Azzolini Riccardo 2018-12-04

Overloading e overriding

1 Overloading

L'**overloading** è la possibilità di avere più metodi o costruttori con lo *stesso nome* ma con *segnature diverse*.

Esso viene risolto *in fase di compilazione* (*early binding*):

1. Tra i metodi con lo stesso nome disponibili per il *tipo del riferimento* usato nell'invocazione, vengono scelte le segnature candidate, cioè quelle
   - compatibili con gli argomenti specificati nella chiamata (stesso numero e tipi assegnabili)
   - accessibili al codice chiamante
2. Viene scelta la segnatura più specifica tra le candidate, cioè quella che richiede meno conversioni implicite (promozioni).

Se non ci sono segnature compatibili, oppure in caso di ambiguità (se non c'è una singola segnatura più specifica di tutte le altre), si verifica un errore in fase di compilazione.

1.1 Esempi

```
class A {
    int m(byte b) {...}
    int m(long l) {...}
    int m(double d) {...}
    ...
}

A r;
...
r.m(2);
```

In questo caso, il compilatore sceglie la segnatura `int m(long l)`:

- `int m(byte b)` non è candidata perché `int` non si può convertire implicitamente a `byte`
- le candidate sono `int m(long l)` e `int m(double d)`

- tra le candidate viene preferita `int m(long l)` perché da `int` a `long` servono meno conversioni implicite che da `int` a `double`

```
z(double x, int y) {...}
z(int x, double y) {...}

z(1, 2)
```

Questo caso è ambiguo: entrambe le segnature di `z` sono candidate e richiedono una conversione implicita da `int` a `double`.

2 Overriding

L'**overriding** è la riscrittura, in una sottoclasse, di un metodo di una superclasse con la *stessa segnatura*.

Esso viene risolto *in fase di esecuzione* (*late binding*): viene cercato un metodo con la segnatura selezionata dal compilatore, risalendo la gerarchia a partire dalla *classe dell'oggetto* a cui è rivolta l'invocazione.