Istruzioni di controllo

Walter Didimo

Limite delle istruzioni viste

L’insieme delle istruzioni che abbiamo visto fino ad ora consiste per lo più di:

- dichiarazioni e assegnazioni di variabili
- invocazione di metodi
- restituzione di valori (return)

Con queste istruzioni non sono molte le cose che possiamo fare

Istruzioni di controllo

In questa lezione introduciamo le istruzioni di controllo; esse consentono di:

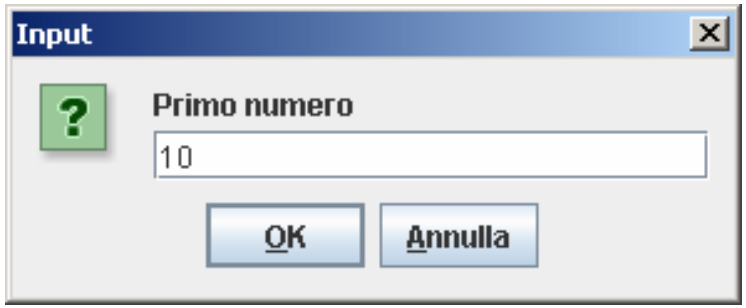
- confrontare dei dati e stabilire comportamenti diversi in funzione dell'esito del confronto (istruzioni condizionali)
- ripetere una stessa sequenza di istruzioni più volte, fintanto che certe condizioni permangono (istruzioni iterative)

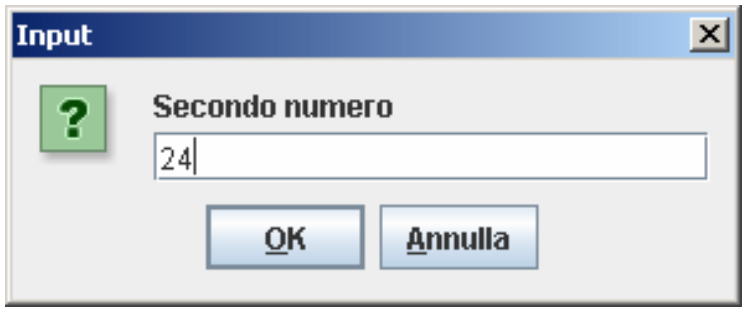
Istruzioni condizionali

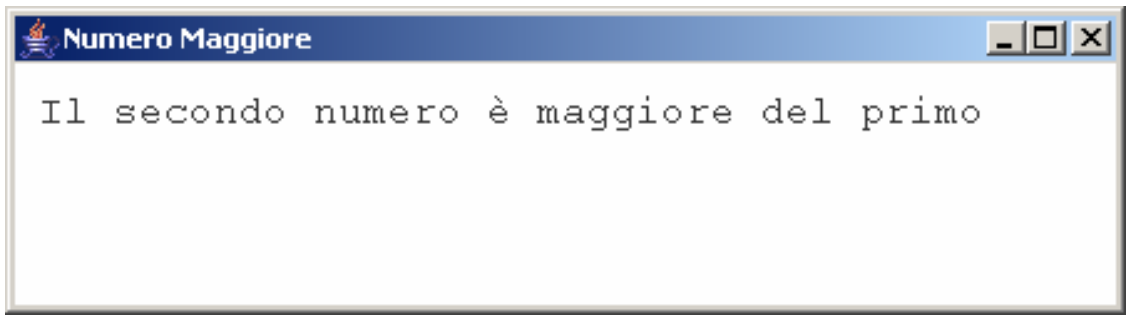
Vogliamo scrivere un semplice programma che chiede all'utente di inserire due numeri interi e che dice all'utente quale dei due è maggiore

Ipotizziamo per il momento che i due numeri non siano uguali

Esecuzione del programma







Il programma

```
class NumeroMaggiore{
   public static void main (String[] args){
       /* acquisisce i numeri */
       InputWindow in = new InputWindow ();
       int primo = in.readInt("Primo numero");
       int secondo = in.readInt("Secondo numero");

       /* confronta i numeri e stampa messaggio */
       OutputWindow out = new OutputWindow ("Numero Maggiore");
       if (primo > secondo)
         out.writeln ("Il primo numero è maggiore del secondo");
       else
         out.writeln ("Il secondo numero è maggiore del primo");
   }
}
```

L’istruzione if-else

Per confrontare due valori, e stabilire cosa fare in funzione dell’esito del confronto, si può usare l’istruzione condizionale if-else

if (condizione)

istruzione da eseguire se *condizione* è *true*

else

istruzione da eseguire se *condizione* è *false*

parte else

parte if

Regole dell'if-else

if (condizione)
parte if
else
parte else

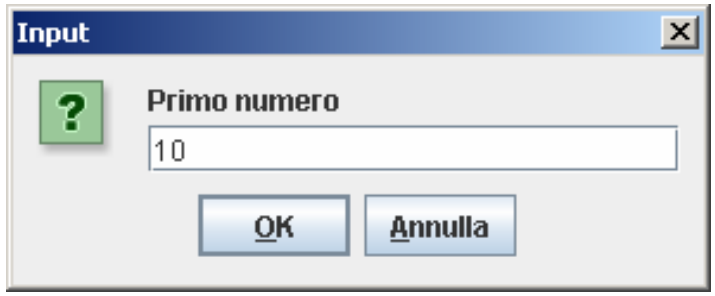
- *condizione* deve essere una espressione di tipo *boolean*, cioè un predicato
- *parte if* viene eseguita se condizione è *true*
- *parte else* viene eseguita se condizione è *false*

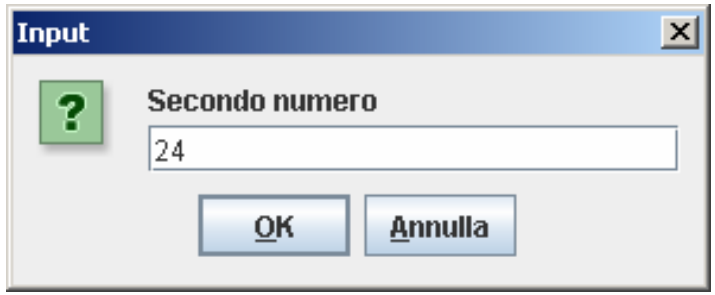
Istruzioni multiple nell'if-else

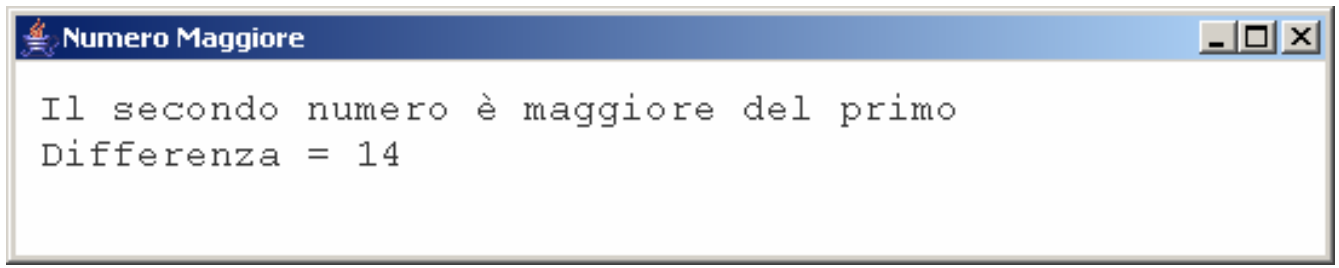
E' possibile eseguire più istruzioni nella *parte if* o nella *parte else* di una istruzione *if-else*?

Supponiamo ad esempio di voler modificare il programma di prima in modo che, oltre a dire quale è il numero più grande, visualizzi anche la differenza tra il più grande ed il più piccolo

Esecuzione del nuovo programma



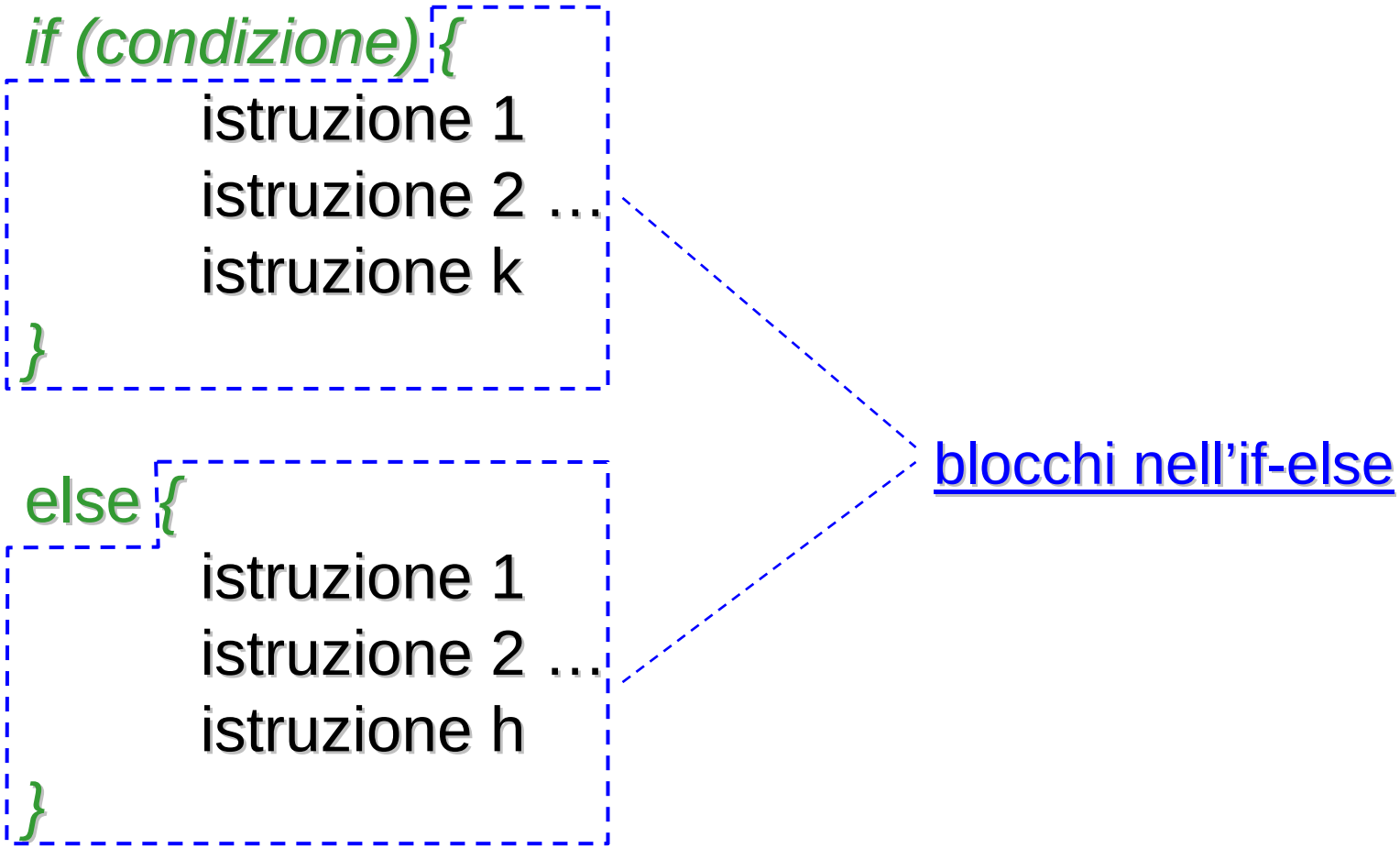




Il nuovo programma

```
class NumeroMaggiore{
   public static void main (String[] args){
       /* acquisisce i numeri */
       InputWindow in = new InputWindow ();
       int primo = in.readInt("Primo numero");
       int secondo = in.readInt("Secondo numero");

       /* confronta i numeri e stampa messaggio */
       OutputWindow out = new OutputWindow ("Numero Maggiore");
       if (primo > secondo){
         out.writeln ("Il primo numero è maggiore del secondo");
         out.writeln ("Differenza = " + (primo-secondo));
       }
       else{
         out.writeln ("Il secondo numero è maggiore del primo");
         out.writeln ("Differenza = " + (secondo-primo));
       }
   }
}
```

Blocchi di codice nell'if-else



Blocchi di istruzioni

In qualunque punto di un programma Java è possibile definire un blocco di istruzioni

- un *blocco di istruzioni* è sempre racchiuso da due parentesi graffe *{ }*
- è possibile annidare blocchi in altri blocchi
- eventuali variabili dichiarate in un blocco di istruzioni sono visibili solo nel blocco in cui sono dichiarate

Le regole sopra citate valgono anche per i blocchi nell'*if-else*

Codici errati

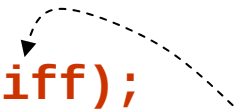
Ecco un esempio di codice errato

```
class NumeroMaggiore{
   public static void main (String[] args){
       ……
       if (primo > secondo)
         out.writeln (“Il primo numero è maggiore del secondo”);
         out.writeln (“Differenza = ” + (primo-secondo));
       else
         out.writeln (“Il secondo numero è maggiore del primo”);
         out.writeln (“Differenza = ” + (secondo-primo));
   }
}
```

Codici errati

Ecco un altro esempio di codice errato

```
class NumeroMaggiore{
   public static void main (String[] args){
       ……
       if (primo > secondo){
         out.writeln (“Il primo numero è maggiore del secondo”);
         int diff = primo – secondo;
       }
       else{
         out.writeln (“Il secondo numero è maggiore del primo”);
         int diff = secondo – primo;
       }
       out.writeln (“Differenza = ” + diff);
   }
}
```



Variabile non visibile
in questo punto

Codici corretto

Ecco invece un codice corretto alternativo

```
class NumeroMaggiore{
   public static void main (String[] args){
       ……
       int diff;
       if (primo > secondo){
         out.writeln (“Il primo numero è maggiore del secondo”);
         diff = primo – secondo;
       }
       else{
         out.writeln (“Il secondo numero è maggiore del primo”);
         diff = secondo – primo;
       }
       out.writeln (“Differenza = ” + diff);
   }
}
```

Condizioni multiple

Nel programma visto, avevamo escluso che i due numeri inseriti potessero essere uguali

Consideriamo ora anche questa possibilità; se i due numeri sono uguali, vogliamo visualizzare un esplicito messaggio

Vediamo come è possibile variare il codice del programma precedente (mostreremo solo la parte di interesse)

if-else annidati

```
class NumeroMaggiore{
   public static void main (String[] args){
      ……
      if (primo > secondo)
        out.writeln (“Il primo numero è maggiore del secondo”);
      else
        if (secondo > primo)
          out.writeln (“Il secondo numero è maggiore del primo”);
        else
          out.writeln (“I due numeri sono uguali”);
   }
}
```

L’istruzione nella *parte else* può essere a sua volta una nuova istruzione *if-else*

Cascate di if-else

```
if (condizione 1)                    // if 1
        parte if 1
else                                 // else 1
        if (condizione 2)            // if 2
                parte if 2
        else                                        parte else 1
                                     // else 2
                parte else 2
```

Istruzioni *if-else* annidate si chiamano anche cascate di if-else; il livello di annidamento può essere qualsiasi – le *parti if* e le *parti else* possono essere blocchi di istruzioni

Indentazione delle cascate di if-else

Ecco un altro modo, compatto e molto diffuso, di indentare le cascate di *if-else*

```
if (condizione 1)                    // if 1
        parte if 1
else if (condizione 2)               // else 1, if 2
        parte if 2
else                                 // else 2
        parte else 2
```

Indentazione delle cascate di if-else

Più in generale

```
if (condizione 1)                    // if 1
        parte if 1
else if (condizione 2)               // else 1, if 2
        parte if 2
else if (condizione 3)               // else 2, if 3
        parte if 3
….
else if (condizione k)               // else k-1, if k
        parte if k
else                                 // else k
        parte else k
```

L’istruzione if

In Java esiste anche l’istruzione condizionale if

Nell’istruzione *if* manca l’*else*

if (condizione)
parte if

- *condizione* è ancora un predicato
- *parte if* viene eseguita se e solo se *condizione* è *true*

Uso dell'istruzione if

Ad esempio, il seguente programma fa inserire due numeri all'utente e lo informa se essi sono uguali (non dice niente se sono diversi)

```
class NumeroMaggiore{
   public static void main (String[] args){
       /* acquisisce i numeri */
       InputWindow in = new InputWindow ();
       int primo = in.readInt("Primo numero");
       int secondo = in.readInt("Secondo numero");

       /* confronta i numeri e stampa messaggio */
       OutputWindow out = new OutputWindow ("Numero Maggiore");
       if (primo == secondo)
         out.writeln ("Numeri uguali");
   }
}
```

Esercizio sull'uso di if-else

Vogliamo scrivere un programma che chiede all'utente di inserire tre numeri e che visualizza il maggiore dei tre numeri inseriti

- esistono vari modi di utilizzare le istruzioni condizionali per risolvere il problema
- vedremo tre diverse soluzioni

Il programma: la parte comune

```
class MaggioreDiTre{
  public static void main (String[] args){
    InputWindow in = new InputWindow ();
    int a = in.readInt(“Primo numero”);
    int b = in.readInt(“Secondo numero”);
    int c = in.readInt(“Terzo numero”);

    OutputWindow out = new OutputWindow (“Numero Maggiore”);

    …… ( da scrivere )

  }
}
```

Prima soluzione

Una prima soluzione può consistere dei seguenti passi:

- confronta i primi due numeri, *a* e *b*, e memorizza il maggiore dei due in una variabile di appoggio, *maggioreTraAeB*
- confronta il terzo numero, *c*, con la variabile di appoggio *maggioreTraAeB*, e memorizza il maggiore in una nuova variabile, *maggiore*
- visualizza il valore della variabile *maggiore*

Prima soluzione: codice

```
class MaggioreDiTre{
  public static void main (String[] args){
        ……
        int maggiore,        // maggiore dei tre numeri
            maggioreTraAeB; // maggiore tra a e b

        if (a>b)
            maggioreTraAeB = a;
        else
            maggioreTraAeB = b;

        if (maggioreTraAeB > c)
            maggiore = maggioreTraAeB;
        else
            maggiore = c;

        out.writeln (“Maggiore dei tre numeri: " + maggiore);
  }
}
```

Seconda soluzione

Una seconda soluzione può consistere dei seguenti passi:

- confronta *a* con *b*;
  - – se *a* è maggiore di *b*, allora il maggiore è *a* oppure *c* (ripeti il confronto tra *a* e *c*)
  - – se *a* è minore di *b*, allora il maggiore è *b* oppure *c* (ripeti il confronto tra *b* e *c*)

Seconda soluzione: codice

```
class MaggioreDiTre{
  public static void main (String[] args){
        ……
        int maggiore;        // maggiore dei tre numeri

        if (a > b)    // il maggiore è a oppure c
          if (a > c)
               maggiore = a;
          else
               maggiore = c;
        else          // il maggiore è b oppure c
          if (b > c)
               maggiore = b;
          else
               maggiore = c;
        out.writeln (“Maggiore dei tre numeri: ” + maggiore);
  }
}
```

Terza soluzione: codice

```
class MaggioreDiTre{
  public static void main (String[] args){
        ……
        int maggiore;
        if (a>=b && a>=c)
              maggiore = a;
        else if (b>=a && b>=c)
              maggiore = b;
        else
              maggiore = c;
        out.writeln (“Maggiore dei tre numeri: ” + maggiore);
  }
}
```

Osservazioni sulle tre soluzioni

- La prima soluzione è molto chiara da leggere, ma richiede l'uso di una variabile di appoggio
- La seconda soluzione è meno chiara da leggere, ma non richiede l'uso di variabili di appoggio
- La terza soluzione è chiara, compatta e non richiede l'uso di variabili di appoggio
  - la terza soluzione è tuttavia meno efficiente della seconda, e nasconde delle insidie
  - vediamo di seguito alcune varianti della terza soluzione, logicamente non corrette

Terza soluzione: codice errato!

```
class MaggioreDiTre{
  public static void main (String[] args){
        ……
        int maggiore;
        if (a>b && a>c)
               maggiore = a;
        else if (b>a && b>c)
               maggiore = b;
        else
               maggiore = c;
        out.writeln (“Maggiore dei tre numeri: ” + maggiore);
  }
}
```

Perché questo codice non è corretto? In quali circostanze può fornire un risultato sbagliato?

Terza soluzione: codice errato!

```
class MaggioreDiTre{
  public static void main (String[] args){
        ……
        int maggiore;
        if (a>b && a>c)
              maggiore = a;
        else if (b>a && b>c)
              maggiore = b;
        else if (c>a && c>b)
              maggiore = c;
        out.writeln (“Maggiore dei tre numeri: ” + maggiore);
  }
}
```

Perché questo codice non è ancora corretto? Cosa succede se provo a compilare?

Uso di if ed if-else insieme

Supponiamo di voler far inserire due numeri interi *a* e *b* all'utente e di voler soltanto visualizzare uno dei due seguenti messaggi:

- *a* maggiore di *b*
(se *a* risulta strettamente maggiore di *b*)

- *a* uguale a *b*
(se *a* e *b* sono uguali)

Una soluzione errata!

```
……
if (a != b)
  if (a > b)
      out.writeln (“a maggiore di b”);
else
      out.writeln (“a uguale a b”);
```

Quale è l’errore in questo codice? Come si riscrive un codice corretto?

Istruzioni iterative

Molto spesso nei programmi si ha la necessità di ripetere sequenze di istruzioni più volte

- a tal fine Java mette a disposizione tre tipi di istruzioni, note come istruzioni iterative (o ripetitive)
- nel seguito illustriamo queste istruzioni procedendo per esempi

Un primo problema

Supponiamo di voler scrivere un programma che chiede all'utente di inserire un numero intero e che visualizza tutti i numeri pari tra 0 ed il numero inserito (incluso)

- ci serve un modo "automatico" per visitare tutti i numeri tra 0 e quello inserito dall'utente
- non conosciamo questi numeri al momento in cui scriviamo il programma – essi dipendono dal dato dell'utente

Un codice parziale

```
class NumeriPari{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i = 0;    // usata per scandire i numeri in [0,n]
       mentre (i è minore di n+1){
         if (i è pari)
              out.write (i + " ");
         incrementa i di una unità
       }
   }
}
```

ciclo di istruzioni da ripetere fino a quando i < n+1

Il codice Java completo

```
class NumeriPari{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i = 0;
       while (i < n+1){
         if (i%2 == 0)
              out.write (i + " ");
         i++;
       }
   }
}
```

la condizione può essere anche scritta come **(i <= n)**

L’istruzione while: sintassi

L’istruzione while è una istruzione iterativa; ecco la sua sintassi

```
while (condizione){
        istruzione 1
        istruzione 2 ….
        istruzione k
}
```

corpo

• *condizione* deve essere un predicato, cioè una espressione di tipo *boolean*

L’istruzione while: semantica

while (condizione){
istruzione 1
istruzione 2 ….
istruzione k
}

• le istruzioni nel corpo del *while* vengono ripetute fino a quando *condizione* si mantiene *true*

• un ciclo consiste nell’esecuzione di tutte le istruzioni nel *while*

• prima di iniziare un nuovo ciclo, *condizione* viene ricontrollata – se condizione è *true* un nuovo ciclo viene eseguito, altrimenti si esce dal *while* (l’iterazione termina)

Una soluzione alternativa

```
class NumeriPari{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i = 0;
       while (i < n+1){
         out.write (i + " ");
         i += 2;
       }
   }
}
```

la condizione può essere anche scritta come **(i <= n)**

Osservazioni sul while

- Il corpo del *while* può anche consistere di una sola istruzione – in tal caso si possono omettere le parentesi graffe

- nel corpo del *while* dovrebbero esserci delle istruzioni che modificano la condizione, altrimenti il ciclo non terminerebbe mai (loop infinito)

- il corpo del *while* potrebbe anche non essere mai eseguito – la condizione viene controllata anche prima di iniziare il primo ciclo

L’istruzione for

Il problema appena visto può essere risolto anche tramite l’uso di una differente istruzione iterativa di Java, l’istruzione for

- l’istruzione *for* ha una sintassi ed una semantica differenti rispetto all’istruzione *while*
- analizziamo nel seguito sintassi e semantica dell’istruzione *for*

Il codice con l'istruzione for

```
class NumeriPari{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i;
       for (i=0; i<n+1; i++){
         if (i%2 == 0)
              out.write (i + " ");
       }
   }
}
```

L’istruzione for: sintassi

Ecco la sintassi dell’istruzione *for*

```
for (inizializzazione; condizione; aggiornamento) {
    istruzione 1
    istruzione 2 ….
    istruzione k

}
```

corpo

- *condizione* deve essere un predicato
- *inizializzazione* è una istruzione-espressione (tipicamente una assegnazione)
- *aggiornamento* è una istruzione-espressione (tipicamente un incremento o un decremento)

L’istruzione for: semantica

for (inizializzazione; condizione; aggiornamento) {

istruzione 1
istruzione 2 ….
istruzione k

}

corpo

1. esegui *inizializzazione*
2. valuta *condizione* – se vale *true* esegui il corpo, se vale *false* termina il *for*
3. al termine di ogni ciclo esegui *aggiornamento* e torna al punto 2

Una soluzione alternativa

```
class NumeriPari{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i;
       for (i=0; i<n+1; i+=2)
           out.write (i + " ");
   }
}
```

Osservazioni sul for

• ogni parte del *for* è opzionale (cioè può mancare)
– è però buona norma che siano tutte specificate

• la parte *aggiornamento* serve tipicamente ad influenzare la parte *condizione* col passare dei cicli

• le istruzioni nel corpo del *for* tipicamente non influenzano la condizione (cosa invece indispensabile nel *while*)

L’istruzione do-while

Esiste infine una terza istruzione ripetitiva; l’istruzione do-while

- vediamo nel seguito come si può riscrivere il codice per la classe *NumeriPari* usando il *do-while*
- analizzeremo poi in dettaglio la sintassi e semantica dell’istruzione *do-while*

Il codice con il do-while

```
class NumeriPari{
  public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int n = in.readInt ("Inserire un intero positivo");
       OutputWindow out = new OutputWindow ("Numeri Pari");

       int i = 0;
       do{
         if (i%2 == 0)
              out.write (i + " ");
         i++;
       } while (i<n+1);

  }
}
```

L’istruzione do-while: sintassi

Ecco la sintassi dell’istruzione *do-while*

```
do {
    istruzione 1
    istruzione 2 ….
    istruzione k
} while (condizione);
```

corpo

- *condizione* deve essere un predicato

L’istruzione do-while: semantica

```
do {
    istruzione 1
    istruzione 2 ….
    istruzione k
} while (condizione);
```

corpo

- il corpo viene sempre eseguito la prima volta (cioè il primo ciclo è svolto incondizionatamente)
- al termine di ogni ciclo si valuta *condizione* – se è *true* si esegue un nuovo ciclo, altrimenti si termina il *do-while*

Osservazioni sul do-while

• la differenza principale tra *while* e *do-while* sta nel fatto che con il *while* il primo ciclo potrebbe non essere eseguito, mentre *do-while* lo esegue sempre (il controllo della condizione è alla fine)

• il *do-while* è meno usato del *while*, ma ci sono casi in cui può tornare utile

Confronto di istruzioni iterative

Abbiamo visto un problema che poteva essere risolto indifferentemente con una delle tre possibili istruzioni iterative:

- ci sono problemi che richiedono necessariamente l'uso di una specifica istruzione iterativa?
- come si sceglie l'istruzione iterativa da usare?
- rispondiamo nel seguito a queste domande

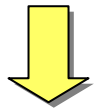
Equivalenza delle istruzioni iterative

Si può mostrare che ogni istruzione iterativa può essere opportunamente espressa in termini delle altre due (eventualmente con l'uso delle istruzioni condizionali)

Mostriamo ad esempio come si può esprimere un *for* in termini di un *while*

Dal for al while

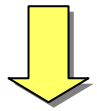
```
for (inizializzazione; condizione; aggiornamento) {
        istruzione 1
        istruzione 2 …
        istruzione k

}
```



```
inizializzazione;
while (condizione) {
        istruzione 1
        istruzione 2 …
        istruzione k
        aggiornamento;

}
```

Dal for al while: esempio

```
for (i = 0; i < n+1; i++) {
        if (i%2 == 0)
                out.write (i + " ");

}
```



```
i = 0;
while (i < n+1) {
        if (i%2 == 0)
                out.write (i + " ");

        i++;

}
```

Scelta dell'istruzione iterativa

Se le istruzioni iterative hanno tutte lo stesso potere espressivo, come si sceglie quella più adatta da usare?

- non esistono regole, ma l'esperienza aiuta
- tipicamente il *while* è molto usata, mentre il *do-while* si usa di meno (è meno flessibile)
- il *for* si usa molto nei casi in cui si deve effettuare un numero di cicli che può essere calcolato in anticipo (come nel problema visto)

Esempi di uso

Vediamo nel seguito altri semplici esempi di uso delle istruzioni iterative

- gli esempi ci aiuteranno a capire come scegliere l'istruzione da usare
- talvolta la scelta non influisce troppo, ed è pertanto una questione di "gusti personali"

Esempio 1

Vogliamo scrivere un semplice programma che:

- chiede all'utente di inserire due numeri interi *a* e *b*, tali che *a*<*b*
- visualizza all'utente la somma di tutti i numeri interi compresi nell'intervallo [*a*,*b*]

Esempio 1: scelta dell'istruzione

Dopo che l'utente ha inserito *a* e *b*, sono in grado di conoscere con certezza il numero di iterazioni richieste per calcolare la somma?

- debbo sommare tutti i numeri compresi tra *a* e *b*, e quindi dovrò eseguire *b*-*a*+1 istruzioni di somma
- conviene utilizzare una istruzione *for*

Esempio 1: il codice Java

```
class SommaNumeri{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int a = in.readInt ("Inserire un intero");
       int b = in.readInt ("Inserire un intero > " + a);
       OutputWindow out = new OutputWindow ("Somma Numeri");

       int somma = 0;       // memorizza la somma
       int i;
       for (i=a; i<=b; i++)
              somma += i;

       out.writeln (“Somma dei numeri = ” + somma);
   }
}
```

Esempio 1: una variante

```
class SommaNumeri{
   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       int a = in.readInt ("Inserire un intero");
       int b = in.readInt ("Inserire un intero > " + a);
       OutputWindow out = new OutputWindow ("Somma Numeri");

       int somma = 0;       // memorizza la somma
       for (int i=a; i<=b; i++)
            somma += i;

       out.writeln (“Somma dei numeri = ” + somma);
   }
}
```

La variabile *i* si può dichiarare direttamente nella parte inizializzazione del *for* – in questo modo però non sarà visibile fuori del corpo del *for*

Esempio 2

Vogliamo scrivere un semplice programma che chiede all'utente di inserire una stringa *s*:

- se la stringa *s* inserita è diversa dalla parola "FINE", allora visualizza all'utente la lunghezza di *s*, e poi gli chiede di inserire nuovamente una stringa *s*
- se la stringa *s* inserita è uguale alla parola "FINE", allora il programma termina

Esempio 2: scelta dell'istruzione

Il programma deve far inserire ripetutamente una stringa all'utente, e si deve arrestare quando l'utente inserisce "FINE"

- non si può conoscere a priori il numero di iterazioni
- in questo caso il *while* o il *do-while* sono la scelta più indicata

Esempio 2: il codice Java

```
class LunghezzaStringhe{
    public static void main (String[] args){
       InputWindow in = new InputWindow ();
       OutputWindow out = new OutputWindow ();

       String s = in.readString(); // prima lettura
       while (!s.equals("FINE")){
              out.writeln ("Lunghezza = " + s.length());
              s = in.readString();
       }
       out.writeln (“Fine programma”);
    }
}
```

Esempio 2: uso del do-while

```
class LunghezzaStringhe{
    public static void main (String[] args){
       InputWindow in = new InputWindow ();
       OutputWindow out = new OutputWindow ();

       String s;
       do{
         s = in.readString();
         if (!s.equals("FINE"))
              out.writeln ("Lunghezza = " + s.length());
       }while (!s.equals("FINE"));
       out.writeln ("Fine programma");
    }
}
```

Esempio 2: uso del do-while (2)

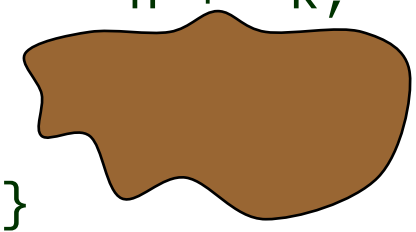
```
class LunghezzaStringhe{
    public static void main (String[] args){
       InputWindow in = new InputWindow ();
       OutputWindow out = new OutputWindow ();

       String s;
       boolean fine;
       do{
         s = in.readString();
         fine = s.equals("FINE");
         if (!fine)
              out.writeln ("Lunghezza = " + s.length());
       }while (!fine);
       out.writeln ("Fine programma");
    }
}
```

Un ultimo esercizio

Considera il seguente frammento di codice, su cui, purtroppo, è caduto del cioccolato (parte di esso non è leggibile)

–assumendo che il *while* sicuramente terminerà, puoi dire cosa viene visualizzato?

```
int n, k;
n = 19;
k = 3;
while (n!=7) {
    n += k;
}
System.out.println(n);
```



Glossario dei termini principali

Termine	Significato
Istruzioni di controllo	Insieme delle istruzioni condizionali e delle istruzioni iterative
Istruzioni condizionali	Istruzioni utilizzate per verificare il valore di predicati e creare differenti rami di esecuzione (istruzioni if e if-else)
Istruzioni iterative	Istruzioni utilizzate per ripetere più volte sequenze di istruzioni (istruzioni while, for e do-while)
Parte if	Blocco di istruzioni eseguite se la condizione di un if è true
Parte else	Blocco di istruzioni eseguite se la condizione di un if è false
Blocco di istruzioni	Sequenza di istruzioni tra parentesi graffe
Cascate di if-else	Sequenze annidate di istruzioni if-else
Ciclo	Esecuzione del corpo di una istruzione iterativa