Università degli Studi di Palermo

Corso di Laurea
Ingegneria Gestionale

Lezione 10

Le Istruzioni di Controllo

Ing. Massimo Cossentino

Sistemi Informativi Aziendali

a.a. 2008/2009

Risolvere Problemi

- Qualsiasi problema di informatica può essere risolto eseguendo una serie di azioni in un ordine specifico.
- Algoritmi:
  - Azioni da eseguire
  - Ordine in cui queste azioni vengono eseguite (-> flusso di controllo del programma).
- Flusso di controllo del programma:
  - Specifica l'ordine in cui le azioni vanno eseguite all'interno di un programma
  - Il controllo del programma viene effettuato attraverso quelle che si chiamano Strutture di Controllo di un determinato linguaggio.

Le strutture di controllo

- Esecuzione sequenziale:
  - Normalmente le istruzioni vengono eseguite una dopo l'altra nell'ordine in cui sono scritte
- Trasferimento del controllo:
  - Specifica l'istruzione successiva da eseguire che non è esattamente la prossima
  - Può essere fatto attraverso una delle istruzioni di controllo messe a disposizione dal linguaggio

Le strutture di controllo

- Bohm e Jacopini dimostrarono che tutti i programmi potevano essere scritti attraverso tre strutture di controllo:
  - La struttura sequenziale
  - La struttura di selezione
  - La struttura iterativa

Le strutture sequenziali

- Le strutture sequenziali sono implicite in Java, in mancanza di indicazioni diverse le istruzioni vengono eseguite in maniera sequenziale.

Le strutture di selezione

- Java prevede tre tipi di strutture di selezione:
  - If -> istruzione di selezione singola
    - Una istruzione viene eseguita nel caso di condizione true, altrimenti viene saltata
  - If..else -> istruzione di selezione doppia
    - Permette di scegliere tra due azioni alternative, o tra gruppi di azioni
  - Switch -> istruzione di selezione multipla
    - Permette di scegliere tra varie azioni alternative o gruppi di azioni

Le strutture iterative

- Le strutture iterative:
  - Sono note anche come strutture di looping
  - Permettono di fare azioni ripetutamente fina a quando una condizione (chiamata di continuazione) rimane vera.
- Ci sono tre tipi di strutture iterative:
  - While, do..while e for
    - permettono di ripetere le azioni all'interno del loro corpo zero o tante volte
    - While e for non eseguono le azioni se la condizione iniziale di continuazione è falsa, do ..while esegue le azioni almeno la prima volta.

Riepilogo

- Java ha tre tipi di strutture di controllo:
  - le istruzioni sequenziali
  - Le istruzioni di selezione (tre tipi)
  - Le istruzioni di iterazione (tre tipi)
  - TUTTI I programmi sono composti da queste istruzioni di controllo secondo due tecniche:
    - Sovrapposizione delle strutture si controllo (ad un ingresso di un'azione corrisponde l'uscita di un'altra)
    - Nidificazione.

Istruzioni per il controllo di flusso	
Istruzione	**Descrizione**
if	Esegue o no un blocco di codice a seconda del valore restituito da una espressione booleana.
if-else	Esegue permette di selezionare tra due blocchi di codice quello da eseguire a seconda del valore restituito da una espressione booleana.
switch	Utile in tutti quei casi in cui sia necessario decidere tra opzioni multiple prese in base al controllo di una sola variabile.
for	Esegue ripetutamente un blocco di codice.
while	Esegue ripetutamente un blocco di codice controllando il valore di una espressione booleana.
do-while	Esegue ripetutamente un blocco di codice controllando il valore di una espressione booleana.

Università degli Studi di Palermo

Strutture di controllo: sintassi Java

If : istruzione di selezione singola

- Serve a decidere quale azione intraprendere tra diverse alternative.

Se il voto dello studente è maggiore o uguale a 60

Visualizza "Promosso"

```
if (votoStudente >=60 )
   System.out.println ("Promosso");
```

If..else : selezione doppia

- L'istruzione if esegue l'azione solo se la condizione è vera altrimenti esegue l'alternativa.

Se il voto dello studente è maggiore o uguale a 60

Visualizza "Promosso"

Altrimenti

Visualizza "Bocciato"

```
if (votoStudente >=60 )
      System.out.println ("Promosso");
else
          System.out.println ("Promosso");
```

Il problema dell'else pendente

- Il compilatore Java associa sempre un else con un if precedente
- E' sempre conveniente racchiudere tra parentesi graffe il contenuto di un if per indicare al compilatore la giusta sequenza di istruzioni.
- Esempio:

```
if (x>5)
 {
    if (y>5)
       System.out.println("x e y sono > 5);
}
else
    System.out.println("x è <= 5");
```

Switch

- L'istruzione switch è utile in tutti quei casi in cui sia necessario decidere tra opzioni multiple prese in base al controllo di una sola variabile.

```
switch (espressione)
{
      case espr_costante:
                                  istruzione1
                                  break        \
   \opzionale
      case espr_costante:
                                  istruzione2
                                  break        \
   \opzionale
……..
      default:
                                  istruzione4
}
```

Switch: esempio

```
int i = getUserSelection();
switch (i)
{
      case 1:
                  faiQualcosa();
                  break;
      case 2:
                  faiQualcosAltro();
                  break;
      default:
                  nonFareNulla();
}
```

While

- Una istruzione while permette la esecuzione ripetitiva di una istruzione utilizzando una espressione booleana per determinare se eseguire il blocco di istruzioni, eseguendolo quindi fino a che l'espressione booleana non restituisce il valore false.

```
while (espressione_booleana){
        istruzione
}
```

While

- Il codice di esempio utilizza due cicli while annidati:

```
int i=0;
int j;
while(i<10)
{
        j=10;
        while(j>0)
        {
            System.out.println("i="+i+" e j="+j);
            j--;
        }
        i++;
}
```

While esercizi

- Modificare l'esempio precedente per stampare 10 volte il testo 'ciao'

Do - while

- Una alternativa alla istruzione while è rappresentata dall'istruzione do-while a differenza della precedente, controlla il valore della espressione booleana alla fine del blocco di istruzioni.
- In questo caso quindi **il blocco di istruzioni verrà eseguito sicuramente almeno una volta.**

- La sintassi di do-while è la seguente:

```
do {
    istruzione;
} while (espressione_booleana);
```

Do-while esempio

```
int i=0;
do
{
  i++;
  System.out.println("i= "+i);
} while (i<=10);
```

NB: il programma verrà eseguito fino a i=11

Do-while esercizi

- Modificare l'esempio per stampare la sequenza:
  - 1, 4, 7, 10
  - 2, 4, 8, 16 (potenze di 2)

For

- Il ciclo `for` esegue una istruzione per un certo numero di volte (finchè il valore della variabile iteratore non raggiunge il valore massimo specificato)

```
for(inizializzazione; condizione; incremento)
{
        istruzione
}
```

For: esempio

```
for (int i=0 ; i<10 ; i++)
{
    System.out.println("i= "+i);
}
```

modificare il ciclo per:

- Contare da 10 a 1 in modo decrescente
- Mostrare tutti i numeri pari <=20
- Mostrare tutti i numeri dispari <=20

L'istruzione return

- Questa istruzione può essere utilizzata per terminare l'esecuzione del metodo corrente tornando il controllo al metodo chiamante.
- Return può essere utilizzata in due forme:

```
return valore;

return;
```

- La prima forma viene utilizzata per consentire ad un metodo di ritornare valori al metodo chiamante e pertanto deve ritornare un valore compatibile con quello dichiarato nella definizione del metodo. La seconda può essere utilizzata per interrompere l'esecuzione di un metodo qualora il metodo ritorni un tipo void.

Università degli Studi di Palermo

Operatori di confronto

Confrontare valori: Operatori relazionali

Operatore Java	Notazione matematica	Descrizione
>	>	Maggiore
>=	≥	Maggiore di o uguale a
<	<	Minore
<=	≤	Minore di o uguale a
==	=	Uguale
!=	≠	Diverso

Confrontare valori: Operatori relazionali

Operatore Java	Notazione matematica	Descrizione
>	>	Maggiore
>=	≥	Maggiore di o uguale a
<	<	Minore
<=	≤	Minore di o uguale a
==	=	Uguale
!=	≠	Diverso

- L'operatore == verifica l'uguaglianza.

```
a = 5; // assegna 5 ad a
if (a == 5) . . . // verifica se a è uguale a 5
```

Confrontare numeri in virgola mobile

- Confrontando numeri in virgola mobile, non si fanno verifiche di uguaglianza (= =), ma si controlla se i valori sono *sufficientemente prossimi*.
- Per confrontare numeri double, di solito si usa un valore di ε uguale a 10^{-14}.
- Analogamente, potete verificare se due numeri sono prossimi l'uno all'altro controllando se la loro differenza è prossima a 0.

$$|x - y| \leq \varepsilon$$

```
final double EPSILON = 1E-14;
if (Math.abs(x - y) <= EPSILON)
   // x è quasi uguale a y
```

Confrontare stringhe

- Non utilizzare == per le stringhe

```
if (string1 == string2) // inutile
```

- Utilizzare il metodo `equals`:

```
if (input.equals("Y"))
```

- == verifica se le due variabili stringa si riferiscono al medesimo oggetto stringa (in pratica se le variabili reference contengono lo stesso valore).
  `equals` verifica che il contenuto sia uguale.

- Per ignorare le differenze tra maiuscolo e minuscolo usate il metodo `equalsIgnoreCase`

```
if (input.equalsIgnoreCase("Y"))
```

Confronto tra stringhe: esempio

```
String s1="prova";
String s2="sbagliato";
String s3="prova";
if (s1.equals(s2))
{
    System.out.println("s1 è uguale a s2");
}
else
     System.out.println("s1 non è uguale a s2");
if (s1.equals(s3))
{
     System.out.println("s1 è uguale a s3");
}
else
         System.out.println("s1 non è uguale a s3");
```

Confrontare stringhe

- `s.compareTo(t) < 0` significa che `s` precede la `t` in ordine alfabetico.
- "car" precede "cargo"
- Tutte le lettere maiuscole precedono quelle minuscole

  “B” precede “a”

Tabella ASCII

CARATTERI STAMPABILI					
DEC	CARATTERE	DEC	CARATTERE	DEC	CARATTERE
32	<SPACE>	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39		71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	<DEL>

Confrontare oggetti

- L'operatore == verifica se due riferimenti a oggetto sono identici. Per confrontare, invece, i contenuti di oggetti, si deve usare il metodo `equals`.

```
Rectangle box1 = new Rectangle(5, 10, 20, 30);
Rectangle box2 = box1;
Rectangle box3 = new Rectangle(5, 10, 20, 30);
```

- `box1 != box3`,
  ma `box1.equals(box3)`
- `box1 == box2`
- `equals` deve essere definito nella classe

L'istruzione import

L'istruzione import ha come unico effetto quello di identificare univocamente una classe e quindi di consentire al compilatore di risolvere nomi di classe senza ricorrere ogni volta a nomi qualificati. Grazie all'istruzione

```
import app.stack.Stack;
```

una applicazione sarà in grado di risolvere il nome di Stack ogni volta che sia necessario semplicemente utilizzando il nome di classe Stack.

Capita spesso di dover però utilizzare un gran numero di classi appartenenti ad un unico package. Per questi casi l'istruzione import supporta l'uso di un carattere fantasma :

```
import app.stack.*;
```

che risolve il nome di tutte le classi pubbliche di un package (app.stack nell'esempio).

Package

- I package sono meccanismi per raggruppare definizioni di classe in librerie, similmente ad altri linguaggi di programmazione. Il meccanismo è provvisto di una struttura gerarchica per l'assegnamento di nomi alle classi in modo da evitare eventuali collisioni in caso in cui alcuni programmatori usino lo stesso nome per differenti definizioni di classe.

- La API Java è composta di molti package in cui sono memorizzate le classi costituenti le librerie del linguaggio che possono essere usate per comporre rapidamente programmi estremamente complessi

Come studiare

- Memorizzare la sintassi delle istruzioni di controllo
- Provare nel riquadro del codice (Code Pad) di BlueJ gli esempi di strutture di controllo.
  - Provare a modificare il codice ed i parametri e prevedere il funzionamento
  - Ripetere più volte
    - È necessario non dichiarare di nuovo le variabili già definite (es. `int i=0;` la seconda volta che si esegue il codice dovrà essere: `i=0;`)