Operatori logici per le operazioni sui dati bit a bit

Il linguaggio Java, oltre agli operatori booleani *And* (&&), *Or* (||) e *Not* (!), possiede anche gli **operatori logici** che operano sui dati **bit a bit** anziché globalmente. Essi sono:

~	per il complemento a uno, nega cioè bit a bit il dato
&	per la congiunzione logica bit a bit (And)
|	per la disgiunzione logica bit a bit (Or)
^	per l'Or esclusivo bit a bit (Xor)
>>	per lo scorrimento a destra (*shift right*) del primo operando di tanti bit quanti indicati dal secondo
<<	per lo scorrimento a sinistra (*shift left*).

Per esempio, consideriamo due variabili numeriche di tipo byte:

byte x, y;

Entrambe possono contenere un valore intero rappresentabile con 8 bit.
Supponiamo che alla variabile *x* venga assegnato il valore 35 (00100011 in binario) e alla variabile *y* il valore 9 (00001001 in binario). La seguente tabella mostra qual è il risultato dell'applicazione degli operatori logici bit a bit, usando le variabili *x* e *y*:

Espressione	Valore in bit dopo la valutazione	Valore decimale dopo la valutazione
~ x	11011100	220
x & y	00000001	1
x | y	00101011	43
x ^ y	00101010	42
x >> 3	00000100	4
y << 2	00100100	36

Il seguente programma mostra l'utilizzo di tutti gli operatori bit a bit.

```
class OperatoriBit
{
  public static void main(String[] args) {
    byte x = 35;
    byte y = 9;
    int z;

    stampa("x", x);
    stampa("y", y);

    z = ~x;
    stampa("~x", z);
```

```
    z = x & y;
    stampa("x & y", z);

    z = x | y;
    stampa("x | y", z);

    z = x ^ y;
    stampa("x ^ y", z);

    z = x >> 3;
    stampa("x >> 3", z);

    z = y << 2;
    stampa("y << 2", z);
  }

  public static void stampa(String espressione, int valore) {
    String valoreInBit;
    int valoreDecimale;

    valoreDecimale = (valore & 0x000000FF);
    valoreInBit    = Integer.toBinaryString(valoreDecimale);

    System.out.println(espressione
        + " = " + valoreInBit
        + " = " + valoreDecimale);
  }
}
```

Si noti che il risultato delle espressioni deve essere memorizzato in una variabile *z* di tipo *int* composta da 32 bit. All'interno del metodo *stampa*, per estrarre solo gli 8 bit da visualizzare, si applica l'operatore *&* alla variabile di tipo *int*, usando come secondo operando il valore esadecimale 0x000000FF, che corrisponde ad una sequenza binaria di 24 zero seguita da 8 uno.

Per visualizzare un valore numerico in formato binario è stato utilizzato il metodo statico **Integer.toBinaryString**, passando il numero come parametro.