Dynamic dispatch e invocazione di metodi in Java (caso ereditarietà singola)

Distinguiamo:

tipo apparente determinato staticamente in base ai tipi inseriti nelle dichiarazioni, e

tipo effettivo presente a runtime.

Vale che tipo effettivo <: apparente.

Metodi virtuali

Il compilatore:

- verifica l'appartenenza alla classe corrispondente al tipo apparente, e che
- i parametri attuali siano compatibili con i tipi dei parametri formali;
- determina l'offset del metodo nella tabella dei metodi del tipo apparente.

Se il tipo apparente non ha il metodo ma si è certi che il tipo effettivo lo abbia (per esempio dentro una guardia con `instanceof`), è necessario un downcast esplicito.

A runtime, la chiamata avviene secondo dynamic dispatch:

- il descrittore di dato del metodo permette di accedere alla dispatch table dell'oggetto;
- questa è una copia della tabella della superclasse, con i metodi aggiuntivi inseriti in coda e quelli sovrascritti mantenuti nella stessa posizione; perciò l'offset determinato staticamente in base al tipo apparente è valido anche per il tipo effettivo;
- viene eseguito il codice a cui punta l'entrata della tabella selezionata tramite l'offset, passando come parametro implicito l'oggetto `this` (TODO regola inferenza, dovrebbe avere tipo effettivo invece di statico).

Il risultato è che la chiamata avviene in tempo costante. L'overhead di trovare la tabella dei metodi viene pagato solo la prima volta, poi il risultato viene memoizzato. Nel bytecode, corrisponde all'istruzione `invokevirtual #`*offset*.

Metodi statici

Implementate come procedure a top-level; sostanzialmente non sono metodi. Istruzione `invokestatic` nel bytecode.

Variabili d'istanza

I metodi hanno accesso alle variabili d'istanza tramite un parametro implicito `this` (riferimento all'oggetto su cui è stato invocato) che il compilatore aggiunge automaticamente ad ogni chiamata.