Programmazione ad'Oggetti

Leonardo Mengozzi

Contents

1 Approccio object-oriented

2 Variabili, Oggetti, Classi, Campi e Metodi

2.1 (Almost) Everything is an object

Le variabili, contenitori con nomi, ora non denotano solo valori numerici (come in C), ma anche veri e propri oggetti irriducibili.

Non ci sono meccanismi per controllo diretto memoria. Le variabili sono nomi "locali" con riferimenti ad'*oggetti* e non maschere di indirizzi in memoria a cui accedere direttamente.

Le variabili posso essere di tipo *Java Types* quindi classi predefinite e autoimplementate oppure *tipi primitivi*.

2.1.1 Tipi Primitivi

Non conviene trattare tutto come oggetto. I tipi atomici del C si sono mantenuti definendo una dimensione fissa e rimuovendo gli unsigned. Si è introdotto boolean con **true/false**.

Questi tipi sono unici e fissi da linguaggio.

Tipi primitivi	Dimensione	Minimo	Massimo
boolean	–	–	–
char	16bits	Unicode 0	Unicode $2^{16}-1$
byte	8bits	-128	+127
short	16bits	-2^{15}	$-2^{15}-1$
int	32bits	-2^{31}	$+2^{31}-1$
long	64bits	-2^{63}	$-2^{63}-1$
float	32bits	IEEE754	IEEE754
double	64bits	IEEE754	IEEE754

Le librerie *BigDecimal, BigInteger* gestiscono numeri di dimensione/precisione arbitraria.

Nota: In Java l'uso della memoria per i valori true/false di boolean, e tante altre cose, non sono date a sapere al programmatore dato che si dovrebbe concentrare su altro.

2.1.2 Stack e Heap

Tutte le variabili sono memorizzate nello **stack**. Le variabili di tipo primitivo sono affiancate dal loro valore mentre le variabili tipo classe sono affiancate dall'identità del loro oggetto. Gli oggetti sono memorizzati nell'heap.

Uno stesso oggetto può essere puntato da variabili che si riferiscono alla stessa identità.

2.2 Oggetti

1. Creazione

```
<Tipo | var> <nome> = <new Tipo([Tipo1 par1, ...])| altraVariabile |
 null>;
```

altraVariabile deve essere della stessa classe della variabile che sto definendo.

Nota: solo quando si scrive *new* (Keyword di linguaggio) si crea un oggetto dalla classe indicata.

`var`[1] fa infierire[2] il tipo della variabile locale per allegerire il codice. Se manca l'espressione non va, esempio `var i;`.

2.3 Classi

Nomi classi sempre con la maiuscola.

Definisco le configurazioni.

Le classi sono tipi di dato in un linguaggio a oggetti tutto è un oggetto fino a un certo punto.

2.3.1 Classe Object

Definizione, Creazione ed'Inizializzazione:

1. `<Tipo> <nome>;` sola definizione.
2. `<Tipo> <nome> = valore;` definizone ed'inizializzazione.

2.3.2 Precisazioni slasse String

Diversi modi per inizializzare un oggetto stringa:

1. `... = new String();` inizializza con una stringa (sequenza di caratteri) vuota.
2. `... = new String("...");` inizializza con la stringa passata come parametro.
3. `... = "..."` inizializza direttamente con una stringa (come in C), caso unico.

[1] Local variable type inference

[2] Far dedurre al compilatore il tipo della variabile locale dall'espressione assegnata.