | Moltiplica i due operandi |
| Divisione | / | Divide il primo operando per il secondo. Applicato a variabili di tipo intero produce un risultato intero, troncato della parte decimale |
| Resto (Modulo) | % | Fornisce il resto della divisione intera. Può essere usato solo con operandi interi. |

In genere gli operatori sono utilizzati in espressioni del tipo:

|  |
| --- |
| <**nome variabile**> **=** <**nome variabile**> <**operatore**> <**operando 2**>**;** |

In tal caso l’espressione può anche essere scritta nella forma compatta:

|  |
| --- |
| <**nome variabile**> <**operatore**>**=** <**operando 2**>**;** |
| **Un esempio:**    a = a + 1;  // equivale a  a += 1; |

**Operatori Relazionali**

Gli operatori relazionali richiedono operandi di tipo primitivo. Questi operatori eseguono un confronto dei valori dei due operandi e producono un risultato booleano.

|  |  |
| --- | --- |
| **Operatore** | **Simbolo** |
| Maggiore | > |
| Maggiore o uguale | >= |
| Minore | < |
| Minore o uguale | <= |
| Uguale | == |
| Diverso | != |

|  |
| --- |
| **Alcuni esempi:**    a > b // retituisce true se a è maggiore di b  a != 0 // retituisce true se a è diverso da 0  a == 0 // retituisce true se a è uguale a 0 |

**Operatori Logici**

Gli operatori logici si applicano a operandi booleani e producono un risultato booleano. Java dispone dei seguenti operatori binari:

|  |  |  |
| --- | --- | --- |
| **Operatore** | **Simbolo** | **Significato** |
| OR | ⎮⎮ | OR logico, disgiunzione |
| AND | && | AND logico, congiunzione |
| NOT | ! | Negazione |
| XOR | ^ | OR esclusivo, disgiunzione esclusiva |

|  |
| --- |
| **Alcuni esempi:**    a || b // a OR b  a && 0 // a AND b  a ^ b // a XOR b  !(a && b) // NOT(a AND b) |

**Operatori su bit**

Gli operatori “bit a bit” richiedono operandi di tipo primitivo. Questi operatori operano rappresentazione binaria degli operandi.

|  |  |
| --- | --- |
| **Operatore** | **Simbolo** |
| AND | & |
| OR | | |
| XOR | ^ |
| Complemento a uno | ~ |
| Shift a destra | >> |
| Shift a destra unsigned | >>> |
| Shift a sinistra | << |

|  |
| --- |
| **Alcuni esempi:**    int a = 1; // 0000’0000’0000’0000’0000’0000’0000’0001  int b = 2; // 0000’0000’0000’0000’0000’0000’0000’0010  int c = a | b; // 0000’0000’0000’0000’0000’0000’0000’0011  …  int a = 1; // 0000’0000’0000’0000’0000’0000’0000’0001  int b = 3; // 0000’0000’0000’0000’0000’0000’0000’0011  int c = a & b; // 0000’0000’0000’0000’0000’0000’0000’0001  …  int a = 1; // 0000’0000’0000’0000’0000’0000’0000’0001  int b = a << 1; // 0000’0000’0000’0000’0000’0000’0000’0010  int c = a << 2; // 0000’0000’0000’0000’0000’0000’0000’0100 |