Lab. Programmazione (CdL Informatica) & Informatica (CdL Matematica) a.a. 2022-23

Monica Nesi

Università degli Studi dell'Aquila

19 Ottobre 2022

Costrutti di controllo in Java

Sequenzializzazione:
per indicare che le istruzioni (o comandi) I_1 ed I_2 sono in sequenza, ovvero I_1 deve essere eseguita prima di I_2 (nel diagramma di flusso ciò è rappresentato mettendo il blocco dell'azione I_1 prima del blocco dell'azione I_2), basta scrivere (la codifica in Java di) I_1 prima di (quella di) I_2.

```
int x = 4, y = 5;
x = x+1;
y = y-1;
```

I due comandi di assegnamento I_1: `x = x+1;` ed I_2: `y = y-1;` sono in sequenza. Prima si esegue I_1 e poi I_2, e si ha `x=5`, `y=4`.

In questo caso modificare la sequenza, i.e. scrivere prima I_2 e poi I_1, risulta in uno stato della macchina equivalente.

Sequenzializzazione

In generale, date due qualsiasi istruzioni I_1, I_2, non è vero che I_1 seguita da I_2 sia *equivalente* a I_2 seguita da I_1.

Dato il frammento di codice

```
int x = 4, y = 5;
x = x+1;
y = x+y;
```

la sua esecuzione risulta in `x=5` ed `y=10`.

Scambiare la sequenza dei due assegnamenti

```
int x = 4, y = 5;
y = x+y;
x = x+1;
```

risulta in `x=5` ed `y=9`, ovvero uno stato diverso della memoria.

Sequenzializzazione (cont.)

Dato il frammento di codice

```
int x,y;
x = 3;
y = x+2;
```

la sua esecuzione risulta in x=3 ed y=5.

Se la sequenza dei due assegnamenti viene modificata:

```
int x,y;
y = x+2;
x = 3;
```

si ha un errore in fase di compilazione, in quanto nel primo assegnamento la variabile x non risulta inizializzata.

Costrutto condizionale in Java

Condizionale (*if-then-else*):
questo costrutto permette di eseguire un'istruzione I_1 se una data condizione logica E è vera e di eseguire un'istruzione I_2 se E è falsa.

In Java questo costrutto è il comando condizionale `if-else`, la cui sintassi può essere data come segue:

$$\text{if } (<BEspr>) \ <Istr> \ [\text{else } <Istr>]$$

dove $<BEspr>$ denota l'insieme delle espressioni booleane e $<Istr>$ denota l'insieme delle istruzioni (comandi) del linguaggio.

N.B. L'espressione booleana deve essere racchiusa tra parentesi tonde.

Le parentesi quadrate non fanno parte del linguaggio.
Sono *meta-simboli* che denotano che la parte racchiusa tra tali parentesi è *opzionale*.

Comando condizionale

Ciò significa che esistono due varianti del comando condizionale: il comando *con* il ramo `else` e quello *senza* il ramo `else` (di cui abbiamo già dato la semantica tramite i diagrammi di flusso).

Dato un generico comando `if-else`

$$\text{if } (E) \ C1 \ \text{else } \ C2$$

se l'espressione booleana E è vera, allora viene eseguito il comando $C1$, altrimenti viene eseguito il comando $C2$.

Dato un generico comando `if`

$$\text{if } (E) \ C1$$

se l'espressione booleana E è vera, allora viene eseguito il comando $C1$, altrimenti non viene eseguito alcun comando.

Comando condizionale (cont.)

I comandi nel *corpo* dei rami vero/falso del condizionale possono essere una *singola* istruzione oppure una *sequenza di istruzioni* in un *blocco*, ovvero racchiuse tra una coppia di parentesi graffe.

Se nel corpo dei rami vero/falso esistono più istruzioni *non* racchiuse in un blocco, l'interpretazione di Java è di considerare come corpo del ramo *solo la prima istruzione* della sequenza.

Dato il comando

$$\begin{array}{c} \text{if } (E) \\ C1 \\ C2 \end{array}$$

solo $C1$ viene considerato come corpo del ramo vero dell'`if`.
Se E è vera, viene eseguito $C1$ e poi si esegue $C2$.
Se E è falsa, si va in sequenza e si esegue $C2$.
Quindi si esegue sempre $C2$, indipendentemente dal valore di E.

Comando condizionale (cont.)

Se si vuole eseguire $C1$ e $C2$ in sequenza solo se E è vera, occorre introdurre un blocco:

```
if (E) {
  C1
  C2
}
```

Dato

```
if (E)
  C1
else
  C2
C3
```

il corpo del ramo falso è dato solo da $C2$.
Il comando $C3$ viene eseguito sempre dopo l'`if-else`.
Se $C3$ deve essere eseguito solo dopo $C2$ nel ramo falso, allora occorre introdurre un blocco dopo `else`.

Variabili locali in un blocco

Una volta aperto un blocco, è possibile dichiarare identificatori *locali* a tale blocco.

Ciò significa che tali identificatori sono visibili, e quindi riferibili, solo in tale blocco.

Eventuali identificatori dichiarati in blocchi esterni sono visibili anche nei blocchi interni.

N.B. In Java non è possibile ridichiarare in un blocco interno un identificatore già dichiarato in un blocco esterno (i.e. un blocco che lo contiene).

Se i blocchi *non* sono uno dentro l'altro (come i corpi dei rami vero/falso dell'`if-else`), allora è possibile dichiarare lo stesso identificatore dichiarato in un blocco distinto.

Comandi condizionali annidati

I comandi condizionali possono contenere nei loro rami vero/falso altri comandi condizionali.

Si parla di `if-else` *annidati*.

Vediamo alcuni esempi in generale.

$$\begin{array}{l} \text{if } (E1) \\ \quad \text{if } (E2) \\ \quad\quad C \end{array}$$

Questo è un comando `if`, il cui ramo vero contiene un comando `if`. C viene eseguito se $E1$ ed $E2$ sono entrambe vere. Altrimenti non si esegue alcun comando.

Comandi condizionali annidati (cont.)

if $(E1)$
  if $(E2)$
    $C1$
  else
    $C2$

Questo è un comando `if`, il cui ramo vero contiene un `if-else`.

Ma potrebbe essere interpretato come un comando `if-else`, il cui ramo vero contiene un comando `if`?

Problema:
quando si incontra un `else`, a quale `if` deve essere associato?

In Java un `else` viene associato all'`if` più vicino che non è stato ancora associato ad un `else`.

Comandi condizionali annidati (cont.)

Se invece si vuole interpretare tale comando come un `if-else`, il cui ramo vero contiene un comando `if`, allora occorre introdurre un blocco nel ramo vero dell'`if-else`.

```
if (E1) {
  if (E2)
    C1
}
else
  C2
```

$C1$ viene eseguito se $E1$ ed $E2$ sono entrambe vere.
$C2$ viene eseguito se $E1$ è falsa.
Se $E1$ è vera ed $E2$ è falsa, non si esegue alcun comando.

Comandi condizionali annidati (cont.)

```
if (E1)
  C1
else
  if (E2)
    C2
```

Comando `if-else`, il cui ramo falso contiene un `if`.

```
if (E1)
  C1
else
  if (E2)
    C2
  else
    C3
```

Comando `if-else`, il cui ramo falso contiene un `if-else`.

Comandi condizionali annidati (cont.)

```
if (E1)
  if (E2)
    C1
  else
    C2
else
  C3
```

Comando `if-else`, il cui ramo vero contiene un `if-else`.

Comandi condizionali annidati (cont.)

```
if (E1)
  if (E2)
    C1
  else
    C2
else
  if (E3)
    C3
  else
    C4
```

Comando `if-else`, i cui rami vero/falso contengono entrambi un `if-else`.

N.B. Non sono richiesti blocchi, i.e. le parentesi graffe.

Massimo di due interi

```
class Max2 {
 public static void main (String[] args) {
  int n = Integer.parseInt(args[0]);
  int m = Integer.parseInt(args[1]);

  if (n>m) {
    System.out.println(n);
  }
  else {
    System.out.println(m);
  }
 }
}
```

Codifica in Java del diagramma di flusso dato a suo tempo.

Massimo di due interi: variante

```
class Max2 {
 public static void main (String[] args) {
  int n = Integer.parseInt(args[0]);
  int m = Integer.parseInt(args[1]);
  int r;
  if (n>m) {
    r = n;
  }
  else {
    r = m;
  }
  System.out.println("Il massimo tra "+n+
  " ed "+m+" e' "+r+".");
 }
}
```

Massimo di due interi: espressione condizionata

```
class Max2 {
 public static void main (String[] args) {
  int n = Integer.parseInt(args[0]);
  int m = Integer.parseInt(args[1]);

  int r = n>m ? n : m;
  System.out.println("Il␣massimo␣tra␣"+n+
  "␣ed␣"+m+"␣e'␣"+r+".");
 }
}
```

È possibile eliminare la variabile `r` ed inserire l'espressione condizionata direttamente nella stringa argomento del `println`.

In modo simile è possibile calcolare il minimo di due interi ed il massimo/minimo di tre interi.