## Premessa

Un programma sorgente è formato da un insieme ordinato di istruzioni (o comandi), che in Java possono essere:

- ✓ di dichiarazione, quando definiscono le strutture di dati che traducono il modello dei dati del problema creato in fase di progettazione;
- ✓ eseguibili, quando codificano nel programma sorgente le operazioni, descritte nell'algoritmo del progetto, necessarie per passare dai dati in ingresso ai risultati.

### *Regole di valutazione di un'espressione*

Le espressioni che incontriamo negli algoritmi sono spesso formate da combinazioni complesse di operandi e di operatori. Le regole di valutazione permettono di assegnare a una espressione un valore unico, stabilendo l'ordine con cui devono essere applicati gli operatori agli operandi, che in Java avviene sulla base dei seguenti criteri.

| | |
|---|---|
| **1. L'impiego delle parentesi tonde** | L'impiego delle parentesi tonde impone al compilatore di eseguire per prime le espressioni racchiuse nelle coppie di simboli (). |
| **2. Il livello (o ordine) di precedenza** | Il linguaggio suddivide l'insieme di tutti gli operatori in gruppi ai quali assegna il medesimo livello di precedenza (mostrato nella tabella che segue). Se in un'espressione compaiono operatori con livelli di precedenza diversi, viene valutato per primo quello che ha una precedenza maggiore. |
| **3. Le regole di associatività** | Se in un'espressione vi sono più operatori con il medesimo livello di precedenza, allora:<br>✓ gli operatori binari (eccetto quello di assegnazione) sono valutati da sinistra a destra (associatività verso destra);<br>✓ gli operatori unari, quello ternario e quelli di assegnazione sono valutati da destra a sinistra (associatività verso sinistra). |

La tabella seguente presenta i livelli di precedenza e il tipo di associatività dei gruppi degli operatori del Java. Il tipo di associatività è riportato nell'ultima riga indicando $S \leftarrow D$, per gli operatori valutati da destra a sinistra, e $S \rightarrow D$, per gli operatori calcolati da sinistra a destra.

| *Precedenza maggiore* | | | | | | | | | | | | | | *Precedenza minore* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| . [] () , | $++\ (prefisso)$<br>$--\ (prefisso)$<br>$+\ (unario)$<br>$-\ (unario)$<br>~<br>!<br>$(tipo\ di\ dato)$ | * / % | + − | << >> >>> | < > <= >= *instanceof* | == ! = | & | ^ | \| | && | \|\| | ? | = *op* = | $++\ (postfisso)$<br>$--\ (postfisso)$ |
| $S \rightarrow D$ | $S \leftarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \rightarrow D$ | $S \leftarrow D$ | $S \leftarrow D$ |

Nella tabella precedente, gli operatori della prima colonna, che incontreremo nel seguito dello studio, sono considerati dal Java come separatori. L'operatore . è l'operatore "punto" per l'accesso degli attributi/metodi di un oggetto, [ ] definisce un oggetto array, ( ) rappresenta la chiamata di un metodo e , è l'operatore virgola. Sempre nella tabella precedente, $instanceof$ viene applicato agli oggetti per verificare il tipo di classe, $op =$ (con $op$ sostituito da $+$, $-$, $*$ ecc.) simboleggia una forma particolare di assegnazione e $(tipo\ di\ dato)$ indica l'operazione di cast (conversione di tipo).

### *Operatori relazionali o di confronto*

Gli operatori relazionali effettuano dei confronti tra due operandi (di tipo numerico e `char`) restituendo come risultato un valore booleano (*true* o *false*). I tipi di confronti disponibili nel Java con i corrispondenti operatori relazionali sono descritti nella tabella che segue.

| Operazione | Operatore relazionale |
|---|---|
| uguale a | == |
| diverso da | ! = |
| minore | < |
| minore o uguale | <= |
| maggiore | > |
| maggiore o uguale | >= |