Azzolini Riccardo 2018-10-12

`if-else`, tipo `boolean` e `do...while`

1 Istruzioni `if-else` e `if`

```
if (condizione)
    istruzione1;
else
    istruzione2;
```

```
if (condizione)
    istruzione1;
```

- La `condizione` è una qualunque espressione di tipo `boolean`
- `istruzione1` (*ramo then*) e `istruzione2` (*ramo else*) possono essere
  - istruzioni singole
  - blocchi di istruzioni racchiusi tra parentesi graffe

Semantica operazionale:

1. viene valutata la `condizione`
   - se è vera, viene eseguita l'`istruzione1`
   - se è falsa, viene eseguita l'`istruzione2` (se presente, cioè se non è stato omesso il ramo else)
2. l'esecuzione prosegue dall'istruzione successiva alla struttura `if-else`

1.1 `if-else` innestati

Siccome `if-else` è un'istruzione, si può utilizzare nel corpo di un altro `if-else`. In questo caso, le due (o più) strutture si dicono `if-else` **innestati** (o **nidificati**, o **nested**).

In assenza di parentesi graffe, ogni `else` si associa al primo `if` che lo precede per il quale *non sia ancora stato identificato* un `else`.

2 Tipo primitivo `boolean`

Ha due valori, denotati dai letterali `true` e `false`.

Un'espressione che restituisce un valore di tipo `boolean` si dice **condizione**.

2.1 Operatori relazionali

Le condizioni più semplici sono quelle composte da un **operatore relazionale** (`< <= > >= == !=`), che effettua un confronto tra due espressioni di tipo primitivo.

2.2 Confronto tra riferimenti

Tra tipi riferimento, gli operatori `==` e `!=` confrontano i riferimenti stessi: se `a` e `b` sono variabili di tipo riferimento, `a == b` restituisce `true` se le due variabili fanno riferimento allo stesso oggetto, altrimenti restituisce `false` (e il contrario vale per `!=`).

Per confrontare invece i valori rappresentati da due oggetti si utilizza il metodo `equals`: `a.equals(b)` restituisce `true` se gli oggetti a cui fanno riferimento `a` e `b` rappresentano valori uguali (secondo un criterio che dipende dalla classe), altrimenti restituisce `false`.

2.2.1 Esempio

```
String u, v;

u = new String("pippo");
v = u;
u == v; // true: u e v fanno riferimento allo stesso oggetto
u.equals(v); // true: un oggetto è uguale a se stesso

u = new String("pippo");
v = new String("pippo");
u == v; // false: u e v fanno riferimento a oggetti diversi
        // (salvo ottimizzazione "string pool")
u.equals(v); // true: u e v rappresentano la stessa stringa
```

2.3 Operatori booleani

Oltre ai valori (`true` e `false`), il tipo `boolean` è caratterizzato dalle operazioni consentite, gli **operatori booleani**:

- 2 operatori binari

  `&&`: *and* o *congiunzione*

  `||`: *or* o *disgiunzione*

- 1 operatore unario

  `!`: *not* o *negazione*

L'operatore `!` ha la precedenza massima, seguito da `&&` e infine `||`.

2.3.1 Leggi di De Morgan

```
!(x && y) == !x || !y
!(x || y) == !x && !y
```

3 Istruzione `do...while`

```
do
    istruzione;
while (condizione);
```

- `condizione` è un'espressione booleana
- `istruzione` (*corpo* del ciclo) è un'istruzione singola o un blocco

Semantica operazionale:

1. Viene eseguita l'`istruzione`
2. Viene valutata la `condizione`
   - se è vera, ritorna al punto 1
   - se è falsa, l'esecuzione prosegue dall'istruzione successiva

Osservazioni:

- il corpo viene eseguito almeno una volta
- l'esecuzione termina quando la condizione è falsa