Università di Torino – Facoltà di Scienze MFN
Corso di Studi in Informatica

Programmazione I - corso B
a.a. 2009-10

prof. Viviana Bono

Blocco 5 – Istruzioni composte.

Terminologia

valutare un'espressione (aritmetica, o booleana, ecc.)
vuol dire
calcolare il valore di tale espressione;
Ad esempio:
valutando *3 > 5 && true* si ottiene *false*;
valutando *3 + 5* si ottiene *8*; ecc.

NB Da non confondere:
le espressioni si valutano,
le istruzioni si eseguono.

ATTENZIONE: NON CONFONDERE
IL TIPO DI UN CAMPO O DI UNA VARIABILE
CON IL SUO NOME!

Esempio 1:

```
static Scanner tastiera;
```

`Scanner` è il TIPO del campo, `tastiera` è il NOME del campo.

Esempio 2:

```
public static void main(String[] args) {
   Scanner tastiera = new Scanner(System.in);
   ...
```

`Scanner` è il TIPO della variabile,
`tastiera` è il NOME della variabile

ATTENZIONE: NON CONFONDERE
LA DICHIARAZIONE DI UN CAMPO O VARIABILE
CON IL SUO USO!

IL TIPO COMPARE SOLO NELLA DICHIARAZIONE!

Esempio:

`int i = 120;` dichiarazione con inizializzazione;

...

`i = i + 14;` assegnazione

...

`int i = i + 5;` è la dichiarazione di una nuova variabile i; se si trova entro la portata (scope) della i precedente, *è un errore di compilazione*

Istruzioni Composte

Blocco

{

istruzione-o-dichiarazione-di-variabile$_1$

istruzione-o-dichiarazione-di-variabile$_2$

. . .

istruzione-o-dichiarazione-di-variabile$_n$

}

le istruzioni o dichiarazioni di un blocco vengono eseguite in sequenza.

Nota Ognuna delle istruzioni, se è semplice, conterrà un punto-e-virgola finale

Attenzione!

Il blocco NON richiede il punto-e-virgola dopo la chiusa graffa: il punto-e-virgola non è un separatore, ma un **terminatore di istruzione semplice**.

La scrittura:

```
{
    istruzione-o-dichiarazione-di-variabile1
    istruzione-o-dichiarazione-di-variabile2
    . . .
};
```

è un blocco seguito da un'istruzione nulla.

Istruzioni di selezione

```
if(espressione-booleana)
    istruzione
```

oppure

```
if(espressione-booleana)
    istruzione1
else
    istruzione2
```

istruzione, *istruzione*$_1$ e *istruzione*$_2$ possono essere blocchi

Ad esempio…

```
if(espressione-booleana) {
   istruzione11
      ...
   istruzione1m
}
else {
   istruzione21
      ..... ... ...
   istruzione2n
}
```


Istruzione virtuale if ... else if ... else if ... else ...

```
if(espress-booleana1)
   istruzione1
else if(espress-booleana2)
   istruzione2
else if(espress-booleana3)
   istruzione3
...
else
   istruzione n
```

Istruzione virtuale if... else if... else if... else... (cont.)

Vengono valutate in sequenza le condizioni

espress-booleana$_1$

espress-booleana$_2$

...

finché si trova una condizione *espress-booleana*$_i$ che vale *true* : si esegue la corrispondente *istruzione*$_i$ e poi si "saltano" tutte le istruzioni corrispondenti alle condizioni successive.

Se nessuna delle *espress-booleana*$_i$ vale *true,* si esegue l'istruzione corrispondente alla condizione "pigliatutto" *else* (che esegue *istruzione*$_n$).

Esempio

```
out.print("digita la temperatura di oggi in gradi centigradi (intero): ");
  int t = input.nextInt();
  if(t < 5) {
    out.print("brr... ce l'hai il cappotto? ");
    input = new Scanner(System.in);
    String risposta = input.nextLine();
    if(risposta.equals("no")) out.println("ti prenderai un accidente!");
  }
  else if(t < 15) out.println("freschetto!");
  else if(t < 28) out.println("bella temperatura!");
  else {
    out.println("hai il condizionatore?");
    //input = new Scanner(System.in);
    String risposta = input.nextLine();
    if(risposta.equals("si"))
      out.println("attento a non prnderti un accidente!");
    else out.println("usa un ventaglio!");
  }
```

Portata (*scope*) di una dichiarazione di variabile (*argomento avanzato*)

La portata (in inglese *scope*) di una dichiarazione è la parte di programma in cui tale dichiarazione ha effetto.

Una dichiarazione di variabile in un metodo ha effetto (cioè la variabile esiste) da punto della dichiarazione fino alla fine del blocco di istruzioni (cioè fino alla }) in cui è situata.

Una dichiarazione di variabile *maschera* eventuali campi statici con lo stesso nome.

Una dichiarazione di variabile entro la portata di una dichiarazione di una variabile con lo stesso nome è un *errore di compilazione*.

Esempio (difficile!)

```
public class ProvaScope {
  static String s = "stringa globale";

  public static void main(String[] args) {
    if(args.length == 0) {
      System.out.println(s);  //    stringa globale
      String s = "stringa nell' if-then";
      System.out.println(s); //     stringa nell' if-then
    }
    System.out.println(s); //       stringa globale
  }
}
```

Altro esempio (difficile!)

```
public static void main(String[] args) {
   String s = "Stringa nel main";
   if (args.length == 0) {
     System.out.println(s);
     String s = "Stringa nell'if-then"
     // ERRORE DI COMPILAZIONE!
     ...
   }
}
```


Che cos'è un ciclo

- Un ciclo è il ripetere una serie di azioni fino a che non si è arrivati al risultato voluto
- Un insieme di azioni ripetute permette di arrivare incrementalmente al risultato
- Esempi:
  - fare un esercizio a un attrezzo in palestra;
  - ordinare in ordine alfabetico dei nomi
- Il risultato voluto è esprimibile tramite una condizione, che fa terminare l'iterazione della serie di azioni:
  - abbiamo raggiunto il numero voluto di ripetizioni dell'esercizio;
  - tutti i nomi sono stati ordinati (o anche: non ci sono piú nomi fuori posto: il come esprimere una condizione è molto importante)

Istruzioni iterative (cicli): while

`while` **`(`** *espressione-booleana* **`)`**

istruzione

Nota *istruzione* può essere un blocco; essa viene chiamata *corpo* del while.

Esecuzione:

1. si calcola il valore della *espressione-booleana*:
   - false*:* si salta all'istruzione successiva al ciclo;
   - true: si esegue il corpo, poi si torna al punto 1.

Attenzione!

Istruzione while: tutto il costrutto

`while` **`(`** ***Test*** **`)`**

Corpo

è UNA istruzione (composta):
più precisamente, è una istruzione while, la cui esecuzione consiste nella ripetizione (controllata dal ***Test*** nel modo indicato nelle pag. precedenti) delle istruzioni costituenti il ***Corpo***.

Ciò che viene ripetuto è pertanto il ***Corpo*** del while, non l'istruzione while!

Analoga terminologia si usa per le altre *istruzioni iterative*: *for* e *do-while*.

Note sul while

Se la prima volta il calcolo dell'espressione-test produce il valore *false*, il corpo del while non viene eseguito nemmeno una volta.

Il calcolo dell'espressione-test viene effettuato solamente all'inizio di una possibile esecuzione del corpo;

se durante l'esecuzione del corpo l'espressione-test diventa true, non per questo si esce immediatamente dal ciclo; soltanto alla fine dell'esecuzione del corpo si andrà a valutare di nuovo l'espressione booleana.

Un errore di esecuzione

Il seguente pezzo di programma è sintatticamente corretto (quindi non genera errori di compilazione)

```
int x = 20;
while(x < 100);
{x = x + 6;}
```

Ma l'istruzione while non termina, poiché il suo corpo NON è il blocco `{x = x + 6;}`, bensì l'istruzione vuota costituita dal punto-e-virgola, la quale ovviamente non modifica x e quindi non renderà mai true la condizione.

Il programma è logicamente errato!

Istruzioni iterative: for

```
for(iniz; condizione; incr)
  istruzione
```

Il ciclo for equivale ad un ciclo while:

```
iniz;
while(condizione) {
  istruzione;
  incr;
}
```


In un'istruzione for…

l'inizializzazione *iniz* può essere una dichiarazione con inizializzazione, ad esempio:

```
for(int i = 0; ... ; ...)
```

incr può essere qualunque istruzione semplice (senza il punto-e-virgola finale), ma un buono stile richiede che sia una istruzione di incremento o decremento;

ad esempio:

```
for(int i = 0; i < 10; i++)  out.println(i); // i++ equiv. i=i+1
for(int i = 10; i > 0; i--) out.println(i);
for(int i = 0; i < 10; i = i + 3)  out.println(i);
for(int i = 0; i < 10; i += 3)  out.println(i);
```

ecc.

Istruzioni iterative: do-while

```
do
  istruzione
while(espressione-booleana);
```

Nota: *istruzione* può essere un blocco; essa viene chiamata *corpo* del do.

NOTARE che, AL CONTRARIO che nel ciclo while, il punto-e-virgola finale non solo non è sbagliato, ma è obbligatorio

Esecuzione:

1. si esegue il corpo;
2. si valuta l' *espressione-booleana*:
   - false: si esce dal ciclo;
   - true: si torna al punto 1.

Attenzione!

Il corpo del **do** viene eseguito **almeno una volta**.

NB

La sintassi java dei cicli, ereditata dal C, è un po' infelice. Soprattutto in un programma complesso, attenzione a non confondere:

il `while` alla fine di un'istruzione **do-while**

con il

il `while` all'inizio di un'istruzione **while**